David Duenas-Cid · Peter Roenne ·
Melanie Volkamer ·Michelle Blom ·
Pierrick Gaudry · Isabelle Borucki ·
Leontine Loeber · Alexandre Debant (Eds.)

LNCS 16028

# Electronic Voting

10th International Joint Conference, E-Vote-ID 2025
Nancy, France, October 1–3, 2025
Proceedings



Springer

OPEN ACCESS

**Lecture Notes in Computer Science** **16028**

The series Lecture Notes in Computer Science (LNCS), including its subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics (LNBI), has established itself as a medium for the publication of new developments in computer science and information technology research, teaching, and education.

LNCS enjoys close cooperation with the computer science R & D community, the series counts many renowned academics among its volume editors and paper authors, and collaborates with prestigious societies. Its mission is to serve this international community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings. LNCS commenced publication in 1973.

David Duenas-Cid · Peter Roenne ·
Melanie Volkamer · Michelle Blom ·
Pierrick Gaudry · Isabelle Borucki ·
Leontine Loeber · Alexandre Debant
Editors

# Electronic Voting

10th International Joint Conference, E-Vote-ID 2025
Nancy, France, October 1–3, 2025
Proceedings

Springer

*Editors*
David Duenas-Cid
Kozminski University
Warsaw, Poland

Peter Roenne
University of Luxembourg
Esch-sur-Alzette, Luxembourg

Melanie Volkamer
Karlsruhe Institute of Technology
Karlsruhe, Germany

Michelle Blom
University of Melbourne
Melbourne, VIC, Australia

Pierrick Gaudry
Université de Lorraine
Nancy, France

Isabelle Borucki
Philipps-University Marburg
Marburg, Germany

Leontine Loeber
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands

Alexandre Debant
Inria
Villers-lès-Nancy, France

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-032-05035-9 ISBN 978-3-032-05036-6 (eBook)
https://doi.org/10.1007/978-3-032-05036-6



This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

This volume contains a selection of papers presented at E-Vote-ID 2025, the Tenth International Joint Conference on Electronic Voting, held on October 1–3, 2025. This is the first time the conference traveled to France (Nancy), hosted at LORIA (Lorraine Research Laboratory in Computer Science and its Applications) and Inria (French National Institute for Research in Digital Science and Technology).

The E-Vote-ID Conference resulted from merging EVOTE and Vote-ID, and dates back more than 20 years since the first E-Vote conference in Austria. Since that conference in 2004, over 2000 experts have attended the venue, including scholars, practitioners, representatives of various authorities, electoral managers, vendors, and PhD students. The conference collected the most relevant debates on the development of Electronic and Internet Voting and Electoral Technologies, from aspects relating to security and usability to practical experiences and applications of voting systems, also including legal, social, or political aspects, amongst others, turning out to be an important global reference point concerning these issues.

This year, as in previous editions, the conference consisted of: Security, Usability, and Technical Issues Track; Governance of E-Voting Track; Election and Practical Experiences Track; PhD Colloquium; and Poster and Demo Session.

E-VOTE-ID 2025 received 39 submissions for consideration in the first two tracks (Technical and Governance Tracks), each being reviewed by 3 to 5 Program Committee members using a double-blind review process. As a result, 13 papers were accepted for this volume, representing 33% of the submitted proposals. The selected papers cover a wide range of topics connected with electronic voting, including experiences and revisions of the actual uses of E-voting systems and corresponding processes in elections.

We would like to thank the local organizers, Pierrick Gaudry and Alexandre Debant, for their excellent collaboration in preparing the conference. Special gratitude goes to KASTEL Security Research Labs for granting Open Access to these proceedings. Gratitude also goes to the Swiss Federal Chancellery, to the Université de Lorraine, and to the France 2030 program, who sponsored the event. Finally, we would like to thank and appreciate the international program committee members for their hard work in

reviewing, discussing, and shepherding papers. They ensured, once again, the excellence of these proceedings with their knowledge and experience.

October 2025

David Duenas-Cid
Peter Roenne
Melanie Volkamer
Michelle Blom
Pierrick Gaudry
Isabelle Borucki
Leontine Loeber
Alexandre Debant

# Organization

## Program Committee

| | |
|---|---|
| Marta Aranyossy | Corvinus University of Budapest, Hungary |
| Myrto Arapinis | University of Edinburgh, UK |
| Roberto Araujo | Universidade Federal do Pará, Brazil |
| Jordi Barrat i Esteve | eVoting Legal Lab, Spain |
| Bernhard Beckert | Karlsruhe Institute of Technology, Germany |
| Josh Benaloh | Microsoft, USA |
| Matthew Bernhard | University of Michigan, USA |
| Jurlind Budurushi | Baden-Wuerttemberg Cooperative State University, Karlsruhe, Germany |
| Jeremy Clark | Concordia University, Canada |
| Cesar A. Collazos | Universidad del Cauca-Colombia, Colombia |
| Régis Dandoy | Universidad San Francisco de Quito, Ecuador |
| Staffan Darnolf | International Foundation for Electoral Systems, USA |
| Constantin Catalin Dragan | University of Surrey, UK |
| Alexander Ek | Monash University, Australia |
| Aleksander Essex | University of Western Ontario, Canada |
| Diego F. Aranha | Aarhus University, Denmark |
| Rosa Fernández Riveira | Universidad Complutense de Madrid, Spain |
| Tamara Finogina | Universitat Politècnica de Catalunya, Spain |
| Holly Ann Garnett | Royal Military College of Canada, Canada |
| Micha Germann | University of Bath, UK |
| J. Paul Gibson | Mines Telecom, France |
| Rosario Giustolisi | IT University of Copenhagen, Denmark |
| Kristian Gjosteen | Norwegian University of Science and Technology, Norway |
| Amanda Glazer | University of Texas at Austin, USA |
| Nicole Goodman | University of Toronto, Canada |
| Rajeev Gore | Monash University, Australia |
| Ruediger Grimm | University of Koblenz, Germany |
| Rolf Haenni | Bern University of Applied Sciences, Switzerland |
| Thomas Haines | Australian National University, Australia |
| Thad Hall | Mercer County Elections, USA |
| Bart Jacobs | Radboud University, Netherlands |
| Hugo Jonker | Open University of the Netherlands, Netherlands |

| | |
|---|---|
| Mehmet Sabir Kiraz | De Montfort University, UK |
| Michael Kirsten | Karlsruhe Institute of Technology (KIT), Germany |
| Reto Koenig | Berne University of Applied Sciences, Switzerland |
| Ralf Kuesters | University of Stuttgart, Germany |
| Andreas Mayer | Hochschule Heilbronn, Germany |
| Michael McGregor | Toronto Metropolitan University, Canada |
| Magdalena Musial-Karg | Adam Mickiewicz University |
| Michael Naehrig | Microsoft, USA |
| Stephan Neumann | Landesbank Saar, Germany |
| Hannu Nurmi | University of Turku, Finland |
| Olivier Pereira | UCLouvain, Belgium |
| Thomas Peters | Université catholique de Louvain, Belgium |
| Ismael Peña-López | Universitat Oberta de Catalunya, Spain |
| Pascal Reisert | University of Stuttgart, Germany |
| Stefan Rosemann | Federal Office for Information Security, Germany |
| Mark Ryan | University of Birmingham, UK |
| P. Y. A. Ryan | University of Luxembourg, Luxembourg |
| Peter Sasvari | National University of Public Service, Hungary |
| Steve Schneider | University of Surrey, UK |
| Carsten Schuermann | IT University of Copenhagen, Denmark |
| Uwe Serdült | Ritsumeikan University, Japan |
| Rodney Smith | University of Sydney, Australia |
| Mihkel Solvak | University of Tartu, Estonia |
| Jacob Spertus | University of California, Berkeley, USA |
| Iuliia Spycher | University of Bern, Switzerland |
| Philip Stark | University of California, Berkeley, USA |
| Peter J. Stuckey | Monash University, Australia |
| Vanessa Teague | Thinking Cybersecurity, Australia |
| Tomasz Truderung | University of Trier, Germany |
| Siim Trumm | University of Nottingham, UK |
| Priit Vinkel | E-governance Academy, Estonia |
| Felix-Christopher von Nostitz | Université catholique de Lille, France |
| Damjan Vukcevic | Monash University, Australia |
| Roland Wen | University of New South Wales, Australia |
| Jan Willemson | Cybernetica, Estonia |
| Filip Zagorski | University of Wroclaw, Poland |

# Contents

# MERGE: Matching Electronic Results with Genuine Evidence, for Verifiable Voting in Person at Remote Locations

Ben Adida[1(✉)], John Caron[1,2], Arash Mirzaei[2], and Vanessa Teague[2]

[1] Voting Works, San Francisco, USA
ben@adida.net
[2] Australian National University, Canberra, Australia
vanessa.teague@anu.edu.au

**Abstract.** Overseas military personnel often face significant challenges in participating in elections due to the slow pace of traditional mail systems, which can result in ballots missing crucial deadlines. While internet-based voting offers a faster alternative, it introduces serious risks to the integrity and privacy of the voting process. We introduce the MERGE protocol to address these issues by combining the speed of electronic ballot delivery with the reliability of paper returns. This protocol allows voters to submit an electronic record of their vote quickly while simultaneously mailing a paper ballot for verification. The electronic record can be used for preliminary results, but the paper ballot is used in a Risk Limiting Audit (RLA) if received in time, ensuring the integrity of the election. This approach extends the time window for ballot arrival without undermining the security and accuracy of the vote count.

## 1 Introduction

Slow mail is one of the main barriers to electoral participation for many overseas military personnel. Some jurisdictions send blank ballots by paper mail and require ballot return by the same channel—this can be very slow, often resulting in ballots that are not returned by the deadline. Other jurisdictions allow for paperless voting by Internet, including email or pdf upload—this is very fast, but introduces unacceptable risks to integrity and privacy. One compromise is

Author is listed alphabetically.

CAC-Vote project—This is a protocol proposal for a research project. Like any security protocol, it requires substantial supporting assumptions and processes in order to be secure - there is no claim that any US jurisdiction has, or will have, the necessary supporting legislation or processes to run it as specified. This research was, in part, funded by the U.S. Government. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. Distribution Statement A - Approved for public release: distribution is unlimited.


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 1–16, 2026.
https://doi.org/10.1007/978-3-032-05036-6_1

electronic delivery and paper returns, in which voters download a blank ballot and mail back a printed paper vote, having filled it in either manually or electronically. This has similar integrity and privacy properties to vote-by-mail, and takes only half the time that a fully paper solution would take. Nevertheless, the return mail delivery is often still too slow.

This paper describes the MERGE protocol, which enhances the basic scheme of electronic delivery and paper returns in a way that increases speed without significantly compromising privacy or integrity. The main idea is to send an untrusted electronic record back quickly, which can be confirmed if the paper ballot arrives in time for the Risk Limiting Audit (RLA), or distrusted otherwise. The RLA is a post-election auditing process that uses statistical methods to ensure with high confidence that the reported election outcome is correct, by manually comparing a random sample of paper ballots against the electronic vote tallies. Since RLA normally happens a few days to a few weeks after the election day, this extends the time available for the paper ballot to arrive.

Assume a public bulletin board (BB) which is an un-erasable, authenticated broadcast channel with memory. We will extensively use the assumption that people in different locations can rely on seeing the same data on the BB.

- In the polling place, the voter makes both a verifiable paper record, which is placed inside an envelope and mailed, and an untrusted electronic record, which is encrypted and posted to the BB.
- The voter is issued a digital signature, presented on a sticker, which serves as verifiable evidence of participation. Envelopes are considered acceptable only if they bear stickers with valid signatures upon receipt. This also allows officials at the Local Counting Center to notify the voter when their properly-signed envelope has arrived.[1]
- The untrusted electronic record can be incorporated into preliminary results and used as the cast vote record in an RLA.
- If the paper ballot arrives in time for the RLA, it is used as the ballot, just like any other RLA.
- If no paper ballot arrives that corresponds to a given electronic record, we use the phantoms-to-zombies approach of Bañuelos and Stark [2] to ensure that the risk limit is met even though the electronic record is not trusted.

Two assumptions are unavoidable: first, there must be an accurate count of people who legitimately participated; second, the paper ballot the voter places into the envelope must accurately reflect their intentions. Our protocol also requires someone to check the digital signature on every envelope. These all need to be supported by human verification processes.

[1] This is expected to be part of the practical implementation. It does not, however, form an important part of the cryptographic protocol because it does not include evidence that the paper ballot matched the electronic one. The protocol could be extended, with some extra mixing, to notify each voter of whether their specific ballot matched. The current version simply produces a collective tally of how many of the received ballots matched.

The paper and electronic records could be produced by scanning a hand marked paper ballot, or printing a paper ballot from a ballot marking device (BMD).[2] See Sect. 3 for a detailed discussion of the trust assumptions.

Compared to plain electronic delivery and paper returns, MERGE's main advantage is that, rather than needing ballot papers to arrive in time to be scanned for preliminary results, a vote can be safely counted if it arrives before the RLA. Of course, an electronic commitment to the vote must still be made before the normal close of polls. In practice, for most jurisdictions, this allows a few more weeks for mail to arrive. Its main disadvantage is that—if a large number of ballot papers do not arrive in time—the consequent large discrepancies may cause an RLA to fall back to a manual recount, when it would not have if those votes had simply been excluded from the beginning.

MERGE achieves *Software Independence* [9] based on a collective, probabilistic method of cast-as-intended verification designed to fit in to an existing Risk Limiting Audit. We assume that the attacker can control any electronic components, but at least one of the devices chosen for each verification step is honest. See Subsect. 1.3.

The election occurs in two classes of locations.

**Remote Voting Centers** are controlled polling places in remote areas such as overseas military bases, where voters are able to vote privately without coercion. There are clear instructions (e.g. on a poster on the wall) that do not come from the device used for voting, and some process for ensuring that the protocols assumptions are met.

**Local Counting Centers** are state- or county-based electoral authorities, where the ordinary votes from most citizens are counted. These will also receive paper ballots from Remote Voting Centers and process them.

There are likely to be many Remote Voting Centers and many Local Counting Centers, but we assume only one of each in this paper for simplicity.

*Notation:* References like Full-X refer to the full version.

### 1.1 Authentication Using CAC Cards

Voters at Remote Voting Centers own a smart card called a Common Access Card (CAC) card. It provides a Public Key Infrastructure (PKI) to enable secure authentication and digital signatures. Each CAC card includes a (private) signing key and a certificate linking its ID to the corresponding public key. Observers and officials at each Local Counting Center know the CAC IDs and certificates of the voters to be included. This knowledge is obtained either through individual CAC voters, who must register their public keys with their local electoral

[2] Scientific opinion is divided on whether human verification of BMDs is sufficiently accurate to justify the assumption that the printout matches the voter's intention. See [1,8] for examples of opposing views. At a minimum, this requires careful design to ensure voters are motivated and encouraged to verify, and there is something they can do if a misprint occurs.

authorities, or via a more centrally organized process. Only votes with a valid CAC signature are accepted via the electronic channel. Changes to CAC cards during the protocol, such as name changes, eligibility changes or key refreshes between the time the person votes and the time the votes are tallied, are out of scope for this paper but will need to be handled with specific policies in practice.

### 1.2 MERGE Within the Voting Ecosystem

We assume that most votes are not from MERGE, but are cast locally near the Local Counting Center. The usual audit process is conducted there, with MERGE ballots incorporated from Remote Voting Centers as needed.

The ballot manifest—available to observers at the Local Counting Center—includes both the ordinary local ballots and all the ballots from Remote Voting Centers. So do the preliminary Cast Vote Records (CVRs).

The RLA proceeds exactly as it would if all the votes had been cast locally: observers see the ballot manifest and preliminary CVRs, then they watch a transparent process for seeding the PRNG that will be used to generate ballot samples, then they check the sequence of sampled ballots. For local ballots, CVRs are made in the usual way (e.g. by scanning); for MERGE ballots, preliminary CVRs come from the BB in encrypted form. If the sampled ballot is local, officials retrieve it in the usual way and compute its effect on the audit statistics. If the ballot was cast remotely, it will require some extra work to locate the mailed ballot and compare it with its electronic counterpart. It is also possible that the ballot was delayed in the mail—we deal conservatively with this situation. Figure 1 shows how MERGE fits into the voting and auditing process.

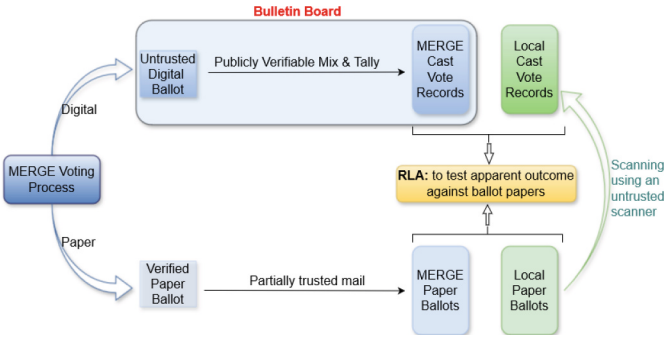


**Fig. 1.** Overview of MERGE in the voting ecosystem.

### 1.3 Security Properties and Contributions

– **Authentication:** Only eligible voters can vote.

- **Privacy:** The protocol does not reveal more about an individual's vote than the published tallies do. See Sect. 5 and Full-C.2.
- **Receipt Freeness:** It is infeasible for a voter to convince anyone of the value of their vote, even if they actively collude with the coercer, unless the coercer observes the serial numbers during the audit process. MERGE reveals who participated, so it is possible to coerce someone to refrain from participating altogether. See Full-5.3.
- **(Individual) paper-based cast-as-intended verification:** Each voter can see that their paper vote accurately reflects their intention and refuse to cast it if it does not.
- **(Public) tally verification:** anyone can verify that the recorded electronic votes are correctly tallied.
- **(Collective, probabilistic) recorded-as-cast verification:** if the apparent election outcome (incorporating the electronic tally) does not accurately reflect the paper evidence (including both MERGE and ordinary ballots), the RLA will fall back to a full manual count except with probability at most the risk limit.

The formalization and proof of the last property is a major contribution of this paper: we employ a game-based approach to demonstrate that MERGE can interface with an existing RLA procedure, ensuring that the *overall* risk limit is still met. While [7] also uses a "physical cryptography game" framework to analyze RLAs in adversarial contexts, our work extends this approach to a complete e-voting protocol, addressing the complexities of processing both electronic and paper ballots prior to the RLA process. This property is different from (and neither weaker nor stronger than) end-to-end verifiability: although individuals cannot verify that *their* vote is properly recorded on the BB, the system can instead guarantee a global limit on the risk that enough votes are misrecorded to alter the outcome.

Our open-source prototype implementation includes all of the protocol except the Bulletin Board. It includes prototype libraries for verifiable mixing and an elliptic-curve implementation of ElectionGuard. Details are in Full-7.

Guarantees achieved in this paper rely entirely on the trustworthiness of the paper trail, which is outside the control of the cryptographic protocol, and may fail for a variety of reasons, such as security problems in the mail channel and difficulties verifying printouts. These need to be addressed by human procedures outside the MERGE protocol.

### 1.4 Related Work

There are many protocols for remote electronic voting, some with very strong security properties of various kinds. However, there is no single protocol that combines good privacy, receipt freeness (even against an attacker who can see only the bulletin board), public verifiability, and voter verification usable enough to constitute a good basis for trust. Since we work in a controlled setting, direct voter verification of a plaintext paper vote has substantial advantages, but the

remote setting introduces practical challenges. This is a relatively under-studied area—most designs focus on either an uncontrolled Internet setting, or an in-person setting in the normal polling place.

Many systems provide end-to-end verifiability for polling-place voting and combine it with plain paper verification. Most could be adapted to remote, controlled polling places, but do not have detailed designs for dealing with ballots that are lost in the mail. An exception is ElectionGuard [3], which includes a vote-by-mail design, but this is intended for situations where preprinted ballots can be sent to voters and returned. It does not attempt to solve the timing issues associated with two-way mail. "Verifiable and Private vote-by-mail" [6] adds cryptographic verifiability to the electronic delivery and paper return strategy, much like MERGE. The main difference is that there is no pre-emptive electronic record sent, so no opportunity to take advantage of a timing difference between initial canvas and audit.

## 2 Technical Background and Notation

### 2.1 Cryptographic Components

*ElectionGuard.* ElectionGuard [4] is a suite of open-source tools to support end-to-end verifiable voting. It uses a variant exponential form of the ElGamal cryptosystem to encrypt a vote. See Full-B.1 for details. This allows:

- homomorphic addition of encrypted values, enabling the decryption of only the overall election tally,
- re-randomization of an encrypted value, which produces a ciphertext indistinguishable from a fresh one,
- threshold distributed decryption, which prevents any set of fewer than a threshold of talliers from colluding to decrypt individual votes.

Our design uses the ElectionGuard toolkit and relies on the paper record rather than trying to gain trust in the electronic record, with some extra details to allow for the sorts of failures that can happen when votes are mailed.

Homomorphic addition is written like multiplication on ciphertexts, either $\cdot$ for a pair or $\Pi$ for a product of multiple values. The encryption of vote vector $\mathbf{b}$ is denoted by $\mathbf{e} = E(\mathbf{b})$.

*Non-Interactive Zero-Knowledge (NIZK) Proofs.* The election transcript includes Noninteractive Zero Knowlege Proofs (NIZKPs) that allow for public verification that input votes are valid and that all decryptions are computed properly.

A NIZK proof of validity for an encrypted vote vector $\mathbf{e} = E(\mathbf{b})$ is denoted by $ZKP_{\text{VALID}}(\mathbf{b})$. It proves:

- the encryption for each option (e.g. $e_j$) encodes a valid value, usually either zero or one (i.e. $b_j \in \{0, 1\}$), and

– the sum of all encrypted options within each contest falls within a specific range (determined by the voting rules).

We also use ElectionGuard's NIZK proofs to demonstrate that an encrypted value (e.g. $e = E(b)$) is decrypted to a specific value (e.g. $b$), without revealing the decryption key. This proof is denoted by $ZKP_{\text{DEC}}(b, e)$.

*Private Decryption.* Sometimes a plaintext, with a proof of proper decryption, will be supplied to some participants but not published on the BB. We call this *private decryption.*

*Mixnet.* To maintain the confidentiality of each voter's ballot, a mixnet is used to mix it with others. It takes a set of encrypted ballots and outputs a shuffled set of re-encrypted ballots, making it challenging to link any specific encrypted ballot in its output to any of its input encrypted ballots. The mixnet consists of multiple mixing servers arranged in sequence, each operated by an independent mixing authority, ensuring that as long as at least one server is honest, the confidentiality of the ballots is preserved. The mixnet operation on the input encrypted ballot set $\{E(\mathbf{b_i})\}_i$ is denoted by $Mix(\{E(\mathbf{b_i})\}_i)$.

The mixnet proves that its input $\mathcal{I}$ (a set of vote vectors) is correctly shuffled and re-encrypted to $\mathcal{O}$, without revealing any information about the permutation from $\mathcal{I}$ to $\mathcal{O}$. This proof is denoted by $ZKP_{\text{MIX}}(\mathcal{I}, \mathcal{O})$

*Digital Signature.* The digital signature on message $m$ by voter $i$ is represented as $sig_i(m)$. Most often we produce both the message and its signature, which we denote by $auth_i(m)$, meaning $sig_i(m)$ concatenated with $m$.

*Hash Function.* $H(\cdot)$ denotes the hash of message $m$ using a collision-resistant cryptographic hash function (e.g., SHA-256).

*Serial Number.* Each ballot includes a unique serial number $sn$ which is randomly generated. The role of the serial number is to allow one-to-one matching between the paper ballot and its electronic counterpart.

### 2.2 Risk Limiting Audit

A *Risk Limiting Audit* (RLA) tests whether a trustworthy set of paper ballots implies that the announced election winner(s) won, and corrects the result by a full hand count if not. It is parameterized by a *risk limit* $\alpha$ and guarantees that if the announced election result is wrong, the RLA will progress to a full hand count with probability at least $1 - \alpha$. RLAs were developed by Stark and others and apply to a wide variety of election types and audit situations, including plurality contests, supermajority elections, party-list proportional elections and Instant Runoff Voting. A general framework is given in [10].

The *apparent outcome* is (a set of) announced winner(s). The *actual outcome* is the outcome that would be found if all the ballot papers were correctly counted.

A *ballot manifest* is a catalogue describing which paper ballots are present at which physical location. The apparent outcome is supported by a set of *cast vote records* (CVRs), which may or may not accurately reflect the voter's intentions. In this work we mostly concentrate on *ballot-level comparison audits* in which each individually sampled ballot paper is compared with its corresponding CVR. (Using MERGE with other audit styles is discussed in Full-6.) If the CVR overstates a winner's tally compared with the paper record, it is an *overstatement* (for example, if the CVR is a vote for the winner and the paper ballot is a vote for the loser, it is a two-vote overstatement). If the CVR understates a winner's tally compared with the paper ballot, it is an *understatement*.

An RLA technique that is particularly useful in our setting is the phantoms-to-zombies approach devised by Bañuelos and Stark [2]. This is a way of dealing with ballots that appear in the manifest but cannot be found on paper, a problem that may occur frequently when MERGE ballots are delayed or dropped in the mail. The idea is to "Pretend that the audit actually finds a ballot, an evil zombie ballot that shows whatever would increase the risk value the most." Bañuelos and Stark prove this to be conservative, in the sense that the RLA property still holds, assuming that it would have held had the correct ballot paper been located. The correct evil zombie ballot might be slightly different depending on the kind of RLA, but is generally a vote for the highest loser, or possibly an imaginary vote for all the losers, or (in the case of separate comparisons between the winner and each loser) a vote for whichever loser is being compared.

There are various RLA approaches, each with unique properties, defined by different mathematical functions for deciding when to accept the outcome. The following definition outlines their general functionalities and risk limit properties.

**Definition 1.** ***RL Functionality***

***Setup parameters:*** *An apparent outcome, a ballot manifest, a risk limit $\alpha$, and optionally the cast vote records.*

***Input:*** *A sample of ballots (their manifest ID and vote contents), optionally indexed by their cast vote record.*

***Procedure:*** *Perform one of the following actions:*

- *Output "accept" and halt, or*
- *Output "reject" and halt, or*
- *Take a new sample of ballots, combine it with the previous sample, update risk calculations and repeat.*

***Risk Limit Property:*** *For any setup parameters with an incorrect apparent outcome, the probability that the above procedure outputs "accept" over properly generated random samples is at most $\alpha$.*

It is important to understand that applying an RL Functionality does not necessarily imply running a valid Risk Limiting Audit. For example, if the ballot papers are not an accurate representation of the voters' intent, if they have not been securely stored, if the CVRs were not properly committed before the random sample was taken, or the sampled ballots were not honestly randomly

generated, the audit may not actually limit risk. The exact definition of a valid risk limiting audit is outside the scope of this paper. We can, however, prove that our RL functionality maintains the risk limit property, which implies that it can be fitted in to an existing Risk Limiting Audit process.

*Remark 1.* The risk limit property holds regardless of how the wrong outcome is constructed: the adversary may choose any margin, or any way to distribute the wrong CVRs, but must commit to CVRs before samples are chosen randomly.

## 3 Threat Model and Trust Assumptions

The trust assumptions are as follows.

1. The local electoral authorities maintain a list of CAC IDs assigned to registered eligible voters within their county.
   - Each voter's CAC card securely stores the corresponding private key.
   - The CAC Certificate Authority issues trustworthy certificates linking each CAC ID to its public key(s).
   - The local electoral authorities validate the CAC certificate for each signed ballot posted to the BB.[3]
2. Each voter verifies that the paper ballot accurately reflects their intention.
3. Each printed signature on the sticker is verified before sending, either by the voter or by some other trustworthy assistant at the Remote Voting Center.
4. The mailing addresses on the envelopes are correct[4].
5. The voter instructions have to come from some trustworthy source, e.g. a poster on the wall, and not from the BMD itself.

The below additional assumptions are relevant only for privacy.

6. The voter remains hidden from others within the confines of the voting booth. The privacy adversary's visibility is restricted to observing
   - the BB,
   - other information remembered or captured by the voter, which might be susceptible to forgery.
7. Fewer than a pre-determined threshold of tallying authorities are dishonest.
8. At least one of the mixing authorities is honest.

Integrity also depends on an accurate count of the number of voters who participated—the procedures for achieving this are described in Full-4.6.

[3] In principle, the certificates could be posted on the BB with the signatures. However, in practice CAC certificates contain significant personal information that precludes public distribution. We therefore need to assume that they are verified by local authorities but not the public.

[4] This assumption can hold if: the sender knows the intended destinations, there is a supply of preprinted trustworthy envelopes, or an authority verifies the address printed on the envelope by the BMD.

### 3.1 Attacker Model

The adversary may control all computers (except at least one chosen for verification) and do any polynomial-time computation. Any envelope, mailed by a legitimate sender, may be selected by the adversary to experience extended delays or even go missing. However, if delivered, the envelope and its contents are unaltered. The adversary is unable to read the contents of a mailed envelope.

Consequently, the attacker can manipulate a compromised voting machine with the voter's CAC card to sign any arbitrary message. We therefore do not make it harder for this *internal* attacker to stuff a valid-looking paper ballot into the paper mail than it would have been with traditional postal mail. If the jurisdiction was diligently checking each voter's handwritten signature, or if some other form of registered or secure mail is being used, our protocol does not undermine it, but nor do we add to the defences. So we simply assume that *something* prevents the internal attacker from adding fraudulent mail ballots.

MERGE does not claim to defend against paper ballot stuffing.

## 4 MERGE protocol

This section describes the protocol. The example given here uses a BMD to print a paper record, but the protocol would work just as well with a hand-marked paper ballot interpreted by a scanner. In that case, the "voting computer" would be a scanner with the capacity to produce an encrypted record and upload it to the BB. The serial numbers could either be generated by the scanner, or printed in advance. We will use the term "voting computer" or "voting machine" to mean either a BMD or a scanner with computation and communication capacity.

### 4.1 Voting Process for Voter $i$

The voting process from the voter's perspective is described below.

1. The voting machine displays the options to voter $i$.
2. The voter interacts with the machine to make their selection, vote $\mathbf{b}_i$.
3. The voting machine assigns a unique serial number $sn_i$ to this vote. The voting machine produces:
   - a signed, encrypted electronic record and serial number, with a proof of ballot validity, for the BB:

$$BB \hookleftarrow auth_i(E(sn_i), E(\mathbf{b_i}), ZKP_{\text{VALID}}(\mathbf{b_i})),$$

   - a plain paper record of the ballot $(sn_i, \mathbf{b_i})$, which is placed in an envelope,
   - a sticker that includes an address, a traditional mail tracking number, a space for a pen-and-ink signature (if required by regulation—not relevant to our cryptographic protocol) and a digital signature $auth_{Voting\ Computer}(Id_i, Id_e, H_i)$, where $Id_i$ is voter $i$'s unique identifier, $Id_e$ is the election identifier, and $H_i$ is the hash value created from the digital record that is stored for voter $i$, i.e. $H_i = H(auth_i(E(sn_i), E(\mathbf{b_i}), ZKP_{\text{VALID}}(\mathbf{b_i})))$.

The voter must verify that her plaintext printout matches her intention, then stick the sticker on her envelope and mail it. Someone also needs to use an electronic device, whether their own or a third party's, to

- verify the digital signature and the signed data on the sticker, and
- perform recorded-as-cast verification using the "BB Inclusion Check" or "Sticker BB Upload" to check that the electronic record was included on the BB. (See Full-B.3.)

After the voting period ends, electronic records undergo a series of processing steps. Additionally, paper ballots are sent to the Local Counting Center for auditing purposes. A diagram illustrating the whole process is in Fig. 2.

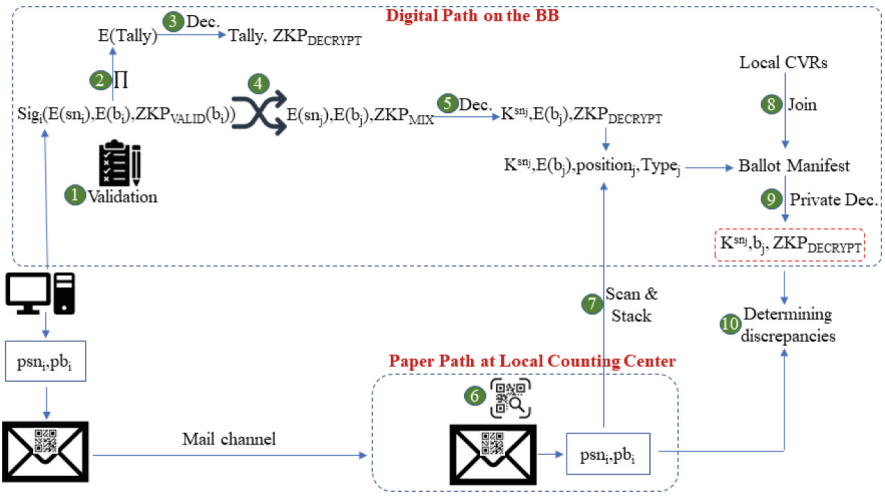


**Fig. 2.** The whole process for MERGE based on ballot comparison. Each Local Counting Center's votes are dealt with separately—the diagram shows the process for only one Local Counting Center.

### 4.2 Digital Path

We explain digital processing of electronic ballots for one Local Counting Center's votes. Every other Local Counting Center's data is processed similarly—they do not interact because their results need to be processed by the appropriate Local Counting Center, even for statewide contests.

At the conclusion of the voting process, each jurisdiction processes the votes on the BB corresponding to CAC IDs registered in that jurisdiction.

The digital record undergoes the following processing steps.

4. The validity of the signature and ZKP for each electronic ballot on the BB is verified. It is also verified that no two ballots on the bulletin board contain a common ciphertext. If this occurs, it indicates serious misbehaviour by a voting computer—the process stops. If any two valid ballots come from the same CAC ID, the first one is selected and the rest are withheld from any further processing.[5]
5. The digital record on the BB now contains the information of all $n$ voters. The record with index $i$ denotes the ballot from voter $i$, where $i = 1 \ldots n$.

$$BB \hookleftarrow \mathcal{D}_1 := auth_i(E(sn_i), E(\mathbf{b_i}), ZKP_{\text{VALID}}(\mathbf{b_i}))) : i = 1 \ldots n$$

6. The encryption of the total tally is calculated and published on the BB

$$BB \hookleftarrow E(\mathbf{Tally}) := \prod_{i=1}^{n} E(\mathbf{b_i}).$$

7. $E(\mathbf{Tally})$ is decrypted and published on the BB with its decryption proof.

$$BB \hookleftarrow E(\mathbf{Tally}), \mathbf{Tally}, ZKP_{\text{DEC}}(\mathbf{Tally}, E(\mathbf{Tally}))$$

   (In some jurisdictions, this value may be sensitive because of a small set of MERGE voters. See Full-6 for variations that allow public verifiability without publication of the separate tally.)
8. The votes and serial numbers are extracted from $\mathcal{D}_1$ and then mixed—denote this by $\mathcal{D}_2 := E(sn_i), E(\mathbf{b_i}) : i = 1 \ldots n$. The mixed pairs along with the proof of the correct mix operation are published on the BB. So, at the end of this step, we have the following data on the BB:

$$BB \hookleftarrow \mathcal{D}_3 := (E(sn_j), E(\mathbf{b_j})) : j = \pi(i), j = 1 \ldots n.$$
$$BB \hookleftarrow ZKP_{\text{MIX}}(\mathcal{D}_2, \mathcal{D}_3)$$

   where $\pi$ is the (secret) permutation applied by the Mixnet.
9. Serial numbers are decrypted and the proof of the correct decryption is published. At the end of this step, we have the following data on the BB.

$$BB \hookleftarrow \quad \mathcal{D}_4 := (K^{sn_j}, E(\mathbf{b_j}), ZKP_{\text{DEC}}(K^{sn_j}, E(sn_j))) : j = 1 \ldots n.$$

   It is also checked that there are no duplicate serial numbers. Any occurrence of duplicate serial numbers indicates significant problems with a voting computer—if there are any, the process should stop.

[5] It does not matter which is selected, as long as only one is chosen, and all $n$ selected for processing by a given jurisdiction are unambiguously marked on the BB.

### 4.3 Paper Path

At the end of the voting process, the ballot papers are sent to the Local Counting Center in individual envelopes. Each paper record is processed as follows.

10. If the jurisdiction has an existing process of scanning traditional handwritten signatures, this would apply at this point. Envelopes with invalid signatures must be set aside and handled properly.
    Each incoming envelope's sticker is scanned and its corresponding digital signature is verified. It is necessary to check that the data being signed matches the corresponding data on the BB for this voter, that the voting computer's signature is valid, and that the CAC ID is registered to vote in this jurisdiction. Then, the envelope is accepted and its corresponding record on the BB is marked as received. If any of these verifications fails, the process continues but the envelope is set aside unopened in a stack called "Rejected Envelopes" for further investigations. If the signature is valid but another validly signed envelope has already been received from the same voter, it is set aside in a stack called the "Eligibility Problem" stack.
    For all validly signed envelopes (excluding the ones inside the "Eligibility Problem" stack), the envelope contents are removed and physically shuffled, similar to standard postal voting procedures.
11. Each incoming ballot's serial number $psn$ is scanned. For each serial number, $K^{psn}$ is computed and it is checked if there is a digital ballot with a matching $K^{sn}$ on the BB.[6] Then, the location of the paper ballot is appended to the corresponding digital ballot on the BB.[7] Each digital ballot is accompanied by a parameter called TYPE where TYPE = "not matching", TYPE = "one-to-one" and TYPE = "duplicated" respectively indicate if there exist zero, one or multiple paper ballots with the matching serial number. We then have the following data on the BB:

$$BB \hookleftarrow \mathcal{D}_5 := \{(K^{sn_j}, E(\mathbf{b_j}), \{\text{POS}_{j'}\}_{j'}, \text{TYPE}_j) : \quad j = \pi(i), j' = \pi'(i), j = 1 \dots n.\}$$

    The permutation $\pi'$ represents a physical shuffle of paper ballots, and $\text{POS}_{j'}$ denotes the location of the associated paper ballots in storage. For digital ballots with TYPE = "duplicated" the positions of all matching ballots are recorded on the BB. For digital ballots with TYPE = "not matching", the parameter POS is null.
    Any paper ballot with no matching $K^{sn}$ on the BB is set aside in the "paper-only" stack. These need to be dealt with outside the cryptographic protocol.

[6] The number that naturally drops out after decryption is $K^{sn}$, not $sn$. Because we care only about exact matches, it does not matter whether we work in the exponential or plain form. We do not assume that $K^{sn}$ hides $sn$, and do not need to hide the values of $sn$, because these are not publicly associated with the voter.

[7] This is its physical location in paper ballot storage, as in a ballot manifest.

### 4.4 RLA

At the Local Counting Center the usual local Cast Vote Records (CVRs) are joined with the MERGE CVRs for auditing purposes, and the ballot manifest is updated to include both kinds of records. When a local ballot is sampled, it is retrieved and dealt with as usual. When a MERGE ballot is sampled, its discrepancy is determined according to guidelines below. Then RLA statistics are updated accordingly, in exactly the same way for remote ballots as they would be for local ones. Local officials make local decisions to accept the result or continue the audit, according to whatever RLA calculations they usually conduct. Assigning discrepancies to digital ballots is primarily determined by the stack to which the ballot belongs.

12. If the type for the selected ballot is either "one-to-one" or "duplicated", its encrypted vote $E(\mathbf{b})$ is privately decrypted to $\mathbf{b}$ and, alongside the proof of correct decryption, is supplied to all auditors and observers in the Local Counting Center, who verify the NIZKPs. They have access to the following data for the selected ballot: **Observers:** $K^{sn_j}, E(\mathbf{b_j}), \mathbf{b_j}, ZKP_{\text{DEC}}($ $, \mathbf{b_j})E(\mathbf{b_j}), \{\text{POS}_{j'}\}_{j'}$
    Then, its discrepancy is determined as follows.
    - Read the serial number *psn* of the paper ballot situated at position $\text{POS}_{j'}$. If $K^{psn} \neq K^{sn_j}$, output "serial number error" and terminate.
    - Compare the vote on the paper $\mathbf{pb}$ to $\mathbf{b_j}$ and record the discrepancy.

    For ballots of type "duplicated", the above 2-step procedure is performed for all $j'$ values and the final discrepancy is set to the *minimum* discrepancy among different $j'$ values for the digital ballot. That is, in case of several paper ballots, one of the ballots with lowest discrepancy value is selected.
    Any occurrences of the "serial number error" indicates a significant problem with the scanning device and hence necessitates restarting the entire process from Step 11 onward using another scanning device.
13. For ballots with TYPE = "not matching", the encrypted ballot is privately decrypted and, alongside the proof of correct decryption, is supplied to all auditors and observers. Then, the discrepancy is determined by setting it to the maximum possible value for such a ballot, or in other words, employing a worst-case paper assumption for the given ballot.

Verification steps are detailed in Full-B.3, with error handling in Full-B.4.

## 5 Security Analysis

*Verifiability.* We begin by outlining the audit process in an ideal scenario featuring trustworthy paper ballots and a one-to-one correspondence between each paper ballot and its electronic record. A subset of voters in the ideal world is controlled by the adversary, but other voters follow the voting instructions.

Next, we progressively transform this ideal world into our protocol (i.e., the Real World) through a series of intermediate worlds. At each stage, we establish the validity of the following statement: *any combination of digital and paper ballots in the newly defined world corresponds to a set of digital and paper ballots in the present world. This correspondence ensures identical (1) reported totals, (2) number of voters, and (3) honest voters. Furthermore,*

- *The digital ballots in the newly defined world are a permutation of those in the present world, or*
- *For any ballot selected for the RLA in the present world, the discrepancy value is equal to or smaller than the discrepancy value of its corresponding ballot in the newly defined world.*

These statements, based on Remark 1, imply that for any given risk limit, when the announced outcome is wrong, if the RLA functionality with any given sample set accepts the announced outcome in a newly defined world, it will accept the corresponding outcome in the present world. The intuition is that inconsistencies between CVRs and paper records are handled in such a way that the overall risk value does not decrease from one scenario to the next. Since the last defined world is the same as the real MERGE protocol, the proof shows that it has no greater likelihood of accepting a wrong result than an 'ordinary' RLA conducted in the ideal world on the trustworthy paper ballots with a one-to-one correspondence between each paper ballot and its CVR. A precise statement and proof are contained in Full-5.1 and C.1.

*Privacy.* MERGE does not provide ballot privacy, because the adversary in this model might drop envelopes and thus use differencing attacks to infer individual votes. For example, if the adversary drops one ballot, labelled on the sticker as Alice's, then the adversary knows whose vote is the unique one with no match. Since there is a non-zero probability that this is audited and opened, Alice's privacy cannot be guaranteed. More generally, since the RLA process may lead to a full manual recount of the arrived paper ballots, differencing attacks are always possible. Full-5.2 and C.2 examine the protocol's privacy, and prove that when the attacker cannot drop ballots the protocol satisfies the definition of privacy given in [5]. Full-5.3 contains a discussion of receipt freeness.

## 6 Conclusion and Future Work

Usable voter-verification—without undermining coercion resistance—remains the major open problem in this field. Using paper requires transmitting it back to the counting location; using cryptography demands that the voter successfully performs one of the human-computer-interaction protocols which, despite ingenious recent advances, remain very hard for voters to run successfully. Our protocol combines plain paper verification with cryptographic assurance to get the best properties from each.

**Acknowledgements.** Thanks to Josh Benaloh, Thomas Haines, Michael Naehrig, Olivier Pereira, John Sebes, Dan Wallach and Trail of Bits for helpful comments and review. The UI team at Rice and the development team at VotingWorks greatly improved the practicality and usability of the system. Thanks also to Andrew Appel and Philip Stark for their critical review of an earlier version of this work, which led to some significant improvements.

## References

1. Appel, A.W., DeMillo, R.A., Stark, P.B.: Ballot-marking devices cannot ensure the will of the voters. Election Law J. Rules Politics Policy **19**(3), 432–450 (2020)
2. Banuelos, J.H., Stark, P.B.: Limiting risk by turning manifest phantoms into evil zombies (2012). https://arxiv.org/abs/1207.3413
3. Benaloh, J., Naehrig, M.: Electionguard specification v2.0 (2023)
4. Benaloh, J., Naehrig, M., Pereira, O., Wallach, D.S.: ElectionGuard: a cryptographic toolkit to enable verifiable elections 2024. Cryptology ePrint Archive, Paper 2024/955 (2024)
5. Bernhard, D., Cortier, V., Galindo, D., Pereira, O., Warinschi, B.: SOK: a comprehensive analysis of game-based ballot privacy definitions. In: 2015 IEEE S&P, pp. 499–516. IEEE (2015)
6. Devillez, H., Pereira, O., Peters, T.: Verifiable and private vote-by-mail. Cryptology ePrint Archive (2024)
7. Fuller, B., Harrison, A., Russell, A.: Adaptive risk-limiting comparison audits. In: 2023 IEEE S & P, pp. 3314–3331. IEEE (2023)
8. Kortum, P., Byrne, M.D., Whitmore, J.: Voter verification of ballot marking device ballots is a two-part question: can they? mostly, they can. do they? Mostly, they don't. Election Law J. Rules, Politics Policy **20**(3), 243–253 (2021)
9. Rivest, R.L.: On the notion of 'software independence' in voting systems. Philos. Trans. Royal Soc. A: Math. Phys. Eng. Sci. **366**(1881), 3759–3767 (2008). https://people.csail.mit.edu/rivest/pubs/RW06.pdf
10. Stark, P.B.: Sets of half-average nulls generate risk-limiting audits: SHANGRLA. In: International Conference on Financial Cryptography and Data Security, pp. 319–336. Springer (2020). https://doi.org/10.1007/978-3-030-54455-3_23, https://arxiv.org/abs/1911.10035



# REACTIVE: Rethinking Effective Approaches Concerning Trustees in Verifiable Elections

Josh Benaloh[1], Michael Naehrig[1], and Olivier Pereira[1,2(✉)]

[1] Microsoft Research, Redmond, WA, USA
olivier.pereira@uclouvain.be
[2] UCLouvain, 1348 Louvain-la-Neuve, Belgium

**Abstract.** For more than forty years, two principal questions have been asked when designing verifiable election systems: how will the integrity of the results be demonstrated and how will the privacy of votes be preserved? Many approaches have been taken towards answering the first question such as use of mixnets and homomorphic tallying. But, in the case of large-scale elections, the second question has always been answered in the same way: decryption capabilities are divided amongst multiple independent "trustees" so that a collusion is required to compromise privacy.

In practice, however, this approach can be fairly challenging to deploy. Even if multiple human trustees are chosen, they typically use software and often also hardware provided by a single voting system supplier, and they rarely have any real opportunity to confirm its correctness. As a result, we observe that trustees are generally not in a position to exercise the independent judgment necessary to ensure privacy.

This Systematization of Knowledge (SoK) paper looks at several aspects of the trustee experience. It begins by surveying and discussing various cryptographic protocols that have been used for key generation in elections, explores their impact on the role of trustees, and notes that even the theory of proper use of trustees is more challenging than it might seem. This is illustrated by showing that one of the only references defining a full threshold distributed key generation (DKG) for elections defines an insecure protocol. Belenios, a broadly used open-source voting system, claims to rely on that reference for its DKG and security proof. Fortunately, it does not inherit the same vulnerability, and we offer a security proof for the Belenios DKG.

The paper then discusses various practical contexts, in terms of humans, software, and hardware, and their impact on the practical deployment of a trustee-based privacy model.

**Keywords:** Verifiable elections · trustees · distributed key generation · secure hardware


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 17–37, 2026.
https://doi.org/10.1007/978-3-032-05036-6_2

## 1 Introduction

The academic approach to preserving privacy in large scale verifiable election protocols is to rely on independent *trustees* who are each responsible for production, maintenance, use, and ultimately destruction of their own cryptographic keys. Examples, in chronological order, include [2,5,8,11,12,16,17,32,37].

Decades of literature, beginning with Benaloh and Yung [8], offer creative cryptographic protocols with excellent properties to achieve precisely this kind of separation during key generation [15,22,27,31]. Such distributed key generation mechanisms are key to enable threshold encryption [19], a well-established technique that provides robustness by allowing a pre-determined number of keyholders to perform the necessary decryptions to complete an election while preventing any set of insufficient size from gaining any information at all.

These approaches are very effective at addressing collusion amongst keyholders and are used both in systems which employ homomorphic tallying and those using modern re-encryption mixnets. Parameters can be set to prevent collusions smaller than a defined size from compromising voter privacy. But this relies upon the assumption that the trustees in an election are technical experts with full agency over their decisions and environment.

In practice, the trustees follow specified protocols for distributed key generation before the election, and must validate that the key they generated is actually used in the election. When encrypted tallies need to be decrypted after the voting period, they must validate a set of ciphertexts to be decrypted and follow protocols for threshold decryption. And, during the whole process, the trustees need to make sure that they remain in control of their secret key material, in order to prevent any abuse. This high level blueprint is common to virtually every verifiable voting system that has been deployed at scale, including Helios [2], Scantegrity II [12], vVote [11], the Verificatum mixnet [37] (used in Estonia for instance), or the Swiss Internet voting system [32].

Human trustees are aided by trustee hardware, computing devices that run software for carrying out the necessary cryptographic operations and securely communicate with the other trustee devices. These protocols are often facilitated by a central authority such as an election administrator that routes the trustees inputs and outputs.

However, ideally, to guarantee independence, trustees should be able to independently evaluate and possibly choose the hardware and software they use: corrupted hardware and software might display everything that a human trustee expects to see while leaking every secret to interested parties. Trusted computing techniques may offer some help but, if they are used to assist trustees, they only help as far as trustees are able to verify that the trusted execution technology is actually in use and is used with the expected code, which may be a challenge of its own.

We observe in this paper that the reality is often quite different. In practice, some or all trustees are often chosen from local communities as representatives of the public, and the key management happens during well orchestrated public ceremonies, using software and hardware provided by authorities. In such a con-

text, trustees do not have the opportunity or the tools to perform any thorough verification steps—even when they happen to have the technical expertise to fulfill such tasks. They are instead required to press the buttons they are told to press on the devices provided to them, which contain pre-loaded software. As a result, there is no real independence and no substantive protection from curious election administrators who want to view election data that is supposed to remain confidential.

Even when trustees are chosen for their expertise, they are usually required to utilize software and hardware provided to them by an external provider who they may not trust. Furthermore, selecting trustees only from amongst "elites" may restrict their independence and create suspicion from the public.

As a result, the ideal situation of trustees selected as well-trained representatives of the public who utilize hardware and software that they have the means to evaluate as trustworthy seems quite distant from the current practices, and seems to be unlikely to become the rule rather than the exception in the foreseeable future.

Our focus here is on the role of trustees in protecting voter privacy and, in particular, how trustees generate the election keys—similar issues arise around decryption and mixing operations. There are some election systems in which integrity also depends upon trustees (see, for instance, [14,32]). However, the essence of this work is an exploration of how well trustees serve to protect privacy in practice.

While the integrity of the results can potentially be verified over and over by independent parties without access to privileged information and at any chosen time, privacy currently depends on a closed set of trustees being able to complete their tasks diligently. Therefore, without this true independent trustee expertise, and hardware and software independent evaluation, there is very limited basis for confidence in the privacy of the votes, and there is essentially no option to verify that privacy holds.

This paper starts by reviewing protocols for trustees, focusing on distributed key generation (DKG) protocols used in real-world verifiable elections. We discuss the properties of these protocols and the practical demands that they place on the trustees. On our way, we explore technical difficulties in some DKGs and propose a fresh security analysis of the DKG protocol of the Belenios Internet voting system.

We then turn to a discussion of the potential benefits and trade-offs that the reliance on secure hardware can bring. In particular, attestation mechanisms can shift the trust that currently needs to be placed in humans, with limited assessment and verification options, to verifiable attestations produced by hardware that can be independently reviewed and challenged by arbitrary auditors rather than by selected trustees. In such a setting, privacy may become a universally verifiable property rather than a property that depends on a specific set of trustees.

## 2 Key Generation Protocols

In order to understand the tasks that trustees need to accomplish, we start by describing the most common cryptographic protocols that are used in existing verifiable voting systems.

We focus on distributed key generation (DKG) protocols as a central ingredient of the kind of tasks that trustees are expected to perform. There is of course more in the role of trustees than running a DKG, but the cryptographic operations associated with decryption and mixing bring requirements that are similar to those for key generation, and are described elsewhere, so we do not systematically elaborate on them here. We will discuss post-election operations below.

### 2.1 Single Key

ElGamal encryption [20] is by far the most common algorithm for encrypting votes in verifiable voting systems: it is for instance used in all voting systems mentioned in the introduction above [2,11,12,32,37]. In the simplest form of key generation, the key pair used to encrypt ballots can be generated by one single entity. This approach has been used in Helios 1.0 [1] and, for a long time, in the Estonian Internet voting system, in which all ballots are decrypted using a single key held in an HSM [14]. This solution relies entirely on the security of the HSM device: if the key is somehow extracted from the device, or if someone manages to make that device decrypt non-anonymized encrypted votes, then ballots are not secret anymore. If the HSM device fails and (access to) the decryption key is lost, the election tally cannot be computed. This second concern can be mitigated easily with secret key backups. However, the existence of such backups creates new opportunities for stealing the secret key, increasing the risk of violating vote secrecy.

From a technical point of view, and in the context of ElGamal encryption, the generation process of a single key is extremely simple. A group $\mathbb{G}$ of prime order $q$ is chosen (typically as a public parameter of the election system), together with a generator $g$ of that group. The key generation consists in selecting a secret key $s \leftarrow \mathbb{Z}_q$ uniformly at random, and publishing $K = g^s$ as the encryption public key.

### 2.2 One-Round DKG

To limit the risks of having the single decryption key stolen and potentially used to break the secrecy of all the votes, various voting systems turn to a very simple extension of the above key generation mechanism: $n$ trustees $T_1, \ldots, T_n$ independently run the single key generation and publish their public keys. These are then (publicly) combined into a single encryption public key. Here, all $n$ secret keys are needed for decryption.

Concretely, each $T_i$ selects a secret key $s_i \leftarrow \mathbb{Z}_q$ as before and publishes the corresponding $K_i = g^{s_i}$ together with a Schnorr proof of knowledge of $s_i$ in

order to prevent so-called rogue key attacks [28]. The encryption public key is computed as $K = \prod_{i=1}^{n} K_i$, with a corresponding secret key $s = \sum_{i=1}^{n} s_i$ that is never explicitly computed.

This approach is used in various systems, including Helios [2] (since version 2.0), Belenios [16], and the Swiss Internet voting system [32]. Its use in the context of elections was analyzed by Bernhard et al. [9].

This protocol offers two important benefits. First, it addresses the privacy risk when relying only on a single decryption key: now, in order to break privacy, all $n$ private keys are needed. Second, it keeps most of the simplicity of the single-key protocol. In particular, key generation can still be implemented in a dozen lines of Python code, which can be reviewed easily by a knowledgable trustee, even during a key generation ceremony.

On the flip side, this protocol is even more fragile than the single-key protocol: losing any one of the $n$ secret keys is sufficient to prevent tallying the election. Again, this can be addressed with backups, which may be less sensitive than in the single-key case since a copy of every secret key is needed to perform decryption operations. Concretely, various solutions have been used: trustees can be paired, and each pair of trustees generates a single secret key of which two copies are kept [10]. In some cases, more elegant solutions can be proposed: for instance, in a setting with 3 trustees sitting around a table, each trustee can give a copy of its secret key to the trustee sitting to its left. This, in effect, offers a two-out-of-three threshold key generation process.

### 2.3 Threshold DKG

A more general approach to handle the risks of some trustees (or secret keys) being unavailable, is to rely on a threshold distributed key generation protocol, in which only $k$ out of $n$ trustees are needed to perform a decryption operation. As far as we know, the protocols that have been deployed for elections all follow a structure proposed by Pedersen: the $n$ trustees start by generating their key pairs as in the one-round DKG protocol outlined above, then run a verifiable secret sharing (VSS) protocol to share their secret keys $s_i$ with the $n - 1$ other trustees, using a threshold of $k$ [31] (each trustee keeps one share of its own key, so this is $k$-out-of-$n$ secret sharing). The trustees can combine the shares they received into shares of the secret key $s$ that is the sum of the $s_i$. The encryption public key $K$ is computed exactly as in the one-round DKG above.

Variations of this protocol are used in various election protocols and have been implemented in software packages, including Verificatum [37], which relies on a protocol proposed by Gennaro et al. [22], Belenios [16], which relies on a protocol by Cortier et al. [15] and ElectionGuard [6], whose DKG protocol is analyzed by Benaloh et al. [7].

The obvious benefit of these protocols is their flexibility for choosing the level of robustness: the original protocol by Pedersen and the one by Gennaro et al. make it possible to choose any $k < n/2$, while the two other protocols aim at supporting any choice of $k \leq n$.

However, they are less convenient. First, they require exchanging encrypted shares between trustees, who are required to all be present at the same time, or to be present multiple times in order to complete the multiple rounds of the protocol. Second, the secret and authentic exchange of shares requires the exchange and authentication of keys for direct communication between the trustees, which may not be trivial to perform: PKIs are usually not structurally available and may need to be built as part of the DKG. Eventually, the extra protocol complexity also results in extra code complexity: even an expert may be uncomfortable reviewing the code of a full threshold DKG during a key generation ceremony (of course, the review may happen in advance and code hashes can be compared, but this again brings additional complexity).

The additional complexity of a threshold DKG may also impact the election verifier: a verifier for Verificatum and Belenios needs to compute Lagrange coefficients and adjust for the set of trustees present for decryption, which is much less straightforward than in the one-round DKG protocol. ElectionGuard, though, administratively combines all the decryption proofs so that the verification of a decryption operation is as simple as in the single-key case. The focus on simplifying the process of election verification is explicit there.

Eventually, the additional complexity in the key generation also leads to a more complicated security analysis, which we now illustrate in the context of the Belenios DKG.

### 2.4 The Belenios DKG

Belenios offers two DKG protocols: one that is essentially the single-round protocol above, and a threshold protocol that, on page 2 of the Belenios specification (v. 2.5) [23], is claimed to be "described in [Cortier et al.] [15] and proved in [the works of Cortier et al. and Bernhard et al.] [9,15]".

As mentioned before, the threshold DKG protocol follows the Pedersen design, which we outline here, given a set of trustees $T_1, \ldots, T_n$, a chosen threshold $k$, and assuming the availability of a public bulletin board that behaves as a broadcast channel.

1. Each trustee $T_i$ chooses a random polynomial $P_i(x) = a_{i,0} + a_{i,1}x + \cdots + a_{i,k-1}x^{k-1}$ of degree $k-1$ with coefficients in $\mathbb{Z}_q$ and posts $K_{i,j} = g^{a_{i,j}}$ for $j \in [0, k-1]$ on the bulletin board.
2. Each trustee $T_i$ sends $s_{i,j} = P_i(j)$ to every other trustee $T_j$ on a secret and authentic channel.
3. Each trustee $T_j$ verifies the shares it received by checking that $g^{s_{i,j}} = \prod_{\ell=0}^{k-1} (K_{i,\ell})^{(j^\ell)}$ for every value of $i$. If any check fails, a recovery phase starts, which we do not discuss here.
4. The public key is defined as $K = \prod_{i=1}^{n} K_{i,0}$ and each trustee $T_i$ computes its decryption key share $z_i = \sum_{j=1}^{n} s_{j,i}$.

Pedersen proposed this protocol for an honest majority, that is, $n \geq 2k-1$, and this is the model adopted by Verificatum [37]. However, Cortier et al. offer a

proof that, when the protocol is used in combination with ElGamal encryption, it offers IND-CPA security under the DDH assumption for arbitrary threshold choices, that is, for any $k \leq n$ [15]. (The other reference [9] mentioned in the Belenios specification [23] focuses on the one-round non-threshold protocol.)

**Pedersen's DKG is Insecure in the Case of a Dishonest Majority.** As discussed above, a central aspect of the security of a DKG is to make sure that no subset of less than $k$ trustees can "force" the public key to take a value of their choice, of which they would know the corresponding discrete logarithm. For the single-round protocol (and the ElectionGuard protocol), this is achieved by forcing every trustee to offer a Schnorr proof that it knows the $s_i$ value corresponding to its public key $K_i$. For the Pedersen protocol, this is achieved by having at least $k$ trustees verify the correctness of the shares they received at Step 3 of the protocol: if $k$ trustees hold correct shares of $a_{i,0}$, then it must be the case that $T_i$ knows $a_{i,0}$. The assumption of an honest majority guarantees that at least $k$ trustees will perform the expected verification steps. However, when there is a dishonest majority, this is not the case anymore (this is overlooked in [15]), and a dishonest trustee might be able to simulate the VSS steps for the honest trustees, while ignoring its secret key share $s_i$, opening the way for arbitrarily choosing the final public key $K$.

Similar attacks have been known for a long time (see Langford [28] for instance). Common mitigations include an initial round during which every trustee commits to its $K_{i,j}$ values before opening them and resuming the protocol, or requiring every trustee to provide a Schnorr proof that it knows the discrete logarithms of its $K_{i,j}$ values w.r.t. $g$. The second option is the one adopted in ElectionGuard [6].

**The DKG in Belenios.** Fortunately, Belenios does *not* exactly follow Cortier et al. [15] but does something slightly different: it requires each trustee $T_j$ to offer a proof of knowledge of $z_j$ defined as the logarithm in base $g$ of $g^{z_j} = \prod_{i=1}^{n} g^{s_{i,j}} = \prod_{i=1}^{n} \prod_{\ell=0}^{k-1} (K_{i,\ell})^{j^\ell}$. To the best of our knowledge, this option is original and has never been publicly analyzed.

We take the opportunity here to offer the first proof of security of the Belenios DKG protocol, which is now the basis of the DKG discussed in version 3.0 of the Belenios specification [24]. We use the name *ElGamal encryption* to designate the traditional ElGamal encryption scheme with single-party key generation, and use *Belenios ElGamal encryption* for the threshold encryption scheme that uses the Pedersen DKG augmented with a Schnorr proof of knowledge of $z_j$ provided by each trustee as its key generation protocol and encrypts with ElGamal encryption. Following other works in the DKG literature [22], we assume here that secret and authentic communication channels for communicating the shares are available between all pairs of trustees, and that trustees verify that they have a common view of the protocol's public elements.

**Theorem 1.** *Belenios ElGamal encryption is IND-CPA secure under static corruption of up to $k-1$ trustees ($1 \leq k \leq n$) if ElGamal encryption (with standard single-party key generation) is IND-CPA secure with the same public parameters.*

In other words, any PPT adversary that can break the IND-CPA security of Belenios ElGamal encryption while controlling up to $k-1$ trustees can be efficiently transformed into an IND-CPA adversary against ElGamal encryption.

We observe that this statement makes no assumption regarding the Schnorr protocol that is part of Belenios ElGamal: indeed, this one can be proven to be secure (in the random oracle model) in the presence of a computationally bounded adversary, independently of any computational assumption.

*Proof.* Let $k$ and $n$ be fixed and define $C = \{1, \ldots, k-1\}$ and $H = \{k, \ldots, n\}$. Assume, w.l.o.g., that the trustees in the set $\{T_i\}_{i \in C}$ are corrupted, while the others are honest. We design an adversary $B$ against ElGamal encryption that wins the IND-CPA game with a probability equal, up to a negligible difference, to the probability that an adversary $A$ controlling the corrupted trustees breaks the IND-CPA security of Belenios ElGamal encryption.

Given $A$, we design $B$ as follows: when $B$ receives the ElGamal public parameters $(\mathbb{G}, q, g)$ and the public key $K^*$ from the standard ElGamal challenger, it forwards the public parameters to $A$. $B$ honestly plays the role of the honest trustees, except for $T_n$. For emulating $T_n$, it simulates the sharing of the discrete logarithm of $K_{n,0} = K^*$ by picking $s_{n,i}$ as a random element of $\mathbb{Z}_q$ for every $i \in C$. It then derives $\{K_{n,i}\}_{i \in C}$ so that $g^{s_{n,i}} = \prod_{j=0}^{k-1} (K_{n,j})^{i^j}$ for every $i \in C$, by Lagrange interpolation "in the exponent"—this works because $|C| < k$. The view of $A$ is distributed in a way that is identical to what it would be in a normal execution of the Belenios ElGamal key generation, and $A$ completes the protocol on behalf of the corrupted trustees. When $B$ received all its shares from the corrupted trustees, it can program the random oracle in order to produce simulated Schnorr proofs of knowledge of $z_i$ as the logarithm of $\prod_{j=1}^{n} g^{s_{j,i}} = \prod_{j=1}^{n} \prod_{\ell=0}^{k-1} (K_{j,\ell})^{i^\ell}$ in base $g$ for every $i \in H$. $B$ checks the resulting transcripts and fails if anything is wrong.

If all the verification steps succeed, $B$ can now extract the $z_i = \sum_{j=1}^{n} s_{j,i}$ values for $i \in C$ from the Schnorr proofs provided by $A$. Extraction from the multiple proofs can be performed because all the proofs are available at the same time. Subtracting the shares that it sent on behalf of the honest trustees, $B$ can also compute $z_{C,i} = \sum_{j \in C} s_{j,i}$ for $i \in C$. When $i \in H$, the values $z_{C,i} = \sum_{j \in C} s_{j,i}$ can also be computed, since the $s_{j,i}$ values are shares that have been sent by $A$. As a result, $B$ has all $n$ shares of $s_C = \sum_{j \in C} s_{j,0}$, and can reconstruct that value.

Eventually, when $A$ asks for the encryption of a pair of messages $(m_0, m_1)$, $B$ forwards it to the ElGamal challenger, who returns a ciphertext $(c_0, c_1) = (g^r, m_b (K^*)^r)$. $B$ then submits to $A$ the ciphertext $(c_0, c_1 (c_0)^{s_C + \sum_{i=k}^{n-1} s_i}) = (g^r, m_b K^r)$. When $A$ outputs a guess $b'$ on $b$, $B$ forwards that guess to the ElGamal challenger. The probability that $b = b'$ is exactly the one that $A$ makes a correct guess in the Belenios ElGamal IND-CPA security game. The only

possible discrepancy comes from the potential failures to simulate or extract a Schnorr proof, which can be made negligible. □

We summarize the properties of the key generation processes used in the systems we discussed in Table 1.

**Table 1.** Overview of key generation in various deployed verifiable voting systems.

| | Single trustee | $n$-out-of-$n$ DKG | $k$-out-of-$n$ DKG (honest maj.) | $k$-out-of-$n$ DKG (dishonest maj.) |
|---|---|---|---|---|
| Helios 1.0 [1] | ✓ | | | |
| Estonia [14,38] | ✓ | | | |
| Helios 2.0 [2] | | ✓ | | |
| Belenios [16] | | ✓ | | ✓ |
| Swiss e-Voting [32] | | ✓ | | |
| Verificatum [37] | | | ✓ | |
| ElectionGuard [6] | | | | ✓ |

## 3 Protocol Setups

The above protocols may be cryptographically correct, but could be useless if their setup assumptions are not satisfied when they are deployed, or if a malicious election administrator gets the possibility to completely circumvent these protocols. For instance, after completing a session of a DKG protocol, trustees need a mechanism to verify that the key used in the election is really the one they generated and has not been replaced with a key that is fully controlled by another entity.

Elections, whether they are electronic or not, cannot exist in isolation: voters need a way to access authentic blank ballots, they need to know where and when to cast their votes, and where to read the election results. This is often achieved by relying on some form of a public bulletin board, and we will assume that such a public bulletin board is available—how to build bulletin boards has been largely discussed in other places [18,26].

The single-key and one-round DKG protocols can directly be implemented when a bulletin board is available: trustees must verify that the public key they produced has been posted on the bulletin board, and that it has not been replaced by the bulletin board manager for instance. Election verifiers can check that all ballots are encrypted and then tallied w.r.t. the correct public keys.

The threshold DKG protocol, however, additionally requires secret and authentic communication channels between trustees to exchange key shares. The assumption that such channels exist is standard and appears virtually everywhere in the DKG literature [22,31].

In many cases, such channels are easy to implement using encryption and signature mechanisms (e.g., via TLS) assuming a trusted public-key infrastructure (PKI). However, in the context of elections, it is much more challenging to depend on a setting in which a trusted PKI exists and certified keys are in the hands of trustees. Trustees will rarely have a certified signature key and, depending on the context, the distributed trust provided by a threshold DKG might be undermined by relying on certificates signed by a single centralized authority, which in turn may be external to the election process and be driven by different incentives.

In practice, this is addressed in various ways. We describe a few examples from systems that use a threshold DKG. These explorations show that this secure point-to-point communication layer, which is just assumed to be available in the DKG literature, is actually far from being obvious to organize in practice.

1. In Verificatum, each trustee generates signing keys and gives the corresponding verification keys to the other trustees. The Verificatum manual suggests on page 2 that, "in practice, the operators could organize a physical meeting to which they bring their laptops and execute the above steps" and "for convenience, hexadecimal encoded hash digests of files can be computed using *vmni* to allow all parties to check that they hold identical protocol info files at the end" [35].
2. The Belenios specification (v.2.5) indicates that each trustee generates signing and encryption keys, which are shared with the other trustees via the voting server. At the opening of the election, each trustee "checks that [its own certificate] appears in the set of verification keys $PK$ of the election" [23].
3. In ElectionGuard, trustees do not generate signing keys. Instead, the ElectionGuard specification (v.2.1) indicates on page 27 that, at the conclusion of the DKG, a preliminary record that contains all public information from the DKG execution is published on the bulletin board. It includes all encryption keys used to exchange shares, the $K_{i,j}$ values, and matching Schnorr proofs. Trustees must hash this information and compare it to a hash computed from their own view of the DKG execution [6].

Verificatum and Belenios create authentic channels using signing keys. Verificatum suggests direct contact between the trustees, allowing direct comparison and confirmation of all the keys that they are going to use. Of course, it remains important for the trustees to verify that the correct key material is also published as part of the official election record on the bulletin board.

In a different approach, Belenios focuses on the direct verification of the election record and requires that trustees confirm the presence of their own signing (and encryption) keys. We pointed out that trustees should agree on all the keys that have been used to ensure that key shares are not compromised—this verification step was absent from Version 2.5 of the specification and, following our comment, a more thorough verification mechanism has been introduced since Version 2.5.1 in order to avoid potential attacks.

ElectionGuard adopts a completely different approach and authenticates the view of the protocol execution rather than using signing keys to sign that view.

This has essentially the same effect when verification is successful. Of course, signing keys may additionally provide a non-repudiation property, which could be used for accountability should problems occur. However, when signing key generation is part of the protocol, a malicious trustee might as well complain that the signature verification key published on its behalf is incorrect. Here, the in-person protocol suggested in Verificatum may be advantageous when trustees are required to agree on their keys in person, that is, in a setting where authenticity cannot be questioned. It is also the most demanding option for human trustees.

## 4 Trustees, Their Hardware, and Their Software

To the best of our knowledge, the protocols described in Sect. 2 guarantee the expected cryptographic security properties. However, as we have seen in Sect. 3, the associated setup assumption and the context of usage make it so that these security properties cannot be granted by machines *only*: when relying on a designated set of human trustees, these trustees must confirm authenticity of the key material used in the election.

The challenges associated with trustees become particularly pronounced when organizing an election. We elaborate on these issues in this section, again by reviewing different approaches adopted in practice. As before, we focus on the key generation deployment, mainly on the choice of trustees, software, and hardware. The level of detail that we can provide varies a lot, as the available public information is often scarce.

### 4.1 Key Generation Ceremonies in Practice

**Estonia.** At least in the early versions of their Internet voting system (pre-IVXV), Estonia used an HSM to generate a single key and decrypt all ballots after a threshold number of election officials inserted their "cryptosticks" [14].

We do not know what is offered for verification here, but we assume that the HSM can produce an attestation that it generated the election key and that the key was never released, as well as a list of all the ciphertexts that were decrypted using that key over its life time.

The task of trustees is highly limited here: vote secrecy essentially depends on proper anonymization of the encrypted votes before they are decrypted (which may not be an obvious operation in the absence of the verifiable mixnet that was introduced in a later version of the system), and on verifying that the HSM has not been abused to perform unauthorized decryption or key export operations.

**UCLouvain.** Since version 2.0, Helios uses the one-round distributed key generation described above. To the best of our knowledge, Helios has not been deployed in government elections, but we have a description of at least one deployment for a university election at UCLouvain [2].

In that election, trustees were selected from various voter groups and worked with the help of external experts. An in-person meeting was organized for the key

generation. Trustees were provided with laptops from which hard-disk drives and network interfaces had been removed. The laptops were booted on Linux using live-CDs, and minimal Python code was provided to generate the election keys, under control of the external experts. Public and secret keys were generated and saved on multiple USB sticks (including copies of the secret keys as backup). The laptops were then turned off and the CDs destroyed. Copies of the public keys were uploaded in the system, and trustees were asked to compare the public keys published on their behalf with their own copies.

So, efforts were made to restrict the ways in which secret keys could be exfiltrated from the hardware, but such measures obviously require expert control.

The software used by the trustees apparently was provided by the same source, but it seems that this code was simple enough to be reviewed during the key generation process—again, this requires expert control. Throughout, trustees remained in control of their secret keys. The potential loss of any one of these keys could have prevented the election organizers from tallying the election—hence the existence of backup copies.

We can see that there are tensions between different incentives here. On the one hand, the cryptographic protocol is designed to put the trustees in control of their keys. On the other hand, should a trustee be missing or lose a key, the tallying process might fail. In such circumstances, the blame is likely to be put more strongly on the election organizer and technology supplier than on the citizen who volunteered to help as a trustee. This actually places a strong incentive on the organizer and technology provider to "tweak" key generation in a way that allows them to obtain copies of all the keys, not to break the privacy of the votes, but to recover from a human failure by a trustee that would badly reflect on them. If the experts are chosen by the organizers, this may undermine the trust that can be granted in the key generation process.

**Switzerland.** The hardware and roles for key management in the Swiss Post voting system used for Swiss government elections is publicly available [33].

Its DKG is a variation of the one-pass protocol above, with some keys generated by Swiss Post and other keys generated by the cantons organizing elections.

Swiss Post holds four keys, generated on four control components, running four different hardened operating systems in order to mitigate the risks of OS-level exploits (Debian, RedHat, Ubuntu, and Windows are listed). The cantons use multiple laptops deployed in a secure offline environment: they receive the election data through USB sticks.

At the Swiss Post level, the operations are under control of multiple expert teams, even though all of them seem to work under the authority of Swiss Post. The security operations and human resources at the canton level are less documented, and we can guess that they vary among cantons.

At least at Swiss Post, the trustee tasks are performed by experts who would be able to verify the software that they are running, but this comes at the cost of limited independence. The generation of other keys at the Canton level may offer an important level of human independence though. However, the level of

independence regarding the software and hardware is not clear: if the voting software, OS, and hardware are provided by Swiss Post, then the effective independence may be reduced. Again, we do not know how this is handled at the canton level.

**Franklin County, Idaho.** The hardware and key management in a deployment of the ElectionGuard SDK during the 2022 General Election in Franklin County, Idaho is documented [29]. This case is interesting in its own right because ElectionGuard uses a threshold DGK protocol, i.e., trustee devices had to communicate with each other during key generation.

Trustee devices were connected to an administrator device on a local network. Trustees were ordinary citizens who operated devices that they were given, running pre-installed software. At the end of key generation, the trustee devices storing their private keys were kept in safes controlled by the election administrator. The devices themselves required the trustee fingerprint to be activated. However, in order to recover from device failures (and despite the use of a threshold DKG), keys were also exported on thumb drives and stored in the same safe.

The real impact the trustees had here seems minimal. Trustees had no control of the software and hardware, and even the keys were kept in the custody of the election administrator. However, one should not expect that the expertise that can be deployed to run a national voting system in Switzerland can also be deployed in much smaller elections such as those in Franklin County: the election records show that 113 ballots were cast and 2 were spoiled in that election. In that specific deployment, no linkage between the individual paper ballots and the voters who cast them was maintained; so encrypted ballots have a much weaker link to the voter identity than in the Internet voting systems described above, in which a voting server controls the identity of the voter and receives its encrypted ballot. So, even though the security of the keys seems to have been lower in this case, the reliance on the keys for secret ballots seems lower as well.

**College Park, Maryland.** ElectionGuard was also used in College Park, Maryland in 2023—one year after Franklin County, Idaho. The set up and process were very similar—with trustees (termed as "guardians") selected from the public without special expertise by the city Board of Elections.

A public information session was held on September 27, 2023[1] and the tally decryption ceremony was held immediately after the close of the election on November 5, 2025.[2] The tally decryption ceremony shows how little independent agency the trustees had in the process and how ineffective this approach was at protecting voter privacy from a potentially malicious election administrator.

[1] Available at https://college-park.granicus.com/MediaPlayer.php?view_id=2&clip_id=1405.

[2] Available at https://college-park.granicus.com/MediaPlayer.php?view_id=2&clip_id=1424 (see, especially, the five-minutes commencing from the ten-minute mark).

**Summing Up.** It is clear that, in all these cases, keys that safeguard vote privacy are not in the sole control of trustees that are representatives of the general population concerned about the election.

In all cases, the systems rely on authorities to provide the trustees with experts and trustworthy hardware and software. Even though the entire purpose of splitting keys amongst independent entities is to prevent a curious central authority from simply reading votes, in practice, there is little to stop a curious authority from providing software that reveals keys or performs additional decryptions. And, even if the software is correct, it is unlikely that anyone will notice if a curious authority simply asks for additional decryptions.

Trustees charged with generating and managing keys should be well-trained and thoroughly understand their role. They should bring software and devices obtained from sources they choose to trust. They should inspect the data they are asked to decrypt and ensure that it contains only the values necessary and appropriate to complete the election process. While such assumptions are commonplace and may seem reasonable in theory, practical deployments show that the reality makes this challenging.

## 5 The Hardware Alternative

The publication of audit data can guarantee the integrity of election results without any requirement to trust designated parties (trustees, election administration, etc.). However, in the solutions described above, even if deployed perfectly, privacy requires trusting that a subset of a fixed group of trustees are behaving correctly. This behavior includes correct human behavior (humans do not distribute copies or misuse their keys) but also correct behavior of software and hardware (the software and the hardware do not leak or misuse secret key material).

Hardware-based security technologies have existed for a long time and have advanced tremendously during the last few years. The capabilities of hardware security modules (HSMs) dramatically expanded, and trusted execution environments (TEEs, including Intel SGX and AMD SEV-SNP) are now readily available. These technologies offer the opportunity to replace vague and difficult to verify assumptions on human processes with clear, specific assumptions on hardware.

HSMs and TEEs can generate keys, log when and how they are used, delete keys when they are not needed anymore, and publish signed attestations of all of these steps. Independent observers can verify the attestations made by these devices and the certificate chain to the device manufacturers. Of course, it is possible that a device does not perform as advertised [4,34] or that the certificate chain to a manufacturer is compromised. But similar assumptions seem to be necessary when trustees need to use authentic software on a secure platform, and these may come on top of human assumptions.

The ability of hardware to offer detailed logging of its use gives external observers evidence that there has been no inappropriate access—something that

is not available from human trustees, and that has been already considered for different applications, e.g., offering evidence that a website served specific content [39].

### 5.1 Secure Hardware Technologies

**Hardware Security Modules.** Hardware security modules (HSMs) have been used for decades to provide physical protection for high-value keys that do not need to be used often. They offer tamper-resistance and a variety of means for an authorized set of users to request actions be taken such as generating a key pair, exporting a public key, and using a protected private key to sign or decrypt given data. While these actions are often limited by default to a specific set of cryptographic algorithms that may not be compatible with the type of DKG protocols described above, custom firmware can often be loaded, and high-end HSMs even offer enclaves that support arbitrary code execution.

These features nicely match the requirements of an election. A key may be generated a month or more before any decryptions are required, and decryptions can be done together in batch in a single session. The principal drawback of HSMs is their operational complexity and high cost, especially if they are required to offer the flexibility and performance needed to execute DKG protocols.

**Secure Enclaves.** Secure enclaves are a newer technology offered, for example, by Intel's SGX and AMD's SEV-SNP technologies. They are capable of executing arbitrary code and providing attestations about the code they ran and the results produced, while offering strong isolation and encrypting all their communication with memory external to the CPU.

Although they are more flexible and far less expensive than HSMs, secure enclaves offer less robust physical protections, as evidenced by an already long history of side-channel vulnerabilities [34]. Another disadvantage of secure enclaves is that they only live as long as their host devices are powered and running. There are some means for preserving and reconstituting state. However, enabling such mechanisms could ruin the security of logging mechanisms that could be used to keep track of every key usage: so-called rollback attacks could be used to perform decryption tasks while escaping logging mechanisms. Generic solutions aiming at detecting rollbacks are being explored [3], but we will discuss more direct options below.

**Trusted Platform Modules.** Trusted platform modules (TPMs) are security components available in most commercial PCs. Their principal utility is to provide externally-verifiable attestation of the software running on a PC, and they have been successfully deployed to secure open source voting machines [36]. The principal difference between the secure enclave approach and TPMs is that TPMs offer no protection of the data being computed while secure enclaves offer protections to make it more difficult to access data externally. This makes TPM technology less attractive for handling the role of a trustee.

### 5.2 Resilient Usage of Secure Hardware

**What Could Secure Hardware Offer?** We now outline how secure trusted hardware instances, either permanently instantiated on an HSM, or volatilely in a TEE, can be leveraged to offer an alternative to human trustees. The approach that we outline here is radically different from the ones discussed above: there can be a single human trustee, and we will take advantage of the secure hardware instances to make the behavior of the human verifiable by any auditor. So, the trust that is traditionally required to be placed in humans and the software and hardware they use could now be replaced with trust in hardware only: the rest becomes verifiable by any independent auditor, offering a form of privacy verifiability.

**Overview.** The starting point is to deploy threshold encryption and decryption protocols on a set of secure hardware instances running on multiple devices. Each instance runs open-source trustee code, and offers an attestation that the expected code is running. The attestations can be tied from the specific hardware components to their manufacturers.

The instances are coordinated by an election administrator. The administrator submits inputs to them and collects their outputs. Crucially, the administrator is required to publish attestations of all operations performed by each instance. A trustee protocol will be defined as secure if the publication of valid attestations produced by the hardware component guarantees the expected security properties, that is, the only decryption operations that will ever be performed w.r.t. the election public key are those attested to by the secure hardware instance.

Intuitively, if the hardware is trusted, a single secure instance would be enough. However, we are concerned that this single device might fail, and the primary reason for using multiple instances is now to offer resilience to hardware failures. This is very different from the primary goal in the context of human trustees, namely to guarantee that a single trustee cannot silently decrypt ballots. Of course, running instances on a heterogeneous set of devices from a variety of vendors also offers additional privacy assurances, should a specific device fail to provide the isolation and attestation properties expected.

**Setup Phase.** To prepare for an election, a setup round generates signature keys that are used to establish secure communication channels between secure instances: each instance generates a signature key pair and submits the signature verification key to the administrator, who returns an election identifier `eid` and the signature verification keys provided by the other instances. Each instance attests to the list of signature verification keys that it will use in the context of election `eid`. This concludes the setup part of the protocol.

**The DKG.** From this moment on, we can rely on a set of existing protocols as described by Chen and Lindell for instance, who also prove their security [13].

1. *Distributed key generation.* This protocol runs a threshold DKG with a quorum of $k$ within a set of $n$ hardware instances, based on $n$ parallel executions of Feldman's VSS protocol [21].
2. *Refresh.* This protocol refreshes existing shares of a given secret, essentially by adding to the existing shares a set of freshly generated shares of 0.
3. *Removing a device.* This protocol makes it possible to invalidate the share held by a specific instance, essentially by running a refresh of the shares while excluding that instance from the refresh. At the end of this protocol, the quorum of $k$ is maintained, but only $n - 1$ instances hold shares.
4. *Adding a device.* This protocol makes it possible to add a new device, resuming the level of robustness from a set of $n - 1$ instances to $n$ instances (without any change in the quorum $k$).

These protocols can be used as follows. The election administrator starts by triggering the execution of the DKG protocol. At the end of this protocol, each hardware instance attests to its public view of the execution and to the resulting public key in particular. Furthermore, each instance also attests to the secure deletion of all the key shares it saw, except for the single one that needs to be kept for future decryption operations.

On a regular basis, the election administrator starts a refresh protocol. This provides proactive security [25] and protects against an adversary who would manage to extract key shares from an increasing number of devices (e.g., by successfully running side-channel attacks). Each refresh resets all the shares and renders the extracted shares useless. This does not need to reflect on decryption proofs, which can be made share independent [6].

At any point in time, an instance might be identified by the administrator as failing—be it rightly so, as a result of a power failure for example, or in order to maliciously exclude the instance. The purpose of such an exclusion might be to elude the requirement for the instance to attest the destruction of its stored key shares, hence weakening the privacy of the votes. However, any such failure statement is required to be followed by an execution of the device removal protocol by the other instances, which will effectively render the share of the excluded device useless.

Of course, such an exclusion reduces the resilience of the system to the failure of other devices: one may progressively reach a state in which more than $n - k$ instances have been removed, preventing any further operation. In order to avoid this situation, a new secure hardware instance must be started, its signature verification key distributed, and the device addition protocol must be executed.

**Decryption Operations.** Whenever tallying operations start, the election administrator submits the required decryption requests to the instances, who simply decrypt whatever they are asked to decrypt without any specific verification—which they could not perform anyway without visibility of the ballots submitted in the election. However, each decryption operation is securely logged by each device.

When the election is complete, the election administrator requires each device to erase its secret key share. Each device eventually provides an attestation of all the decryption operations that have been performed until its final key shares have been erased, and the final erasure is confirmed. All attestations produced by the devices are finally published for auditing.

The core benefit of this approach is that, if the audit succeeds, and under the assumption that a quorum of the devices on which the instances are running offer the advertised security guarantees, then we obtain guarantees on the secrecy of the votes. In effect, we introduce privacy verifiability in elections. Furthermore, should privacy verification steps fail, the election administrator can be identified as the faulty party, hence offering a form of accountability that may be harder to achieve using traditional approaches, when the origin of an incorrectly decrypted ballot may be harder to identify.

## 6 The Commitment Alternative

When it is possible, an even more radical alternative is to entirely eliminate trustees by replacing encryption with commitments as in [30]. This can be a viable approach for in-person voting systems in which vote collection devices are used to generate commitments consistent with the selections made by voters.

When in-person ballot collection devices are used to encrypt ballots that will ultimately be published, voter privacy is already dependent upon an assumption that the ballot collection device will not compromise voter privacy. So, it can be very appealing to use only this assumption and eliminate the trustee role entirely. Depending on local jurisdictional rules, each vote collection device can directly publish a proof of its own tally, or tallies and associated proofs can be aggregated by an election administrator who learns nothing more than the sub-tallies of each vote collection device.

This approach has many advantages, but it is more difficult to implement in remote voting scenarios.

## 7 Conclusions

It has become clear that the realities associated with the use of trustees in elections do not match the benefits derived in theory. We hope that the alternatives described herein will motivate further research that will better align the theory with practice.

**Acknowledgments.** This paper benefited from many valuable discussions, and we are happy to thank Melissa Chase, Véronique Cortier, Paul England, Pierrick Gaudry, Esha Ghosh, Kim Laine, Jack Richins, Douglas Wikström, and Hervey Wilson for their very helpful comments and inputs.

## References


1. Adida, B.: Helios: web-based open-audit voting. In: Proceedings of the 17th USENIX Security Symposium, pp. 335–348. USENIX Association (2008)
2. Adida, B., de Marneffe, O., Pereira, O., Quisquater, J.: Electing a university president using open-audit voting: analysis of real-world use of Helios. In: EVT/WOTE 2009. USENIX Association (2009)
3. Angel, S., et al.: Nimble: rollback protection for confidential cloud services. In: 17th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2023, pp. 193–208. USENIX (2023)
4. Bedrune, J.B., Campana, G.: Everybody be cool, this is a robbery! In: IACR Real World Crypto (2020)
5. Benaloh, J., et al.: STAR-Vote: a secure, transparent, auditable, and reliable voting system. In: EVT/WOTE 2013. USENIX Association (2013)
6. Benaloh, J., Naehrig, M., Pereira, O.: ElectionGuard design specification version 2.1.0 (2024). https://www.electionguard.vote/spec/
7. Benaloh, J., Naehrig, M., Pereira, O., Wallach, D.S.: ElectionGuard: a cryptographic toolkit to enable verifiable elections. In: 33rd USENIX Security Symposium, USENIX Security 2024. USENIX Association (2024)
8. Benaloh, J.C., Yung, M.: Distributing the power of a government to enhance the privacy of voters (extended abstract). In: Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing, pp. 52–62. ACM (1986)
9. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: pitfalls of the Fiat-Shamir heuristic and applications to Helios. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_38
10. Bulens, P., Giry, D., Pereira, O.: Running mixnet-based elections with Helios. In: 2011 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE 2011. USENIX Association (2011)
11. Burton, C., Culnane, C., Schneider, S.A.: vVote: verifiable electronic voting in practice. IEEE Secur. Priv. **14**(4), 64–73 (2016)
12. Carback, R., et al.: Scantegrity II municipal election at Takoma park: the first E2E binding governmental election with ballot privacy. In: 19th USENIX Security Symposium, pp. 291–306. USENIX Association (2010)
13. Chen, Y.H., Lindell, Y.: Feldman's verifiable secret sharing for a dishonest majority. Cryptology ePrint Archive, Paper 2024/031 (2024). https://eprint.iacr.org/2024/031
14. Clarke, D., Martens, T.: Real-World Electronic Voting: Design, Analysis and Deployment, chap. E-Voting in Estonia, pp. 129–141. CRC Press (2017)
15. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Distributed ElGamal à la Pedersen: Application to Helios. In: Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, pp. 131–142. ACM (2013)
16. Cortier, V., Gaudry, P., Glondu, S.: Belenios: a simple private and verifiable electronic voting system. In: Guttman, J.D., Landwehr, C.E., Meseguer, J., Pavlovic, D. (eds.) Foundations of Security, Protocols, and Equational Reasoning. LNCS, vol. 11565, pp. 214–238. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19052-1_14
17. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_9

18. Culnane, C., Schneider, S.A.: A peered bulletin board for robust use in verifiable voting systems. In: IEEE 27th Computer Security Foundations Symposium, CSF 2014, pp. 169–183. IEEE Computer Society (2014)
19. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_28
20. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985)
21. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th Annual Symposium on Foundations of Computer Science, pp. 427–437. IEEE Computer Society (1987)
22. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. **20**(1), 51–83 (2007)
23. Glondu, S.: Belenios specification. https://github.com/glondu/belenios/blob/2.5/doc/specification.tex, version 2.5
24. Glondu, S.: Belenios specification. https://github.com/glondu/belenios/blob/3.0/doc/specification.tex, version 3.0
25. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: how to cope with perpetual leakage. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_27
26. Hirschi, L., Schmid, L., Basin, D.A.: Fixing the achilles heel of e-voting: the bulletin board. In: 34th IEEE Computer Security Foundations Symposium, CSF 2021, pp. 1–17. IEEE (2021)
27. Komlo, C., Goldberg, I.: FROST: flexible round-optimized Schnorr threshold signatures. In: Dunkelman, O., Jacobson, Jr., M.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 34–65. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81652-0_2
28. Langford, S.K.: Weaknesses in some threshold cryptosystems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 74–82. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_6
29. Microsoft: end-to-end verifiability in real-world elections (2023). https://www.electionguard.vote/images/EAC%20Report%20Final.pdf
30. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Advances in Cryptology - CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer (2006). https://doi.org/10.1007/11818175_22
31. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_47
32. Swiss Post: Cryptographic primitives of the swiss post voting system (2024). https://gitlab.com/swisspost-evoting/crypto-primitives/crypto-primitives
33. Swiss Post: E-voting architecturedocument (2024). https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/raw/master/System/SwissPost_Voting_System_architecture_document.pdf
34. van Schaik, S., et al.: SoK: SGX.Fail: how stuff get eXposed. In: IEEE S&P Symposium (2024)
35. Verificatum: user manual for the verificatum mix-net (2022). https://www.verificatum.org/files/vmnum-3.1.0.pdf
36. VotingWorks: Install.md (2022). https://github.com/votingworks/vxsuite-complete-system/blob/main/INSTALL.md

37. Wikström, D.: Verificatum (2022). https://www.verificatum.org/
38. Willemson, J.: Creating a decryption proof verifier for the Estonian internet voting system. In: Proceedings of the 18th International Conference on Availability, Reliability and Security, ARES 2023, pp. 58:1–58:7. ACM (2023)
39. Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E.: Town crier: an authenticated data feed for smart contracts. In: Proceedings of the 2016 ACM CCS, pp. 270–282. ACM (2016)



# Revisiting Silent Coercion

David Chaum[1], Richard T. Carback[1], Jeremy Clark[2(✉)], Chao Liu[3], Mahdi Nejadgholi[2], Bart Preneel[4], Alan T. Sherman[3], Mario Yaksetig[1], Zeyuan Yin[6], Filip Zagórski[5], and Bingsheng Zhang[6]

[1] xx.network, Los Angeles, USA
[2] Concordia University, Montreal, Canada
j.clark@concordia.ca
[3] Cyber Defense Lab, University of Maryland, Baltimore County (UMBC), Baltimore, USA
[4] COSIC, KU Leuven and imec, Leuven, Belgium
[5] Department of Computer Science, Wroclaw University of Technology, Wroclaw, Poland
[6] Zhejiang University, Hangzhou, China

**Abstract.** We revisit "silent coercion" where an adversary gains access to a voter's credential without the voter's knowledge in an E2E verifiable, coercion-resistant Internet voting system. We argue that in this setting, casting an intended vote is impossible since the cryptographic backend can no longer distinguish the voter and adversary. However, we affirm that the voter can still act to nullify adversarial ballots, which is preferable to inaction. We provide a new instantiation of nullification using zero-knowledge proofs and multiparty computation, which improves on the efficiency of the current state-of-the-art. We also demonstrate an example voting system—VoteXX—that uses nullification. Our nullification protocol can complement new and existing techniques for coercion resistance (which all require voters to hide cryptographic keys from the coercer), providing a failsafe option for voters whose keys leak.

## 1 Our Contributions in Context

A well-studied application of cryptography is to voting systems, including those used in real-world governmental elections [5]. Among other things, cryptography can help provide election integrity (the final tally correctly reflects all ballots exactly as cast), while maintaining secret ballots—referred to as *end-to-end (E2E)* verifiable voting. These schemes generally offer unconditional integrity and rely a set of trustees, which constitute an EA where ballot secrecy is preserved if $m$-out-of-$n$ trustees are honest (for a configurable $m$ and $n$). We study undue influence and credential loss in the setting of remote or Internet voting.

An abstract of this paper appeared previously [8]. A technical report with expanded details and security proofs is also available [9].


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 38–54, 2026.
https://doi.org/10.1007/978-3-032-05036-6_3

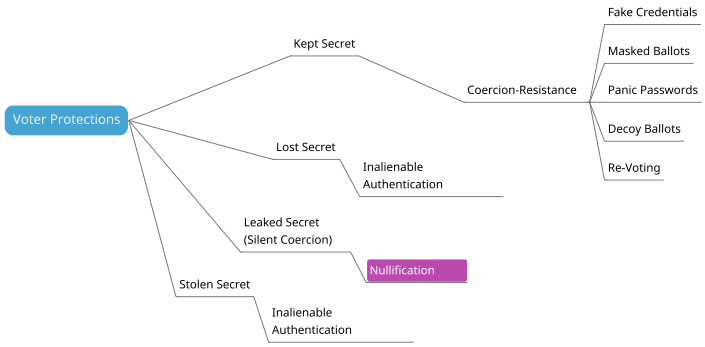


**Fig. 1.** Idea maze for protecting voters from coercion and credentials theft. Idea maze for protecting voters under various forms of credential loss or coercion. Nullification includes both passive approaches (*e.g.,* Caveat Coercitor) and our new active mechanism.

### 1.1 Ballot Privacy is not Always Enough

Definitions of ballot privacy roughly state the following: an adversary with access to the complete cryptographic transcript of the election plus control over a bounded set of other voters and election officials (and knowledge of everything the corrupted parties see), and access to Alice's receipt cannot establish how Alice voted any better than if they only had access to the final tally alone.

An implicit assumption is that Alice follows the protocol with a self-interest in preserving her privacy. What if Alice sabotages her own privacy? First, consider why. If the system design allows for receipts to leak information about how they were cast, Alice might be able to take payment for casting her vote for a specified candidate, and is incentivized to have her receipt serve as proof that she complied. She could also be threatened or coerced to vote a particular way, and an adversary savvy to the details of the voting system could demand a receipt constructed according to the coercion instruction. As common in the literature, we use the term coercion resistance to mean a system defeats both vote buying and actual coercion.

A prominent example of a remote cryptographic voting system that provides basic ballot secrecy but not coercion resistance is Helios [2,3], which originally included a "coerce me" button that would spit out a proof of how the voter voted [2] (a form of critical design to educate the voter on the difference between voter privacy and coercion-resistance). Coercion resistance is challenging in a remote setting because the adversary can observe the voter cast their ballot, or more simply, take the voter's authentication credentials and vote on the voters' behalf themselves.

### 1.2 Coercion Resistance Mechanisms

**Table 1.** Five basic mechanisms from the literature for coercion resistance. The table excludes procedural in-person fallback mechanisms (*e.g.,* Swiss postal voting) in favor of purely cryptographic protocols applicable to remote settings.

| Type | Example | Non-Standard Assumptions |
|---|---|---|
| Fake credentials | JCJ (2005) [24] | Voter maintains at least one secret (real private key) from coercer and registers prior to coercion. Voter can simulate zero knowledge proofs to match lies to coercer. Voter knows of one honest election trustee. Under coercion, voter can always vote their intention but requires a moment alone (any time before or after coercion). |
| Masked ballots | WeBu09 (2009) [33] | Voter maintains at least one secret (integer offset) from coercer and registers prior to coercion. Voter can perform arithmetic in head before lying to coercer. Under coercion, voters can at least spoil their ballot. |
| Panic passwords | Selections (2011) [12] | Voter maintains at least one secret (real password) from coercer and registers prior to coercion. Voter can make up a fake password before lying to coercer. Under coercion, voter can always vote their intention but requires a moment alone (any time before or after coercion). |
| Decoy ballots | RS-Voting (2012) [7] | Not all voters have ballots and voters can opt to receive a decoy ballot that will not be counted. Voter registers prior to coercion. Under coercion, voters can use a decoy or deny having a real ballot. |
| Re-voting (with E2E) | VoteAgain (2020) [26] | Voter maintains an authentication mechanism that cannot be duplicated by the coercer (or eliminated). Voter must submit a corrective ballot after coercion. Assuming this, voter can always vote their intention. |

Many papers have examined a relatively small set of approaches to providing coercion resistance (see Table 1), which we assume are overlayed on an E2E cryptographic voting scheme that provides ballot privacy and tally correctness.

Four of the approaches in Table 1 are based on faking information (credentials, masks, panic passwords, and decoy ballots) to give to the coercer. Decoy ballots can be distinguished by providing a game-theoretic solution (voters can request fake ballots from the EA and use these to flood the market, driving down the economic feasibility of coercion/vote selling) rather than a cryptographic one (although cryptography is used in other aspects of the system). The remaining three make similar cryptographic assumptions about voters being able to keep secrets and deceive the coercer.

The other approach, re-voting, enables votes to be updated over time, counting only the last ballot. Under coercion, a voter allows the adversary to cast their ballot for them, but then secretly changes it after. This mechanism is less rigorous than the fake ballot mechanism as the adversary can simply ensure the voter does not have the ability to change their vote after coercion (by supervising the voter until the election ends). For this reason, re-voting is often offered in a hybrid remote/in-person model. Voters vote online optimistically and then can correct coercion in-person as necessary.

**Table 2.** Types of key loss. Traditional coercion-resistant voting strategies consider the first row ("normal conditions"), whereas we argue all rows of key loss (see Eyal [16]) should be addressed.

| Type | User has key? | Adversary has key? | Countermeasure |
|---|---|---|---|
| Normal conditions | Yes | No | Coercion resistance |
| Lost secret | No | No | Inalienable authentication |
| Leaked secret | Yes | Yes | Nullification |
| Stolen secret | No | Yes | Inalienable authentication |

### 1.3 Relationship to Authentication

It is maybe not be apparent but coercion resistance is related directly to authentication. As pointed out by Hirt and Sako [21,22], a basic axiom of coercion resistance is: if the adversary has/knows/is everything that the voter has/knows/is, the voting system cannot distinguish the voter from the adversary, and therefore cannot provide a mechanism to register the true voter's vote and not the adversary's. Reconsider the coercion-resistance mechanisms above and see that they rely on a secret value or authentication mechanism that can still distinguish a voter under coercion. This is also why most coercion resistance mechanisms assume the voter can register prior to coercion, otherwise the adversary can coerce the voter to register with authentication values only the coercer knows.

For fully online voting, maintaining secrets is particularly challenging because digital forms of secrets are generally "something you have/know" rather than "something you are," and secrets can be stolen through a variety of covert means, including malware and physical access to the voter's device. Directly stealing a secret side-steps known coercion-resistant mechanisms—each assumes somewhere that the voter knows of, and possibly plays a role in, resisting coercion (such as providing a fake secret or by voting again with their real secret).

We have used the term "stolen" loosely. In fact, there are three specific failures (see Table 2) a user can experience in maintaining a secret value [16]: secrets can be *lost*, where neither the user or adversary has the credential; *leaked*, where the user and adversary both have the credential; or *stolen*, where the user loses the credential and the adversary gains the credential (so only the adversary has the credential).

For lost and stolen secrets (Rows 2 and 4), the voter loses their secret. Our countermeasure is to ensure voters cannot lose their secret through *inalienable authentication*, which is a protocol where the voter proves that they have committed their secret to their own human memory without revealing the secret.

The threat of a leaked credential (Row 3) is under-appreciated in the voting literature. It is easy to think that coercion resistance already covers this attack; however, coercion resistance addresses only *overt* attempts by a coercer to learn the key, whereas we also need to consider *covert* attacks for which voters may be unaware. Grewal *et al.* [18] term leaked credentials as "silent coercion." The reader might not consider this "coercion" as there is no pressure or threat,

however we do not attempt to adjust the terminology already accepted in the voting literature.

### 1.4 Silent Coercion

It is easy to think Row 3 is an unsolvable problem: if the adversary knows every secret the voter knows, the remote system cannot distinguish the voter and adversary. The truth of this unpleasant fact implies that a dimension of the problem indeed is unsolvable: the voter cannot always cast their true intent and prevent the adversary from doing the same. Nevertheless, the voter still has a course of action: the voter (or either party) can sabotage (*nullify*) the vote, preventing themselves and the adversary from casting a ballot.

Caveat Coercitor [18] is an extension to JCJ [24]. Like JCJ, it provides coercion resistance through fake credentials. However it adds two extensions: (1) any vote cast multiple times with the same credential but different candidates (in at least one occurrence) is discarded from the tally, and (2) the number of discarded ballots is made public as an indicator of the level of coercion that was present in the election. Caveat Coercitor has two main drawbacks. First is the complexity of the election trustees tabulating the tally. It inherents the quadratic (in the number of cast ballots) complexity of JCJ and the coercion evident extension adds a linear work making it cubic. As JCJ is considered too slow for practical election sizes already [13], Caveat Coercitor is mainly a theoretical advancement. The second drawback is more minor but Caveat Coercitor cannot compose easily with all the coercion resistant mechanisms in Table 1—it works with fake credentials and perhaps others but it cannot work with revoting, as voters who change their mind are excluded from the tally.

We argue there is room for more work on nullification and exploring different approaches might be worthwhile. This paper presents a different design that sidesteps these two drawbacks of Caveat Coercitor—our extension (a) is mostly linear with one quadratic component that can run concurrently [15] over the nullification period and (b) our extension can compose with revoting (along with fake credentials and panic passwords).

Our design also differs from Caveat Coercitor where nullification is an active task taken by a voter (or an enlisted helper) in responses to a ballot that does not reflect their voting intent. The passive nullification of Caveat Coercitor removes a task for the voter, likely improving usability, however our active nullification offers finer grain control: it can be triggered on an adversarial ballot even if the voter abstained from voting, and it can be not triggered in the case of a revote where the voter changes their mind. Whether more control through more decisions is a net benefit for voters will likely divide researchers, but we think it is worth laying out multiple options in the academic literature.

### 1.5 Other Examples of Nullification

Other instances of nullification appear in voting systems that allow voters to submit encrypted deltas to neutralize or adjust their previous votes. In the masked

voting approach of Wen and Buckland [33], each voter receives a secret mask and casts a ballot encrypted as the difference between their vote and this mask. A coerced voter can later submit a masked ballot that offsets their original vote, effectively nullifying it in the additive tally. A similar pattern occurs in the update scheme of Kulyk, Teague, and Volkamer [25], where a voter's row on the *bulletin board (BB)* is multiplicatively combined from multiple encrypted values. To cancel their vote, a voter can submit the multiplicative inverse of their earlier choice, neutralizing its effect on the final tally. These schemes provide important protections against overt coercion or vote selling, where the voter is aware of and responding to coercion. However, they do not address silent coercion scenarios, where an adversary obtains the voter's credentials and casts a ballot without the voter's knowledge. In other words, while these are examples of nullification, they are not designed to prevent or recover from undetected adversarial voting, as in Caveat Coercitor [18] and our work.

As an additional remark, the cryptographic structure of our nullification protocol (see Sect. 3.1) parallels the proof techniques used by Kulyk, Teague, and Volkamer [25]. In both cases, the core idea is to construct a zero-knowledge proof that supports a disjunctive claim—either the value was submitted by an eligible voter who knows the signing key or the value has no effect (*e.g.,* add a 0 or multiply by 1).

### 1.6 Our Contributions

We provide a protocol for *nullification* that can work in concert to resist silent coercion where voter keys or secrets are leaked to the adversary.

The mechanism is an overlay or add-on for coercion resistant voting system and does not compete with any of the mechanisms in Table 1. In fact, even within the table, the basic mechanisms are complimentary to some extent (*e.g.,* an E2E voting system might use panic passwords and revoting and decoy ballots). The basic idea of our paper is that if a voter is not coerced, they vote as normal; if they are coerced but their key is not leaked, they use a mechanism from Table 1; if their key is leaked, they use nullification which is better than doing nothing (see Fig. 1).

With this said, we want to be clear that our nullification protocol cannot be simply glued onto Helios or JCJ or Selections. Our protocol expects a certain interface to work, and some additional changes to the voting system can greatly enhance its usability and safety. For this reason, we do sketch a basic voting system (VoteXX) to demonstrate how a voting system can compose with nullification.

## 2 Preliminaries

We adopt a number of common cryptographic primitives from the voting literature. All operations are performed in the same DDH-hard elliptic curve group. Digital signatures are performed with the Schnorr signature scheme. Encryption

is performed with exponential Elgamal [14], which can be augmented with *distributed key generation (DKG)* and threshold decryption (for $m$ out of $n$ key holders [31]). As standard in the voting literature, we assume the EA has a threshold public key and do not collude beyond the threshold. We use standard $\Sigma-\mathsf{Protocols}$ to prove knowledge of discrete logarithms (Schnorr [32]), knowledge of representations (Okamoto [29]), and knowledge of Diffie-Hellman tuples (Chaum-Pedersen [11]), which also corresponds to ElGamal re-randomizations and decryptions. We also use techniques to allow the trustees to compute jointly, verifiably (*i.e.,* produce $\Sigma-\mathsf{Protocol}$ proofs), and privately on ElGamal ciphertexts the following: (i) a random shuffle of ciphertexts [28], and (ii) the evaluation of an *binary gate* operations based on their logic lookup table (mix and match [23]).

## 3 New Nullification Protocol

We explain our nullification protocol by stating its front-end requirements and introducing the semi-trusted "hedgehog" agents.

### 3.1 Front-End Requirements

Our nullification protocol is an overlay, add-on, or "back-end" for the "front-end" of a E2E coercion resistant voting schemes. As a backend, nullification adds protection for Row 3 in Table 2 to systems that already protect against Row 1. However none of these front-ends will "just work" with nullification, they need to be modified to compose correctly. For space, we cannot fully detail how to add nullification to every scheme however we will sketch how to do it with JCJ [24] and Selections [12], as well as showing a new front-end that adds an additional feature (called hedgehogs). JCJ is based on fake credentials, and Selections allows re-voting in addition to panic passwords, so Selections as augmented below actually provides three lines of defence against coercion (panic passwords, re-voting, and nullification).

The details of E2E voting systems vary widely but we assume a basic architecture. Registration phase: voters register for the election which may or may not post registration data to a BB—an append-only, non-equivocating broadcast channel. Voting phase: voters cast ballots which do not reveal how they voted and are posted on the BB. Tallying Phase: the votes on the BB are used to produce a final tally. Finally we introduce a Nullification Phase: voters (or their agents) have a window of time to inspect the cast ballots and potentially nullify adversarial ballots submitted using their key.

In JCJ, a ballot from the voter comprises their vote, the asserted credential (secret key) of the voter (we place this in the exponent to make it easier to compare side-by-side with Selections), and some zero knowledge proofs of correct formation that we omit: $\left\langle [\![\mathsf{vote}]\!], [\![g^{\hat{\mathsf{sk}}}]\!] \right\rangle$. In Selection, a ballot consists of $\left\langle [\![\mathsf{vote}]\!], g^{\hat{\mathsf{sk}}}, [\![\mathsf{pk}]\!] \right\rangle$ where $\hat{\mathsf{sk}}$ is the asserted password of the voter, and $[\![\mathsf{pk}]\!]$ is the

entry from the voting roster (created at registration time). If $g^{\hat{\mathsf{sk}}} = \mathsf{pk}$, the vote is tallied, otherwise it is discarded (later after anonymization).

We now consider the front-end requirements to work with our nullification back-end:

1. The voter's eligibility to vote is determined by knowing a secret key,
2. The voter can see how they voted from the BB, and
3. The voter can add a secret "flag" to their own cast ballot(s) that can be used as a filter after the ballots are anonymized.

The first requirement is already true of must coercion resistance schemes, including those based on fake credentials [24], panic passwords [12], and re-voting [26]. The second requirement sounds like a strange requirement because, until now, the literature assumes voters are posting their own ballots or in the case of coercion, the voter at least knows coercion is happening. But in our threat model of leaked keys (Row 3 of Table 2), an adversary might vote with a voter's key and the voter is not unaware and, in fact, has no way to tell from the BB because the ballot components are all encrypted under the EA's public key. This is also true of hybrid systems (Internet voting with in-person fallback) where a voter might be able to distinguish their actions from the adversary by attending in-person with government photo identification (or similar).

The fix for JCJ and Selections is to require a voter to encrypt every ballot twice: once under the EA's key and once under the voter's key. The voter must then add a proof (using a $\Sigma-\mathsf{Protocol}$) that: (i) the two encryptions are of the same value (standard plaintext equality) and (ii) the public key used in the second encryption matches the asserted voter credential in the ballot (possible using Mauer's abstraction [27]). In JCJ, voters looking for other ballots cast with their keys will use trial decryption on every ballot to see if any belong to them. In Selections, the ballot includes the password (real or panic) in a deterministic commitment (rather than a randomized encryption) so it is simpler: the voter checks if a commitment to their real password ever shows up in a submitted ballot they did not actually submit. If they find one, they can decrypt the ballot and take an action: leave it if it is for their candidate, re-vote if there is time left, and nullify as a last resort if voting has already closed.

The third requirement is that a flag is added by the voter. The voter will construct a "selector" vector of encrypted values, with one value for every ballot submitted during the voting phase. For example, if there are 5 ballots and the voter wants to flag the 3rd ballot, the vector could be: $\langle [\![0]\!], [\![0]\!], [\![1]\!], [\![0]\!], [\![0]\!] \rangle$. In this example, the unflagged value is 0 and the flagged value is 1. If a voter flags a ballot with ($[\![1]\!]$), it must know the private key used in the construction of the ballot; otherwise, any voter could nullify any vote. However, when the voter submits a false flag ($[\![0]\!]$), it does not need to know the associated key. To enforce these constraints, the voter must construct a ZK proof to prove that: [for each flag, (it is an encryption of 0) or (it is an encryption of 1 and I know $\mathsf{sk}$ corresponding to the $\mathsf{pk}$ used in a component of the ballot)].

In Case 1, the voter proves ($\mathsf{flag} = [\![0]\!]$). For exponential ElGamal, assume $\langle c_1, c_2 \rangle = \mathsf{Enc}(m) = \langle g^r, g^m y^r \rangle$ for generator $g$, public key $y = g^{\hat{sk}}$, and message $m$. A proof it encrypts $\hat{m}$ is equivalent to proving $\langle g, c_1, y, c_2 \hat{m}^{-1} \rangle$ is a DDH tuple, which can be done with the Chaum-Pedersen $\Sigma-\mathsf{Protocol}$. Call this subproof A. In $\Sigma-\mathsf{Protocol}$ format, its transcript is $\langle a_A, e_A, z_A \rangle$.

In Case 2, the voter proves a conjunctive statement: ($\mathsf{flag} = [\![1]\!]$) and it knows $sk$, which corresponds to $pk$ for the associated voter's public key. Call the subproof that ($\mathsf{flag} = [\![1]\!]$) B. It is implemented the same as in subproof A, with transcript $\langle a_B, e_B, z_B \rangle$. Call the proof of knowledge of $sk$ subproof C, which can be implemented with a $\Sigma-\mathsf{Protocol}$ due to Schnorr: $\langle a_C, e_C, z_C \rangle$. To summarize, the hedgehog proves: $\Pi :=$ [A `OR` (B `AND` C)] for each flag.

Further, the resulting proof can be made non-interactive (typically in the random oracle model with the Fiat-Shamir heuristic [17], in its strong form [4], but other heuristics exist [20]). Specifically, the prover generates a single challenge $\hat{e}$ for $\Pi$. To handle the conjunction within Case 2, $e_B = e_C$; for the disjunction across the cases, $\hat{e} = e_A + e_B$. In Case 1, the prover computes $\langle a_A, e_A, z_A \rangle$ and simulates $\langle a_B, e_B, z_B \rangle$ and $\langle a_C, e_B, z_C \rangle$. In Case 2, the prover simulates $\langle a_A, e_A, z_A \rangle$ and computes $\langle a_B, e_B, z_B \rangle$ and $\langle a_C, e_B, z_C \rangle$.

The final consideration is where in the tallying process are the flags used? This will be specific to the voting system. For JCJ and Selections, a suitable design is to add flags to ballots before tallying. As nullification vectors arrive, the EA can flatten them into a single vector that is the logical AND of all vectors, using a binary lookup and the Mix and Match protocol [23]. Probably the simplest way to nullify a ballot that has been flagged is to just replace the key used to cast the ballot with a "junk" key value such as 0, again using a lookup table (outputs are always rerandomized).

$$[\![g^{\hat{sk}}]\!] \leftarrow \begin{array}{c|c} \textbf{Flag} & \textbf{Out} \\ \hline [\![0]\!] & [\![g^{\hat{sk}}]\!] \\ [\![1]\!] & [\![0]\!] \end{array} \tag{1}$$

However a special purpose trick can be used as a shortcut when the discrete logarithm is hard and the encryption is additively homomorphic (both are true in JCJ and Selections): a beacon selects a random exponent $\mathsf{rand}$ after voting closes, nullification uses $[\![0]\!]$ as a non-flag and $[\![\mathsf{rand}]\!]$ as the flag, and then simply: $[\![g^{\hat{sk}}]\!] \leftarrow [\![\mathsf{flag}]\!] \cdot [\![g^{\hat{sk}}]\!] = [\![\mathsf{flag} + g^{\hat{sk}}]\!]$. When the flag is $[\![0]\!]$, $[\![g^{\hat{sk}}]\!]$ is unmodified and when it is $[\![\mathsf{rand}]\!]$, $[\![g^{\hat{sk}}]\!]$ is "ruined" by casting it to an arbitrary integer (that is unpredictable at the time $[\![g^{\hat{sk}}]\!]$ is submitted to the BB) and the secret key can no longer be extracted (due to the discrete logarithm problem). Since the same ballot can be nullified an arbitrary number of times, we assume some system-level restrictions limit an adversary that nullifies so much, the value returns to its original $[\![g^{\hat{sk}}]\!]$ (an *integer overflow* attack). These methods effectively turn any real credentials into fake credentials (or real passwords into panic passwords). Since the voting system will already eliminate (in a privacy-preserving fashion) fake credentials, the tally can be computed as normal after this pre-processing step.

The general approach (using lookups) is expensive: each nullification request requires work linear in the number of ballots, making it quadratic overall. However all the lookup tables can be pre-computed concurrently with the nullification phase of the election, and the flattening (logical AND) can be done immediately by the EA once it receives a new nullification request. Assuming the EA can keep up nullification requests, only an additional linear amount of work is needed to be done at tallying time. The special purpose trick is fast for the EA: it requires only multiplications and no exponentiations.

*Additional Remarks.* Nullifications are not individually revealed on the BB, but the difference between the provisional and final tallies provides coercion evidence analogous to that in Caveat Coercitor. A coercer might attempt to verify whether a coerced vote was nullified by casting a vote using the voter's credential and later checking whether a nullification occurred. Our protocol, however, does not reveal nullification events on a per-ballot or per-credential basis. Ballot nullifications are included only in the transition from the provisional to the final tally, and the nullification request itself is either hidden (via cryptographic flagging) or aggregated. In particular, a coercer who holds the voter's credential cannot determine whether a nullification occurred.

The JCJ definition of coercion resistance includes the ability to avoid forced abstention attacks [24]. In the passive nullification of Caveat Coercitor [18] and the active nullification of our work, a voter can cryptographically prove that the ballot associated with a specific credential was nullified. To evade forced abstention, nullification needs to be composed with a coercion resistance mechanism (see Fig. 1). For example, with fake credentials, the voter can prove credentials are nullified but cannot prove the credential is real. A promising direction for future work is to develop deniable nullification mechanisms where no individual voter can prove that they nullified a ballot.

### 3.2 Hedgehogs

We will present one final idea: can we allow voters to enlist helpers to help them with nullification? This is useful for voters who are not savvy about coercion, which likely overlap with voters likely to leak their keys unintentionally. The idea is to let a voter prearrange with a helper: "I want to vote for Alice and this is the key I will use; if you see a vote for Bob on the BB with my key, please nullify it."

This works if the helper is *fully trusted*. Precisely we are making two trust assumptions: (i) The helper will nullify if there is a vote for Alice, and (ii) the helper will not nullify if there is a vote for Alice (which it has the power to do). The first assumption can be addressed by enlisting many helpers, hoping that at least one will act. The second assumption is difficult to fix. However we will now propose a new front-end voting system that does not require the second trust assumption. In other words, helpers are *semi-trusted*: they are trusted to

**Table 3.** The threat model for different actors in an election with a blue and red candidate. The voter prefers the blue candidate and the adversary the red candidate. The symmetry between the adversary with the leaked credentials and the voter shows why the voter cannot register their true intent reliably. The asymmetry between the voter and the hedgehog is the contribution of the VoteXX front-end.

| Action | Blue Voter | Red Adversary (no credentials) | Red Adversary (with both credentials) | Blue Hedgehog (blue credential only) |
|---|---|---|---|---|
| View bulletin board | Yes | Yes | Yes | Yes |
| Cast a blue vote | Yes | No | Yes, but would not | Yes |
| Cast a red vote | Yes, but would not | No | Yes | No |
| Nullify voter's blue vote | Yes | No | Yes | No |
| Nullify adversary's red vote | Yes | No | Yes | Yes |

**Voting.**
Each voter completes voting online. We assume voters have already registered two keys or passphrases: one used to vote YES and one for NO. At completion, each voter will have submitted their ballot using a passphrase from registration.

1. The value $\mathsf{nonce}$ is a parameter of the election.
2. To mark a ballot for YES, the voter uses their YES passphrase to generate $\mathsf{sk}_{\mathsf{yes}}$ and uses this key to sign $\mathsf{n}_0$: $\sigma_{\mathsf{yes}} \leftarrow \mathsf{Sig}(\mathsf{nonce})$. Corresponding values are used to vote NO.
3. The voter uses the EA's threshold encryption scheme to compute $\mathsf{ballot} \leftarrow \langle [\![pk_{\mathsf{yes}}]\!], [\![\sigma_{\mathsf{yes}}]\!], \pi_{\mathsf{ppk}} \rangle$, where each group element of $\sigma$ is individually encrypted and $\pi_{\mathsf{ppk}}$ is a proof of plaintext knowledge using the Chaum-Pedersen $\Sigma-\mathsf{Protocol}$.
4. The voter submits $\mathsf{ballot}$ over an anonymous channel to the BB. The EA marks it as invalid if it is an exact duplicate or if the proofs are invalid.

Protocol 1: Voting Protocol.

actually act when they are supposed to, but they cannot act against the voter's wishes. We call semi-trusted helpers *hedgehogs*.

Hedgehogs are introduced not because they possess capabilities that voters lack in principle, but because they are more reliable agents for acting on those capabilities in practice. While it is true that a technically aware voter could inspect the BB and identify unauthorized ballots submitted using their credential, most voters are unlikely to do so—especially if they are unaware that their key has been compromised. Hedgehogs serve as dedicated monitors who scan the BB for suspicious ballots and can proactively initiate nullification before the voter ever notices. Hedgehogs might be benign election watchdogs or partisan parties who are selected by voters who share the same political allegiance. For

**Provisional Tally.**
After the voting period ends, the EA produces a verifiable provisional tally.

1. The EA takes the list of $\langle [\![\mathsf{pk}]\!], [\![\sigma]\!] \rangle$, verifiably shuffles them, then threshold-decrypts them: $\langle \mathsf{pk}, \sigma \rangle$.
2. For each ballot, the ballot is marked invalid if $\sigma$ does not verify under its corresponding $\mathsf{pk}$.
3. For each valid signature, $\mathsf{pk}$ is matched to its entry on the $\mathsf{Roster}$. The EA determines if it is a YES or NO key, and counts the vote only if it is the only ballot cast that corresponds to that roster entry.

**Nullification.**
The goal of nullification is to allow voters to modify their cast ballots, particularly in the case of coercion. Unlike other protocols, voters can enlist the help of others parties, called hedgehogs. The nullification period runs after the provisional tallying. If the provisional tally contains $\mathsf{pk}_{\mathsf{no}}$, it can be nullified using $\mathsf{sk}_{\mathsf{yes}}$ (the "opposite" key). In other words, casting a YES and nullifying a NO vote use the *same* key, as these two actions are aligned in their intention.

1. At any convenient time, before or after voting, the voter covertly communicates with a hedgehog to develop a coercion-resistant strategy. For example, assume the following strategy: the voter wants to vote YES and reveals $\mathsf{sk}_{\mathsf{yes}}$ to the hedgehog, along with $\langle \mathsf{pk}_{\mathsf{yes}}, \mathsf{pk}_{\mathsf{no}} \rangle$. They request the hedgehog engage in nullification if $\mathsf{pk}_{\mathsf{no}}$ is in the provisional tally.
2. Using the $\mathsf{Roster}$ and set of valid signatures from the provisional tally, the EA reformats the election data into two lists. The first list establishes, in arbitrary order, the set of $\mathsf{pk}_{\mathsf{no}}$ keys from voters who cast valid votes for YES (call it $\mathsf{yesVotes}$). The second list contains $\mathsf{pk}_{\mathsf{yes}}$ from voters who voted NO.
3. For example, assume YES received six votes in the provisional tally. $\mathsf{yesVotes}$ consists of six $\mathsf{pk}_{\mathsf{no}}$ keys. If the hedgehog wants to nullify the fourth key, it prepares a list of encrypted "flags" marking the ballot it wants to nullify: $\langle [\![0]\!], [\![0]\!], [\![0]\!], [\![1]\!], [\![0]\!], [\![0]\!] \rangle$.
4. The first encrypted flag corresponds to the first $\mathsf{pk}_{\mathsf{no}}$ in $\mathsf{yesVotes}$. The hedgehog adds a proof to this list using the nullification ZK protocol. Concisely, the proof statement is: [for each flag, (it is an encryption of 0) or (it is an encryption of 1 and I know $\mathsf{sk}_{\mathsf{no}}$ corresponding to this $\mathsf{pk}_{\mathsf{no}}$)].

**Final Tally.**
After the nullification period ends, the EA produces a verifiable final tally.

1. The EA takes all the encrypted flags for the first $\mathsf{pk}_{\mathsf{no}}$ key in $\mathsf{yesVotes}$ and computes its logical AND under encryption using the mix and match secure function evaluation (SFE) protocol [23]. It repeats this process for the remaining $\mathsf{pk}_{\mathsf{no}}$ keys.
2. The EA takes the list of encrypted OR-ed flags, sums them under encryption, and verifiably threshold-decrypts the result. The EA subtracts this value from the number of YES votes in the provisional tally to produce the final tally for YES votes.
3. The EA repeats Steps 1–2 for each $\mathsf{pk}_{\mathsf{yes}}$ key in $\mathsf{noVotes}$.

Protocol 2: Tallying Protocol (including nullification).

example, each political party or candidate could offer a hedgehog to their voters (Table 3).

Our approach is to redesign a front-end voting system (we call VoteXX) that uses multiple keys to identify voters, instead of just one (as in JCJ and Selections), where the keys are tied to specific candidates in the election. We will use the example of two keys: $\mathsf{pk}_{\mathsf{yes}}$ to vote yes and $\mathsf{pk}_{\mathsf{no}}$ to vote no. The front-end is based on signature-based mix network voting, *cf.* [6,19,30]. The nullification period runs after a provisional tallying phase. If the provisional tally contains $\mathsf{pk}_{\mathsf{no}}$, it can be nullified using $\mathsf{sk}_{\mathsf{yes}}$ (the "opposite" key). In other words, casting a YES and nullifying a NO vote use the *same* key, as these two actions are aligned in their intention. Thus voters who want to vote yes will provide the hedgehog with only $\mathsf{sk}_{\mathsf{yes}}$ and will not provide $\mathsf{sk}_{\mathsf{no}}$.

VoteXX with nullification is given in Protocols 1 and 2. VoteXX is not necessarily better or worst than augmenting JCJ or Selections with nullification, it just shows the flexibility of nullification and illustrates different trade-offs that are possible:

- **Hedgehogs:** In VoteXX, helpers are only semi-trusted while in JCJ/Selections they are fully trusted.
- **Normal coercion-resistance:** In VoteXX, there is no support for normal coercion resistance (when the adversary does not know the key), while JCJ offers fake credentials and Selections offers both panic passwords and re-voting.
- **Multiple candidates:** In VoteXX, each additional candidate in the election requires an additional inalienable passphrase to be memorized by the voter. In JCJ/Selections, the requirements on the voter does not change with the number of candidates.
- **Provisional Tally:** In VoteXX, a provisional tally is published and then adjusted based on nullification, which reveals early information about the tally. In JCJ/Selections, nullification is done prior to tabulating the tally.
- **Efficiency:** In VoteXX and Selections, tallying is linear without nullification while JCJ is quadratic without nullification.

## 4 Variants and Extensions

*Vote Flipping.* Our design supports a variety of options for implementing the semantics of nullification, including what we call "cancel" or "flip." We recommend cancel, which is the default and is what has been described thus far. Consider a vote that might have been nullified by one or more entities. We will describe the case for when there are two ballot choices. Assume that this vote selects from one of two ballot choices numbered $0, 1$. With *cancel*, the vote is cancelled if and only if at least one entity nullified it (and this idea can be generalized to at least $t$ entities for some threshold $t$). With *flip*, the vote becomes $x+y \bmod 2$, where $x$ is the ballot choice of the vote, and $y$ is the number of times

the vote was nullified. Intuitively, cancel gives the voter the ability to cancel the vote, whereas flip gives the voter the ability to randomize the vote.

Each of these options can be implemented using different algebraic operations during Step 1 of the third phase (Final Tally) of Protocol 2: `AND` for cancel (realized with a homomorphic addition of the encrypted flags for each ballot followed by a plaintext equality test with $[\![0]\!]$), and `ADDITION` modulo 2 for flip (realized with mix and match. Step 2 replaces the final summation with a verifiable shuffle and threshold decryption of the flag set for each key). We view flip not as re-voting, but as "randomizing" the vote, which is a form of nullification.

As we point out below, under stronger assumptions, there are some use cases in which flip can be used to re-vote. Also, nullification can be used as an overlay in conjunction with re-voting strategies. A useful application of flip arises for a common form of low-intensity coercion. Suppose during remote voting at home, a coercer tells their spouse to vote for Alice and watches them comply, but the coercer does not collect the spouse's keys. The spouse can later flip their vote to Bob without the coercer knowing.

*Coercion Evidence.* Our system also supports a form of coercion evidence, though in a different style from Caveat Coercitor [18]. Rather than embedding an explicit coercion-evidence test in the tallying process, we allow coercion evidence to be derived by computing the difference between the provisional tally (before nullification) and the final tally (after nullifications are processed). This difference quantifies the effect of adversarial interference, credential leakage, or voter-initiated ballot cancellations, and can be used by observers to assess the integrity of the election outcome.

*Inalienable Authentication.* A related but orthogonal direction to nullification is inalienable authentication (recall Fig. 1): ensuring that only the voter—rather than their device—possesses the factor needed to cast a valid vote. The idea traces back to Achenbach et al. [1] and is further discussed in ecash 2.0 [10], where a credential is bound not to a device but to a secret memorized by the user. This offers a path to prevent adversaries from voting even when they control the voter's device or possess all stored credentials. However, while prior work assumes such a mechanism exists, no concrete implementation has been adopted in remote voting systems. We view this direction as promising but currently underdeveloped, particularly in terms of usability and resistance to coercion.

There remain substantial challenges in designing usable inalienable authentication. A voter must prove knowledge of a secret without revealing it, and without offloading trust to a potentially compromised device. This raises questions about entropy, memorability, leakage, and fallback. As a conceptual example, a voter could demonstrate knowledge of a passphrase by computing a blinded linear function over a challenge (e.g., $c \cdot x + r$), using a physical aid such as a printed wheel. Though primitive, this illustrates that inalienable authentication has not been fully explored as a human-verifiable building block for remote voting. We leave the development of practical, zero-knowledge, and simulatable versions of this primitive to future work.

*Voting in Person or by Mail.* To support the existing voting infrastructure, VoteXX can allow for a setting where the voting is accomplished by mail or in precincts using paper ballots. This capability can be achieved by incorporating a code-voting protocol, such as that used in Remotegrity [34].
*Malware Protection.* Malicious software and can alter the operations performed by the voter. VoteXX allows for a two-phase voting process. In Phase 1, the user submits a vote or a vote commitment. In Phase 2, using a different device, the voter checks if the submission is correctly posted on the BB. Optionally, this extension can include an additional set of keys, where the user submits a payload signed with the additional keys and thereby "*locks in*" their submission.

## 5 Conclusion

Nullification offers a response to credential compromise in voting systems that are otherwise coercion resistant. While our protocol puts forward a concrete proposal, the broader design space remains far from settled—especially around the practicalities of hedgehogs; the composability of nullification with existing mechanisms like panic passwords, re-voting, and fake credentials; and designs enable a response to silent coercion without imposing unrealistic memory, attention, or procedural demands.

## References

1. Achenbach, D., Kempka, C., Löwe, B., Müller-Quade, J.: Improved coercion-resistant electronic elections through deniable re-voting. USENIX JETS (2015)
2. Adida, B.: Helios: web-based open-audit voting. In: USENIX Security Symposium, pp. 335–348 (2008)
3. Adida, B., Marneffe, O.d., Pereira, O., Quisquater, J.J.: Electing a university president using open-audit voting: analysis of real-world use of Helios. In: EVT/WOTE (2009)
4. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: pitfalls of the Fiat-Shamir heuristic and applications to Helios. In: ASIACRYPT (2012)
5. Carback, R.T., et al.: Scantegrity II municipal election at Takoma Park: the first E2E binding governmental election with ballot privacy. In: USENIX Security Symposium (2010)
6. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (1981)
7. Chaum, D.: Random-Sample Voting (2012). online
8. Chaum, D., et al.: Votexx: a solution to improper influence in voter-verifiable elections. In: E-Vote-ID (2022)
9. Chaum, D., et al.: Votexx: extreme coercion resistance. IARC ePrint 2024/1354 (2024)
10. Chaum, D., Moser, T.: ecash 2.0 inalienably private and quantum-resistant to counterfeiting. Tech. rep. (2022)
11. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: CRYPTO (1992)
12. Clark, J., Hengartner, U.: Selections: internet voting with over-the-shoulder coercion-resistance. In: Financial Cryptography (2011)

13. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: toward a secure voting system. In: IEEE Symposium on Security and Privacy, pp. 354–368 (2008)
14. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: EUROCRYPT (1997)
15. Essex, A., Clark, J., Hengartner, U.: Cobra: Toward concurrent ballot authorization for Internet voting. In: EVT/WOTE (2012)
16. Eyal, I.: On cryptocurrency wallet design. In: Tokenomics (2021)
17. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: CRYPTO, pp. 186–194 (1986)
18. Grewal, G.S., Ryan, M.D., Bursuc, S., Ryan, P.Y.A.: Caveat coercitor: coercion-evidence in electronic voting. In: IEEE Symposium on Security and Privacy (2013)
19. Haenni, R., Spycher, O.: Secure Internet voting on limited devices with anonymized DSA public keys. In: EVT/WOTE (2011)
20. Hazay, C., Lindell, Y.: Search problems. In: Efficient Secure Two-Party Protocols. ISC, pp. 227–254. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14303-8_9
21. Hirt, M.: Multi-Party Computation: Efficient Protocols, General Adversaries and Voting. Ph.D. thesis, ETH Zurich (2001)
22. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: EUROCRYPT (2000)
23. Jakobsson, M., Juels, A.: Mix and match: Secure function evaluation via ciphertexts. In: ASIACRYPT (2000)
24. Juels, A., Catalano, D., Jacobsson, M.: Coercion-Resistant electronic elections. In: ACM WPES (2005)
25. Kulyk, O., Teague, V., Volkamer, M.: Extending helios towards private eligibility verifiability. In: E-Voting and Identity (2015)
26. Lueks, W., Querejeta-Azurmendi, I., Troncoso, C.: Voteagain: A scalable coercion-resistant voting system. In: USENIX Security (2020)
27. Maurer, U.: Unifying zero-knowledge proofs of knowledge. In: AFRICACRYPT (2009)
28. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: ACM CCS (2001)
29. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: CRYPTO (1992)
30. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/nothing election scheme. In: EUROCRYPT (1993)
31. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: EUROCRYPT (1991)
32. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (1991). https://doi.org/10.1007/BF00196725
33. Wen, R., Buckland, R.: Masked ballot voting for receipt-free online elections. In: VOTE-ID (2009)
34. Zagórski, F., Carback III, R.T., Chaum, D., Clark, J., Essex, A., Vora, P.L.: Remotegrity: design and use of an end-to-end verifiable remote voting system. In: ACNS (2013)

# Threshold Receipt-Free Voting with Server-Side Vote Validation

Thi Van Thao Doan(✉), Olivier Pereira, and Thomas Peters

UCLouvain – ICTEAM – Crypto Group, 1348 Louvain-la-Neuve, Belgium
{thi.doan,olivier.pereira,thomas.peters}@uclouvain.be

**Abstract.** Proving the validity of ballots is a central element of verifiable elections. Such proofs can however create challenges when one desires to make a protocol receipt-free.

We explore the challenges raised by validity proofs in the context of protocols where threshold receipt-freeness is obtained by secret sharing an encryption of a vote between multiple authorities. In such contexts, previous solutions verified the validity of votes by decrypting them after passing them through a mix-net. This approach however creates subtle privacy risks, especially when invalid votes leak structural patterns that threaten receipt-freeness.

We propose a different approach of threshold receipt-free voting in which authorities re-randomize ballot shares then jointly compute a ZK proof of ballot validity before letting the ballots enter a (possibly homomorphic) tallying phase. Our approach keeps the voter computational costs limited while offering verifiability and improving the ballot privacy of previous solutions.

We present two protocols that enable a group of servers to verify and publicly prove that encrypted votes satisfy some validity properties: MiniMix, which preserves prior voter-side behavior with minimal overhead, and HomoRand, which requires voters to submit auxiliary data to facilitate validation over large vote domains. We show how to use our two protocols within a threshold receipt-free voting framework. We provide formal security proofs and efficiency analyses to illustrate trade-offs in our designs.

## 1 Introduction

Receipt-freeness (RF), is a central property of voting systems that ensures that voters cannot prove how they voted to a third party. Since its formal introduction by Benaloh and Tuinstra [2], RF has been the subject of extensive research, especially because of the tension that it creates with the transparency and verifiability of the election process.

A standard approach first explored by Hirt [13] consists in asking voters to encrypt their vote and submit the ciphertext to a ballot processing server that privately re-randomizes the ciphertext before posting it on a public bulletin-board. Since the re-randomization removes the voter's knowledge of the encryption randomness, voters cannot offer any evidence of the ciphertext content to


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 55–72, 2026.
https://doi.org/10.1007/978-3-032-05036-6_4

any third party. In Hirt's solution, the ballot processing server must also send a designated verifier proof that it re-randomized the ballot without modifying its content. Subsequent works have aimed to eliminate the need for sending such a proof while preserving RF using various types of randomizable signatures [3,4,6–8]. However, in all these works, receipt-freeness depends on a single trusted ballot processing server: if the ballot processing server leaks the randomness used to re-randomize the ballot, RF is lost (though privacy and verifiability would still be preserved).

In order to remove this single point of failure, Doan et al. [9] introduced a threshold receipt-free voting framework, leveraging multiple ballot processing servers (randomizers) to decentralize trust. Their system ensures RF and correctness as long as the number of corrupted randomizers remains below a threshold. Moreover, it maintains universal verifiability via mixnets.

However, Doan's protocol does not include any ballot validity proof in the ballot submission process: the validity of the votes is only verified when the mixed ciphertexts are decrypted. As a result, a coercer or a vote buyer could ask to see a very unusual invalid vote among the decrypted ballots (it could simply ask to encrypt a large randomly chosen value). This strategy, akin to the Italian attack [2], provides a limited form of receipt, in the sense that the voter can only demonstrate that he submitted an invalid vote, but it would still be desirable to avoid it whenever possible.

Worst situations may happen when the set of valid votes is very large. For instance, if a vote is by approval among 100 candidates, a voter could simply demonstrate how he voted by submitting a highly unusual approval pattern, which will come uniquely in the tally. This limitation highlights the need for alternative solutions that preserve the security requirements of threshold receipt-freeness while avoiding direct decryption of all individual votes, whether they are valid or not.

Tallying processes based on the homomorphic aggregation of ballots instead of a mixnet can offer a solution to these problems in the context of approval voting (and in other cases), while tally-hiding protocols offer more general solutions: in these protocols, individual votes are never decrypted.

**Contributions.** We propose a new paradigm for threshold receipt-free voting that addresses a core limitation in [9]: the lack of vote validity enforcement before tallying. Instead of requiring voters to generate complex zero-knowledge proofs at ballot submission time, we defer validity checking to a *pre-tally phase*, executed after combined ciphertexts are reconstructed but before they are tallied.

In particular, each voter secret-shares their vote and encrypts the shares under a traceable RF encryption scheme ($\mathsf{TREnc}$) [6], then submits the resulting ciphertexts (ballot shares) to a set of processing servers. These servers rerandomize the ballot shares and post them to a public bulletin board. After voting concludes, any party (e.g., the talliers) reconstructs each final ciphertext by combining a threshold of rerandomized ballot shares. These ciphertexts are then validated in a *pre-tally phase* via a multi-party computation (MPC) protocol that checks whether each encrypts a vote in the prescribed domain, without

revealing the vote or auxiliary information. Valid ballots are then forwarded to a standard tallying procedure. This paradigm shift decouples vote-domain enforcement from the voter's submission step and moves it to a joint server-side protocol, overcoming limitations of threshold RF settings, where secret sharing and rerandomization rendered traditional ZK-based validity checks infeasible. We realize this paradigm through two constructions:

- MiniMix: mirrors the architecture of Doan et al. [9], with voters submitting encrypted shares of their vote under TREnc, and performs MPC-based pre-tally validation over the reconstructed ciphertext.
- HomoRand: allows voters to submit auxiliary information that facilitates more efficient validity checks in certain settings, while preserving RF.

These constructions extend the threshold RF model of [9] beyond mixnet-based designs to support homomorphic tallying over binary and general vote domains. HomoRand achieves asymptotically superior pre-tally performance (logarithmic in the bit-length required to represent the domain span), but with greater voter-side computation; MiniMix, by contrast, is optimized for low client overhead, making it well-suited to moderate domain sizes or when client efficiency is critical.

Unlike general-purpose frameworks for collaborative zero-knowledge, such as that of Ozdemir and Boneh [15], which adapt ZK-SNARKs to distributed-witness settings, our protocols are tailored to voting-specific constraints. Here, the witness originates from a potentially malicious voter, and correctness and privacy must be enforced jointly, without compromising RF. Notably, the RF definition we target does not cover scenarios where an adversary prevents voting, such as forcing abstention or invalid ballots.

*Remark.* Although our protocols are motivated by [9], our pre-tally validation technique is broadly applicable. For instance, one could remove the 0/1 proofs from the CGS97 protocol [5] and insert a round of MiniMix to enforce ballot validity. Such a substitution would functionally ensure well-formed ballots, shifting the computational efforts from voting clients to servers.

## 2 Building Blocks

**Secret Sharing.** We use Shamir's $t$-out-of-$n$ threshold secret sharing scheme [16], consisting of two algorithms. The sharing algorithm $\mathsf{Share}(n, t, m)$ splits a secret $m$ into $n$ shares $(m_1, \ldots, m_n)$ such that any subset of at least $t$ shares can reconstruct $m$, while any set of fewer than $t$ shares reveals no information about $m$. The reconstruction algorithm $\mathsf{Combine}(n, t, m_1, \ldots, m_t)$ uses Lagrange interpolation to recover $m$ from any $t$ shares.

**Traceable Receipt-Free Encryption (TREnc).** TREnc [6] is a public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, augmented with a 5-tuple of algorithms: LGen, on input a security parameter $\lambda$ and a public encryption key pk, outputs a link key lk; LEnc encrypts a message $m$ using (pk, lk) and outputs a

ciphertext CT. Trace outputs the trace $\tau$ of CT. Rand randomizes CT to output another ciphertext. Ver checks if a ciphertext is valid and outputs 1 if true, and 0 otherwise.

Informally, a TREnc scheme is *traceable* if no efficient adversary can produce a second ciphertext with the same trace as a given ciphertext, yet which decrypts to a different message. It is *TCCA-secure* (Traceable Chosen-Ciphertext Attack secure) if re-randomized ciphertexts are indistinguishable from fresh ciphertexts with the same trace, even in the presence of a restricted decryption oracle. We assume it is straightforward to extract specific parts of the TREnc's ciphertext CT. In particular:

**Strip(pk, CT):** Returns the CPA-encryption component of CT. For instance, in TREnc [6], this component is $\boldsymbol{c} = (c_0, c_1, c_2) = (G^m f^\theta, g^\theta, h^\theta)$, where $m$ is the message, $\theta \leftarrow\$ \mathbb{Z}_p$, and $(G, f, g, h) \in \mathsf{pk}$.

**CPA(pk, $m$; $r$):** Computes the CPA component of a TREnc ciphertext on message $m$ with randomness $r$ under TREnc's public key pk. We note that this algorithm can be used independently of TREnc.

We further assume that the underlying CPA encryption is additively homomorphic over the message space $\mathbb{Z}_p$ and that the encryption operates over a cyclic group of prime order $p$. That is, given $\boldsymbol{C}_1 = \mathsf{CPA}(\mathsf{pk}, m_1; r_1)$ and $\boldsymbol{C}_2 = \mathsf{CPA}(\mathsf{pk}, m_2; r_2)$, one can compute $\boldsymbol{C}_1 \cdot \boldsymbol{C}_2 = \mathsf{CPA}(\mathsf{pk}, m_1 + m_2; r_1 + r_2)$ (up to group notation). In many cases, we omit the randomness when it is sampled uniformly at random. In the rest of the paper, we use the prefix TREnc. to indicate that an algorithm belongs to the TREnc suite.

**Non-interactive Proofs.** Informally, a proof for an NP relation $R$ is a protocol by which a prover $P$ convinces a probabilistic polynomial-time (PPT) verifier $V$ that $\exists w : R(w, x) = 1$, where $x$ is a called a statement, and $w$ is a witness for $x$. In the non-interactive setting, the proof consists of a single message from $P$ to $V$ and proceeds as follows. A setup algorithm $\mathsf{PrfSetup}(1^\lambda)$, on input a security parameter $\lambda$, generates a common reference string (CRS) $\sigma$. The prover then computes $\mathsf{PrfProve}(\sigma, x; w)$, outputting a proof $\pi$ if $R(x, w) = 1$, or $\perp$ otherwise. Finally, the verifier checks the proof via $\mathsf{PrfVerify}(\sigma, x, \pi)$ and outputs 1 if the proof is valid, and 0 otherwise.

A non-interactive zero-knowledge proof of knowledge (NIZKPoK) satisfies completeness, knowledge soundness, and zero-knowledge [11].

## 3 Verifiable Ciphertext Validity in MPC

We present two MPC-based constructions for verifying whether a ciphertext $\boldsymbol{C} = \mathsf{CPA}(\mathsf{PK}, m)$, under a CPA-secure encryption scheme with the corresponding decryption $\mathsf{Dec}(\mathsf{SK}, \boldsymbol{C}) = m$, encrypts a message $m$ satisfying a given constraint (e.g., $m \in \{v_1, \ldots, v_d\} \subset \mathbb{Z}_p$). The protocol reveals only the outcome of the check and leaks no additional information about the underlying plaintext or any related value, even in the case of failure. In the following, we describe two such approaches in detail.

### 3.1 MPC-MiniMix Protocol

The first construction, referred to as MPC-MiniMix (Algorithm 1), employs an OR-proof mechanism to verify message validity. The core idea is to check whether any of the ciphertexts $\boldsymbol{C}_j = \mathsf{CPA}(\mathsf{PK}, m - v_j)$ encrypts the value 0, where each $\boldsymbol{C}_j$ is derived as $\boldsymbol{C} \cdot \mathsf{CPA}(\mathsf{PK}, v_j)^{-1}$ for $j \in [d]$, exploiting the additive homomorphism of the CPA scheme. Conceptually, this construction generalizes plaintext equivalence proofs [14] to enable a form of privacy-preserving range verification.

The sender first encrypts the message $m$, from which all parties can compute the ciphertexts $\{\boldsymbol{C}_j\}_{j\in[d]}$. The parties then engage in a collaborative verification process, structured as a simplified mixnet operating over the $d$ ciphertexts. Each party $\mathcal{T}_k$, *in turn*, shuffles the ciphertexts (lines 4–9), with the output of $\mathcal{T}_k$ serving as the input to $\mathcal{T}_{k+1}$, for all $k \in [T-1]$, where $T$ denotes the number of parties. Each shuffle includes a re-randomization step (line 7), ensuring that—under honest execution—the ciphertext encrypting 0 becomes unlinkable to its initial position, thereby preserving zero-knowledge.

A significant challenge arises when the message $m$ is a carefully crafted value. In such cases, decryption may reveal not only the success or failure of the range verification but also partial information about $m$, specifically $m - v_j$ for some $j$. This could potentially assist the adversary in extracting structural patterns and re-identify the voter. To mitigate this risk, we incorporate a masking step following the shuffle: after shuffling, each party $\mathcal{T}_k$ raises each ciphertext to a fresh random exponent $\alpha_j^{(k)}$ and proves correctness via a ZKPoK $\boldsymbol{\pi}^{(k)}$ (lines 10–14). These ciphertexts and proofs are then broadcast (line 15). The shuffle proof can be instantiated using any standard zero-knowledge proof system for shuffle correctness (e.g., [1,17]), while the re-randomization proof can be implemented via a standard $\Sigma$-protocol.

We assume an honest majority setting. If the majority finds that a proof fails verification (line 16), the corresponding party $\mathcal{T}_k$ is excluded from the protocol, and the output of the last honest party is used to proceed. Although the shuffling and exponentiation phases are presented separately for clarity and to support later comparative analysis (Sect. 5.4), they can be efficiently merged into a single phase with a unified zero-knowledge proof. Finally, the parties jointly decrypt the resulting ciphertexts (line 20). If the message $m$ is valid, at least one ciphertext will decrypt to 0 while being different of $\mathsf{CPA}(\mathsf{PK}, 0; 0)$; otherwise all decrypted values appear as random group elements, preventing any information leakage about $m$.

### 3.2 MPC-HomoRand Protocol

In this approach, we leverage pairings to ensure that the encrypted message $m$ lies within a prescribed integer interval $[v_1, v_d] \subset \mathbb{Z}_p$. Without loss of generality, let $l$ be the smallest number of bits such that the span of the domain satisfies $v_d - v_1 < 2^l$. To prove that $m \in [v_1, v_d]$ (inclusive), it suffices to verify that $m - v_1 \in [0, 2^l - 1]$ and $v_d - m \in [0, 2^l - 1]$. For simplicity, here we demonstrate how to prove that $m \in [0, 2^l - 1]$. This is accomplished by having the sender decompose $m$ into an $l$-bit string $b_1 b_2 \ldots b_l$, and separately encrypt each bit in

**Algorithm 1.** MPC-MiniMix: Sequential Multi-Party Randomization & Verification

**procedure** MPC-MiniMix($\mathsf{PK}, \mathsf{SK}, \boldsymbol{C} = \{\boldsymbol{C}_j\}_{j\in[d]}$)
  Initialize $\boldsymbol{C}^{(0)} \leftarrow \boldsymbol{C}$ ▷ i.e., $\boldsymbol{C}_j^{(0)} \leftarrow \boldsymbol{C}_j$ for $j \in [d]$
  **for** $k = 1$ to $T$ **do** ▷ Each party acts sequentially
    Sample at random a permutation $P_k$ of $\{1, \ldots, d\}$
    **for** $j = 1$ to $d$ **do** ▷ Shuffle the ciphertexts
      $s_j^{(k)} \leftarrow\$ \mathbb{Z}_p \setminus \{0\}$
      $\boldsymbol{C}'^{(k)}_{P_k(j)} \leftarrow \boldsymbol{C}_j^{(k-1)} \cdot \mathsf{CPA}(\mathsf{PK}, 1_{\mathbb{G}}; s_j^{(k)})$
    **end for**
    $\boldsymbol{\pi}_{sf}^{(k)} \leftarrow \mathsf{PrfProve}(\mathsf{PK}, \{\boldsymbol{C}'^{(k)}_j, \boldsymbol{C}_j^{(k-1)}\}_{j\in[d]}; P_k, \{s_j^{(k)}\}_{j\in[d]})$
    **for** $j = 1$ to $d$ **do** ▷ Apply random factors
      Sample random values $\alpha_j^{(k)}, r_j^{(k)} \leftarrow\$ \mathbb{Z}_p \setminus \{0\}$
      $\boldsymbol{C}_j^{(k)} \leftarrow (\boldsymbol{C}'^{(k)}_j)^{\alpha_j^{(k)}} \cdot \mathsf{CPA}(\mathsf{PK}, 1_{\mathbb{G}}; r_j^{(k)})$
      $\boldsymbol{\pi}_{rd,j}^{(k)} \leftarrow \mathsf{PrfProve}(\mathsf{PK}, \boldsymbol{C}'^{(k)}_j, \boldsymbol{C}_j^{(k)}; \alpha_j^{(k)}, r_j^{(k)})$
    **end for**
    Publish $\boldsymbol{C}^{(k)} = \{\boldsymbol{C}_j^{(k)}\}_{j\in[d]}$ and $\boldsymbol{\pi}^{(k)} = (\boldsymbol{\pi}_{sf}^{(k)}, \{\boldsymbol{\pi}_{rd,j}^{(k)}\}_{j\in[d]})$
    **if** $\mathsf{PrfVer}(\boldsymbol{\pi}^{(k)}) = 0$ **then return** $\perp$
    **end if**
  **end for**
  **for** $j = 1$ to $d$ **do**
    $m_j \leftarrow \mathsf{Dec}(\mathsf{SK}, C_j^{(T)})$ ▷ Decrypt the final ciphertexts
    **if** $m_j = 0$ **then return** 1 ▷ Return 1 if any decrypted value is 0
    **end if**
  **end for**
  **return** 0 ▷ Return 0 if no ciphertext decrypts to 0
**end procedure**

two distinct source groups $\mathbb{G}$ and $\hat{\mathbb{G}}$. Subsequently, a MPC protocol operates in the target group $\mathbb{G}_{\mathcal{T}}$ to jointly verify that each encrypted bit $b_j$ satisfies $b_j \in \{0, 1\}$ for all $j \in [l]$. This binary encoding allows the sender's computational effort to scale logarithmically with the bit length $l$, i.e., $O(\log l)$.

More precisely, the sender encodes each bit $b_j \in \{0, 1\}$ by encrypting $G^{b_j}$ and $\hat{G}^{b_j}$ under public keys $\mathsf{PK}_1$ and $\mathsf{PK}_2$, respectively, yielding ciphertexts $\boldsymbol{c}_j = \mathsf{CPA}(\mathsf{PK}_1, G^{b_j}) \in \mathbb{G}$ and $\hat{\boldsymbol{c}}_j = \mathsf{CPA}(\mathsf{PK}_2, \hat{G}^{b_j}) \in \hat{\mathbb{G}}$. The parties then jointly evaluate the expression $e(G, \hat{G})^{\sum_{j\in[l]} \gamma_j b_j (1-b_j)} \stackrel{?}{=} 1_{\mathbb{G}_{\mathcal{T}}}$, where $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_{\mathcal{T}}$ denotes a bilinear pairing, and each $\gamma_j \leftarrow\$ \mathbb{Z}_p$ is a blinding factor. This check succeeds if and only if all bits are well-formed, i.e., $b_j \in \{0, 1\}$ for every $j \in [l]$. Crucially, the use of masking randomness ensures that the check does not reveal the index or value of any invalid bit, thereby preserving privacy even in the case of malformed inputs. In the MPC setting, the blinding factors $\gamma_j$ are additively shared across parties: each party $\mathcal{T}_k$ for $k \in [T]$ locally samples $\gamma_{k,j} \leftarrow\$ \mathbb{Z}_p$, and $\gamma_j = \sum_{k\in[T]} \gamma_{k,j}$. These values are generated *in parallel* across all parties.

We instantiate this technique in the MPC-HomoRand protocol, specified in Algorithms 2 and 3, using a concrete CPA encryption scheme. Let the $\mathsf{PK} = (G, f, g, h) \in \mathbb{G}^4$, with secret key $\mathsf{SK} = (\alpha, \beta) \in \mathbb{Z}_p^2$ and $f = g^\alpha h^\beta$. Encryp-

**Algorithm 2.** MPC-HomoRand.Part1: Parallel Randomization by each $\mathcal{T}_k$ ($k \in [T]$)

**procedure** MPC-HomoRand.Part1($\mathsf{PK}_1, \mathsf{PK}_2, \boldsymbol{C}$) ▷ Input: $\boldsymbol{C} = \{(\boldsymbol{c}_j, \hat{\boldsymbol{c}}_j)\}_{j\in[l]}$
  **for** $j = 1$ to $l$ **do** ▷ For each vote bit
    $\gamma_{k,j}, \lambda_{k,j}, r_{k,j}, s_{k,j} \leftarrow\$ \mathbb{Z}_p \setminus \{0\}$
    $\boldsymbol{c}''_{k,j} \leftarrow \boldsymbol{c}_j^{\gamma_{k,j}} \cdot \mathsf{CPA}(\mathsf{PK}_1, G^{\lambda_{k,j}}; r_{k,j})$ ▷ Apply random factors
    $\hat{\boldsymbol{c}}''_{k,j} \leftarrow (\mathsf{CPA}(\mathsf{PK}_2, \hat{G}; 0)/\hat{\boldsymbol{c}}_j)^{\lambda_{k,j}} \cdot \mathsf{CPA}(\mathsf{PK}_2, 1_{\hat{\mathbb{G}}}; s_{k,j})$ ▷ Apply random factors
    $\boldsymbol{\pi}_{k,j} \leftarrow \mathsf{PrfProve}(\mathsf{PK}_1, \mathsf{PK}_2, \boldsymbol{C}, \boldsymbol{c}''_{k,j}, \hat{\boldsymbol{c}}''_{k,j}; \gamma_{k,j}, \lambda_{k,j}, r_{k,j}, s_{k,j})$
    $\boldsymbol{C}''_{k,j} \leftarrow (\boldsymbol{c}''_{k,j}, \hat{\boldsymbol{c}}''_{k,j}, \boldsymbol{\pi}_{k,j})$
  **end for**
  **return** $\boldsymbol{C}''_k = \{\boldsymbol{C}''_{k,j}\}_{j\in[l]}$
**end procedure**

**Algorithm 3.** MPC-HomoRand.Part2: Multi-Party Verification

1: **procedure** MPC-HomoRand.Part2($\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{SK}_1, \mathsf{SK}_2, \boldsymbol{C}, \{\boldsymbol{C}''_k\}_{k\in[T]}$)
2:  **for** $j = 1$ to $l$ **do** ▷ For each vote bit
3:  **for** $k = 1$ to $T$ **do**
4:  **if** $\mathsf{PrfVer}(\mathsf{PK}, \boldsymbol{C}''_{k,j}, \boldsymbol{\pi}_{k,j}) = 0$ **then return** $\perp$
5:  **end if**
6:  **end for**
7:  $\boldsymbol{c}''_j = (c''_{0j}, c''_{1j}, c''_{2j}) \leftarrow \prod_{k=1}^{T} \boldsymbol{c}''_{k,j}$ ▷ Aggregate the ciphertexts
8:  $\hat{\boldsymbol{c}}''_j = (\hat{c}''_{0j}, \hat{c}''_{1j}, \hat{c}''_{2j}) \leftarrow \prod_{k=1}^{T} \hat{\boldsymbol{c}}''_{k,j}$
9:  $X_j \leftarrow \mathsf{Dec}(\mathsf{SK}_1, \boldsymbol{c}''_j)$
10:  $\bar{\boldsymbol{c}}'_j = (\bar{c}'_{0j}, \bar{c}'_{1j}, \bar{c}'_{2j}) \leftarrow \mathsf{CPA}(\mathsf{PK}_2, \hat{G}; 0)/\hat{\boldsymbol{c}}_j$
11:  $x_j, y_j, z_j \leftarrow e(X_j, \bar{c}'_{0j})/e(G, \hat{c}''_{0j}), e(X_j, \bar{c}'_{1j})/e(G, \hat{c}''_{1j}), e(X_j, \bar{c}'_{2j})/e(G, \hat{c}''_{2j})$
12:  **end for**
13:  $(x, y, z) \leftarrow (\prod_{j=1}^{l} x_j, \prod_{j=1}^{l} y_j, \prod_{j=1}^{l} z_j)$
14:  $Y \leftarrow \mathsf{Dec}(\mathsf{SK}_2, (x, y, z))$ ▷ Decrypt the aggregated result
15:  **return** $Y = 1_{\mathbb{G}_T}$ ? 1 : 0
16: **end procedure**

tion of a message $m \in \mathbb{Z}_p$ with randomness $r$ proceeds as $\boldsymbol{c} = \mathsf{CPA}(\mathsf{PK}, m; r) = (c_0, c_1, c_2) = (G^m f^r, g^r, h^r)$, and decryption yields $\mathsf{Dec}(\mathsf{SK}, \boldsymbol{c}) = c_0 \cdot c_1^{-\alpha} \cdot c_2^{-\beta} = G^m$. In Algorithm 2, each party independently masks the ciphertexts in parallel and proves correctness via a ZKPoK $\{\boldsymbol{\pi}_{k,j}\}_{j\in[l]}$, which are broadcast alongside the resulting ciphertexts. Algorithm 3 specifies the subsequent verification phase: each proof is validated (line 4), and any party that fails verification is excluded. The combination of valid ciphertexts (lines 7–8) is computed solely from the outputs of honest parties, and the final values are jointly decrypted (line 14). The values $(x_j, y_j, z_j)$ computed in line 11 are used to securely mask the term $b_j(1 - b_j)$ which would otherwise leak information if $b_j \notin \{0, 1\}$. Although the description employs a specific CPA encryption scheme for concreteness, our construction generalizes to any additive homomorphic CPA scheme. The full version, which includes complete proofs and additional technical details, is available in [10]. The core idea, securely verifying bit decomposition without leakage, remains applicable in broader cryptographic contexts.

## 4 Threshold Receipt-Free Voting

We adopt the voting system model of Doan et al. [9], which supports multiple ballot processing servers and ensures both threshold receipt-freeness and threshold correctness. We then review the first threshold receipt-free scheme from [9], whose limitations motivate our new constructions in Sect. 5.

**Definition 4.1 (Voting System [9]).** *A voting system with $n$ ballot processing servers consists of algorithms:* (SetupElection, Vote, ProcessBallot, TraceBallot, Valid, Append, Publish, VerifyVote, Tally, VerifyResult), *and a result function* $\rho_m : \mathbb{V}^m \cup \{\perp\} \to \mathbb{R}$, *where* $\mathbb{V}$ *is the vote domain and* $\mathbb{R}$ *is the result space.*

The election is initialized via SetupElection. Each voter runs Vote to generate a ballot consisting of $n$ shares, one per ballot processing server. Each share is independently randomized by a distinct server using ProcessBallot. A tracking code is derived via TraceBallot, enabling voters to later trace their ballots on the public bulletin board PBB. Validated shares are collected and appended to the ballot box BB using Append, then made publicly available on PBB via Publish. Voters verify correct recording using VerifyVote and their tracking codes. The tally is computed over valid ballots, checked by Valid, using Tally, and its correctness is publicly verifiable via VerifyResult.

**Threshold Receipt-Freeness ($\mathsf{t_{rf}}$).** The definition of threshold RF proceeds in two parts. The first ensures that no adversary–despite corrupting up to $\mathsf{t_{rf}}$ out of $n$ ballot processing servers–can coerce a voter into producing a receipt that convincingly proves how they voted. This guarantee holds even if the adversary learns the voter's randomness or attempts to bias the ballot construction. The second part, extends the indistinguishability-based receipt-freeness notion of Devillez et al. [6] to the threshold setting. It formalizes the requirement that even if a voter colludes with up to $\mathsf{t_{rf}}$ servers and deviates arbitrarily from the honest ballot distribution, a third party, given full knowledge of how the ballot was constructed and access to the public bulletin board, should not be able to determine whether the voter submitted that ballot or a different one encoding another vote.

**Threshold Correctness ($\mathsf{t_{corr}}$).** Threshold correctness ensures that the choices of honest voters are faithfully reflected in the final tally, even in the presence of limited adversarial control over the ballot processing servers. Concretely, it guarantees that as long as at least $n - \mathsf{t_{corr}}$ processed shares of each honestly generated ballot appear on the public bulletin board–regardless of whether they were handled by malicious servers–the election outcome remains uniquely determined. That is, the adversary should not be able to construct two sets of valid-looking ballot boxes that lead to different outcomes.

**The Doan et al.'s Voting Scheme.** The first protocol to simultaneously achieve threshold receipt-freeness and correctness was proposed by Doan et al. [9] (E-Vote-ID 2024). Their construction, illustrated in Fig. 1, introduces a threshold receipt-free voting system that significantly reduces trust assumptions on ballot randomizers.

In their scheme, each voter secret-shares their vote, encrypts each share using a TREnc (Sect. 2), and submits the resulting ballot shares $\{\mathsf{b}_i\}_{i\in[n]}$ to a set of independent randomizers. Each randomizer then rerandomizes one ballot share and output $\mathsf{b}'_i$ made available on the public board BB. After the voting phase, the talliers (or any observer) extract standard CPA components from the TREnc outputs, and use Lagrange interpolation to reconstruct an encryption of the original vote. This process results in a final ciphertext $\boldsymbol{C}$ that is then submitted into a mixnet-based tallying process. Receipt-freeness is preserved as long as at least one honest rerandomized share $\mathsf{b}'_i$ is used in the reconstruction of $\boldsymbol{C}$.

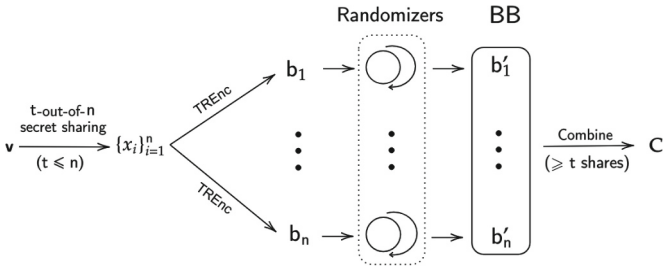


**Fig. 1.** Doan et al.'s voting scheme.

*Limitations.* The protocol does not ensure that $\boldsymbol{C}$ is a valid vote encryption, meaning a malicious $\boldsymbol{C}$ could encode an invalid vote, that, upon decryption, leaks unintended information and potentially violates violating RF. While TREnc provides strong privacy guarantees, it offers limited support when it comes to verifying election-specific constraints on the encrypted vote, which typically require non-linear proofs. For instance, proving a vote $\mathsf{v}$ is a bit requires a quadratic relation, $\mathsf{v}(1-\mathsf{v})=0$, requiring a non-linear proof (see, e.g., [8]).

Although NIZK proofs can, in principle, support such constraints, integrating them into this setting with the secret sharing and rerandomization is non-trivial. In particular, a non-linear proof constructed before randomization would not survive the transformation, and constructing such a proof after randomization is infeasible for the voter, who lacks control over the ciphertext randomness. In a multi-randomizer setting, where multiple servers independently rerandomize different shares of a ballot, adapting a non-linear proof locally is also challenging. The transformation applied by each server generally depends on the randomness chosen by others, rendering any isolated adaptation incorrect or unverifiable. As a result, constructing such proofs typically requires interaction between the voter and randomizing parties, which could introduce complexity and security risks, as malicious servers could collude with voters. To address these issues, we introduce a technique in the next section to enforce non-linear constraints on encrypted votes without requiring any voter-randomizer coordination.

## 5 Voting Schemes

We propose a new approach that avoids requiring voters to construct complex zero-knowledge proofs at ballot submission time. Instead, we shift the validity checking to a dedicated *pre-tally phase*, which takes place after the final combined ciphertexts $\boldsymbol{C}$ have been reconstructed (see Fig. 1), but before tallying begins.

At a high level, our scheme preserves the overall voting flow of Doan et al. from the voter's perspective. The key difference lies in how the reconstructed ciphertexts $\boldsymbol{C}$ are handled. Rather than passing them directly to a mixnet, they are first processed in a multi-party computation (MPC) protocol that verifies whether each ciphertext encodes a valid vote. This verification is performed collaboratively by the talliers or those who hold shares of the decryption key without revealing the vote or any auxiliary information. Only ciphertexts that pass this validation step are forwarded to the standard tallying process.

In the following, we present two concrete instantiations of this paradigm: MiniMix and HomoRand. These schemes differ in the inputs provided by voters to the pre-tally phase and the way the underlying MPC protocol operates. Before delving into the designs of MiniMix (Sect. 5.2) and HomoRand (Sect. 5.3), we first outline their shared structure in Sect. 5.1, following the general voting system definition in Definition 4.1. We then give the correctness and the security theorem statements of the constructions in Sect. 6.

### 5.1 Overview

*Key Generation.* The election authority EA initiates the setup by running the SetupElection algorithm, generating a public/secret key pair $(\mathsf{PK}, \mathsf{SK})$. The public key PK is posted on a public bulletin board PBB, while the private key SK is shared among $T$ talliers using a threshold decryption scheme. Key generation can be distributed securely in prime-order groups through standard techniques.

*Voting Phase.* To cast a vote $\mathsf{v}$, a voter executes the Vote algorithm. In principle, this procedure begins by secret-sharing $\mathsf{v}$ into $n$ shares using a $t$-out-of-$n$ threshold secret sharing scheme (Sect. 2). Each share is then independently encrypted under a TREnc (Sect. 2), producing ballot shares $\{\mathsf{b}_i\}_{i\in[n]}$. The complete ballot $\mathsf{b}$ consists of this set of encrypted shares.

Each $\mathsf{b}_i$ is then submitted to a distinct randomizing server, while the corresponding trace $\tau = \mathsf{TraceBallot}(\mathsf{b})$ is simultaneously transmitted to PBB. Upon receiving a $\mathsf{b}_i$, a server performs a local verification using the validity checks defined by TREnc, optionally including a zero-knowledge proof verification. It further ensures that no duplicate traces are present with respect to all traces on PBB. Valid ballot shares are re-randomized using $\mathsf{ProcessBallot}(\mathsf{b}_i)$ and made available on the bulletin board. Voters can later verify inclusion of their vote by querying VerifyVote on the trace $\tau$ and confirming its appearance on PBB.

*Tallying Phase.* After voting concludes, the tallying phase begins. Talliers aggregate ballot shares by comparing the traces on $\mathsf{PBB}$ with the complete traces $\tau$ posted by each voter. The $\mathsf{Valid}$ function checks whether at least $t$ valid shares are present for each ballot. If the condition is met, the $\mathsf{CPA}$ components are extracted from the corresponding shares and combined using Lagrange interpolation. Owing to the linearity of the interpolation, this process can be performed directly in the encrypted domain, yielding a final ciphertext that encrypts the original vote. Ballots failing the $\mathsf{Valid}$ check are discarded.

As noted by Doan et al. [9], this reconstruction step is critical for mitigating adversarial influence. A malicious voter may attempt to deviate from the protocol by submitting malformed or inconsistent shares. However, since all valid ballot shares on the bulletin board (possibly more than $t$) are combined, only one message can be reconstructed. As the adversary cannot anticipate which subset of shares will be successfully posted (e.g., due to the random failure of non-operational randomizers), they are incentivized to submit a consistent and correct ballot to ensure that the tally reflects their intended vote.

Finally, the reconstructed ciphertexts corresponding to valid ballots are submitted to the $\mathsf{PreTally}$ function, which checks that the encrypted vote is within the intended range. Depending on its inputs, the $\mathsf{PreTally}$ function behaves differently. In the $\mathsf{MiniMix}$ construction (Fig. 2), which utilizes the MPC-$\mathsf{MiniMix}$ protocol from Algorithm 1, the voter submits only the encrypted vote. In contrast, the $\mathsf{HomoRand}$ construction uses the MPC-$\mathsf{HomoRand}$ protocol from Algorithms 2 and 3, where the voter also sends additional values to assist the participating parties in verifying the validity of the encrypted vote. Depending on the deployment scenario, the participating parties may include the talliers or, alternatively, external authorities. For simplicity, we assume that the talliers are responsible for performing this validation in the current setting. Invalid ballots are discarded based on the outcome of the $\mathsf{PreTally}$ check. The talliers then run the $\mathsf{Tally}$ function to compute the election result $\mathsf{r}$ based on the valid ones according to the election rules. A proof of correctness $\Pi$ is generated and can be verified by anyone using the $\mathsf{VerifyResult}$ algorithm. In the instantiations below, the number of servers $n$ and the threshold $t$ are implicit inputs for all algorithms.

*Relationship Between Correctness and Receipt-Freeness Thresholds.* Our constructions rely on a secret sharing scheme where at least $t$ valid ballot shares are needed to reconstruct a vote. This implies that the system can tolerate up to $n - t$ missing or invalid shares without affecting correctness. On the privacy side, receipt-freeness holds as long as the reconstruction includes at least one honestly rerandomized share. So even if up to $t - 1$ ballot shares are maliciously processed, it is infeasible for the adversary to compromise voter privacy. In other words, the system remains secure against up to $t - 1$ compromised processing servers, which matches the tight bound shown in prior work by Doan et al. [9].

### 5.2 The MiniMix Construction

In this scheme, the input to the PreTally is a ballot $\mathsf{b} = \{\mathsf{b}_i\}_{i\in[n]}$ randomized by randomizers, where each $\mathsf{b}_i$ is a TREnc's encryption of a share $x_i$ of the voter's intended message $\mathsf{v}$.

To initiate the PreTally procedure, the talliers (and optionally the public) verify each ballot share $\mathsf{b}_i$ by applying the TREnc.Ver function (see Sect. 2). A ballot $\mathsf{b}$ is deemed valid by $\mathsf{Valid}(\mathsf{BB}, \mathsf{b})$ (see Fig. 2) if at least $t$ valid shares are posted on the public bulletin board. Upon successful validation, the combination algorithm Combine is publicly executed by any of the talliers. It takes as input the homomorphic components of all available valid ballot shares–up to $n$ shares–and outputs a combined ciphertext $\boldsymbol{C}_0$, representing the encrypted vote $\mathsf{v}$. Due to the properties of Lagrange interpolation, reconstructing with more than $t$ valid shares preserves the correctness of the resulting ciphertext.

Subsequently, $T$ talliers jointly executes the MPC-MiniMix as described in Algorithm 1 with the input $\boldsymbol{C} = \{\boldsymbol{C}_j\}_{j\in[d]}$, where $\boldsymbol{C}_j = \boldsymbol{C}_0 \cdot \mathsf{CPA}(\mathsf{PK}, -v_j)$. In particular, each tallier $\mathcal{T}_k$ for $k \in [T]$ shuffles the $d$ ciphertexts in $\boldsymbol{C}$, applying an independent random factor. It also produces a publicly verifiable ZKPoK proof, attesting to correctness. Misbehaving parties are identified through proof verification. After the last tallier has completed their respective round, all the talliers jointly decrypt the final output $\boldsymbol{C}^{(T)} = \{\boldsymbol{C}_j^{(T)}\}_{j\in[d]}$. As $\boldsymbol{C}^{(T)}$ is encrypted under the TREnc's PK, decryption proceeds using $\mathsf{TREnc.Dec}(\mathsf{SK}, \cdot)$ (Algorithm 1, line 20). The PreTally procedure outputs 1 as soon as one of the decrypted ciphertexts equals 0, confirming that the original vote $\mathsf{v}$ belongs to the designated valid range; otherwise, it outputs 0. Invalid votes are discarded, and the Tally function is applied to the combined ciphertext $\boldsymbol{C}_0$ of each valid ballot to compute the final outcome according to the election rules. A formal correctness proof of MPC-MiniMix is provided in the full version [10].

*Discussion.* It is worth noting that the MiniMix construction can be adapted to minimize online-phase computation. In the presented variant, the shuffle is performed after a ballot is available on the PBB, which may lead to inefficiencies such as increased latency and synchronization delays.

As an alternative, the authorities may jointly shuffle the encrypted vote domain $\{\boldsymbol{V}_j = \mathsf{CPA}(v_j)\}_{j\in[d]}$ under a secret permutation $\pi$, yielding a randomized sequence $\{\boldsymbol{V}'_{\pi(j)}\}_{j\in[d]}$. Given a combined ciphertext $\boldsymbol{C}_0$ (as defined earlier), they compute $\boldsymbol{C}'_j = \boldsymbol{C}_0/\boldsymbol{V}'_{\pi(j)}$ for each $j \in [d]$, resulting in ciphertexts encrypting $(\mathsf{v} - v_{\pi(j)})$. These can be tested for equality to zero to verify whether $\mathsf{v}$ belongs to the valid domain. As $\pi$ remains hidden, the test leaks no information about the actual vote. To avoid linkability, each ballot must be verified against a freshly shuffled domain encryption. Otherwise, repeated shuffles could reveal identical vote positions, enabling correlation across ballots. Hence, the number of domain shuffles must scale with the number of ballots or eligible voters.

SetupElection ($\lambda$)

$(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{TREnc.Gen}(1^\lambda)$
**return** PK

Vote(id, v[, aux])

$\{x_1, \ldots, x_n\} \leftarrow \mathsf{Share}(n, t, \mathsf{v})$
**if** aux is empty **for** $i = 1$ to $n$ do
  $\mathsf{lk}_i \leftarrow\$ \mathsf{TREnc.LGen}(\mathsf{PK})$
**else** $\{\mathsf{lk}_i\}_{i=1}^n \leftarrow$ aux
**for** $i = 1$ to $n$ do
  $\mathsf{b}_i \leftarrow \mathsf{TREnc.LEnc}(\mathsf{PK}, \mathsf{lk}_i, x_i)$
**return** $\mathsf{b} = \{\mathsf{b}_i\}_{i=1}^n$

Valid(BB, b)

**if** $\exists \mathsf{b}' \in \mathsf{BB} \wedge \exists \tau_i' \subset \mathsf{TraceBallot}(\mathsf{b}')$ :
$\tau_i' \subset \mathsf{TraceBallot}(\mathsf{b})$ **then return** $\perp$
$k = 0$
**for** $i = 1$ to $|\mathsf{b}|$ do
  **if** $\mathsf{TREnc.Ver}(\mathsf{PK}, \mathsf{b}_i) = 1$
  **then** $k \leftarrow k + 1$
**if** $k \geq t$ **then return** 1 **else return** 0

ProcessBallot($\mathsf{b}_i$)

**return** $\mathsf{TREnc.Rand}(\mathsf{PK}, \mathsf{b}_i)$

TraceBallot(b)

$\tau_i \leftarrow \mathsf{TREnc.Trace}(\mathsf{b}_i)$
**return** $\tau = \{\tau_i\}_{i=1}^n$

PreTally (BB, SK, b)

**if** $\mathsf{Valid}(\mathsf{BB}, \mathsf{b}) = 0$ **then return** 0
**for** $i = 1$ to $|\mathsf{b}|$ do
  $\boldsymbol{c}_i \leftarrow \mathsf{TREnc.Strip}(\mathsf{PK}, \mathsf{b}_i)$
$\boldsymbol{C}_0 \leftarrow \mathsf{Combine}(n, t, \{\boldsymbol{c}_i\}_{i=1}^{\leq n})$
**for** $j = 1$ to $d$ do
  $\boldsymbol{C}_j \leftarrow \boldsymbol{C}_0 \cdot \mathsf{CPA}(\mathsf{PK}, -v_j)$
**return** $\mathrm{MPC\text{-}MiniMix}(\mathsf{PK}, \mathsf{SK}, \boldsymbol{C} = \{\boldsymbol{C}_j\}_{j=1}^d)$

VerifyVote(PBB, $\tau$)

**if** $\exists \mathsf{b} \in \mathsf{PBB} : \mathsf{Valid}(\mathsf{b}) \wedge \tau == \mathsf{TraceBallot}(\mathsf{b})$
**then return** 1 **else return** 0

**Fig. 2.** MiniMix instantiation of our voting scheme.

### 5.3 The HomoRand Construction

Due to space constraints, the full specification of HomoRand is provided in [10] (Appendix A); here we give a high-level overview. The input to PreTally is a randomized ballot $\mathsf{b} = \{\mathsf{b}_i\}_{i\in[n]}$, where each $\mathsf{b}_i$ now consists of $l$ components $\mathsf{b}_{ji}$ for $j \in [l]$. To this end, the vote v is first decomposed into an $l$-bit string $\{b_j\}_{j\in[l]}$, and each bit $b_j$ is shared using $t$-out-of-$n$ secret sharing scheme to obtain $n$ share $\{b_{ji}\}_{i\in[n]}$. Each share $b_{ji}$ is then encrypted separately under two TREnc public keys, yielding a pair of ciphertexts in $\mathbb{G}$ and $\hat{\mathbb{G}}$. A non-interactive randomizable proof (e.g., Groth-Sahai proofs [12]) accompanies each pair, proving consistency across the two encryptions, i.e., demonstrating that they indeed encrypt the same value. As a result, each ballot share contains $l$ components, where each includes two TREnc ciphertexts and a consistency proof. Each ballot share is re-randomized, and the resulting share is posted to the public board.

The PreTally protocol begins by validating each ballot b posted on the public bulletin board. This involves applying TREnc.Ver to the ciphertexts and invokes PrfVerify on the associated proof. A ballot is deemed valid if, for every bit index $j \in [l]$, there exist at least $t$ valid shares $\{\mathsf{b}_{ji}\}_i$ posted on the board. Once a ballot passes verification, the associated proofs are discarded. Next, the Combine algorithm aggregates the homomorphic components of the accepted ballot shares (up to $n$) to reconstruct two CPA encryptions of $\{b_j\}_{j\in[l]}$ in $\mathbb{G}$ and $\hat{\mathbb{G}}$. The protocol then invokes the MPC-HomoRand procedure from Algorithms 2 and 3 to jointly verify that each $b_j$ is a bit.

### 5.4 Efficiency Discussion

Table 1 compares the computational costs incurred during the casting of a single ballot under the two proposed constructions. All costs reported are per voter, per randomizer, or per tallier, as appropriate.

*Computational Costs.* On the voter's side, the Vote algorithm in MiniMix requires $n$ invocations of TREnc.Enc, as each ballot share is encrypted independently. In contrast, HomoRand imposes a higher load: $2ln$ encryptions (two per bit per share) and $16ln$ exponentiations for Groth-Sahai consistency proofs across all $i \in [n]$ and $j \in [l]$ (see [10] for details on proof computation). Each randomizer, executing ProcessBallot, performs 1 TREnc rerandomization in MiniMix, while in HomoRand, this increases to $2l$, alongside rerandomizing the associated proofs. Talliers executing PreTally incur costs primarily from the underlying MPC protocols. In MPC-MiniMix (Algorithm 1), each tallier (i) shuffles $d$ ciphertexts with verifiable proofs at a cost denoted $\mathsf{shuffle}(d)$ (lines 4–9); (ii) applies random factors to shuffled ciphertexts with proofs (lines 11–13), amounting to $d$ rand operations; and (iii) participates in the joint decryption of $d$ ciphertexts (line 19–24), contributing $d$ dec operations. In MPC-HomoRand, each tallier performs $2l$ rand operations (Algorithm 2, lines 2–8) and engages in $l$+1 dec operations (Algorithm 3, lines 9 and 14).

*Verification Costs.* Anyone, including external auditors or voters, can verify the PreTally result by checking the posted proofs. In HomoRand, the verification requires $2l \cdot \mathsf{rand} + (l+1) \cdot \mathsf{dec}$ while in MiniMix it involves $d \cdot (\mathsf{rand} + \mathsf{dec})$, plus $\mathsf{shuffle}(d)$ proofs from each of the $T$ talliers. Since $l = \log_2(d)$, the relative verification cost ratio scales as roughly $d/(2\log_2 d)$ in favor of HomoRand for large domains $d$, though MiniMix incurs additional overhead from shuffle proofs, which scale with both $d$ and $T$.

While HomoRand achieves PreTally costs that scale with $l = \log_2(d)$, asymptotically outperforming the linear-in-$d$ cost of MiniMix, this efficiency comes at the price of substantially higher computational overhead for voters and randomizers. Specifically, the cost of Vote in HomoRand scales with both $n$ and $l$, due to bitwise encryption and the generation of Groth-Sahai proofs. In contrast, MiniMix imposes minimal and domain-independent client-side costs, making it an attractive choice in settings where lightweight voting is essential and the number of valid vote options is modest. Conversely, HomoRand may be preferable in settings where the verification of PreTally is a critical concern–such as public audits or large-scale elections–particularly when voter-side computation is not

**Table 1.** Computational cost per ballot under each construction.

| | SetupElection | Vote | ProcessBallot | PreTally |
|---|---|---|---|---|
| MiniMix | $1 \cdot \mathsf{TREnc.Gen} \in \mathbb{G}$ | $n \cdot \mathsf{TREnc.Enc}$ | $1 \cdot \mathsf{TREnc.Rand}$ | $\mathsf{shuffle}(d) + d \cdot \mathsf{rand} + d \cdot \mathsf{dec}$ |
| HomoRand | $1 \cdot \mathsf{TREnc.Gen} \in \mathbb{G}$<br>$1 \cdot \mathsf{TREnc.Gen} \in \hat{\mathbb{G}}$ | $2ln \cdot \mathsf{TREnc.Enc}$<br>$\mathbb{G}^{16ln} \times \hat{\mathbb{G}}^{16ln}$ | $2l \cdot \mathsf{TREnc.Rand}$<br>$\mathbb{G}^{14l} \times \hat{\mathbb{G}}^{14l}$ | $2l \cdot \mathsf{rand} + (l+1) \cdot \mathsf{dec}$ |

a bottleneck and the domain size $d$ is large enough to outweigh its setup and voting costs.

## 6 Security of the Voting Schemes

The proposed voting schemes achieve both *threshold receipt-freeness* and *threshold correctness* (Sect. 4). Due to space constraints, we provide proof sketches here. Formal proofs appear in the full version [10].

**Theorem 6.1 (Threshold Receipt-Freeness).** *Let* TREnc *be verifiable and* TCCA*-secure, and let the proof systems employed for the pre-tally and for verifying tally correctness be zero-knowledge. Then both constructions achieve threshold receipt-freeness under a $t$-out-of-$n$ sharing scheme with threshold $\mathsf{t_{rf}} = t-1$. More precisely, for the* MiniMix *scheme,* $\Pr[\mathsf{Exp}^{\mathsf{deceive}}_{\mathcal{A},\mathcal{V},\mathsf{t_{rf}}}(\lambda) = 1] \leq \varepsilon_{\mathrm{verif}}$ *and* $\mathsf{Adv}^{\mathsf{rf},\mathsf{t_{rf}},\beta}_{\mathcal{A},\mathcal{V}}(1^\lambda) \leq \varepsilon_{\mathrm{ZK}} + q(n-\mathsf{t_{rf}})\varepsilon_{\mathrm{tcca}}$. *For the* HomoRand *scheme,* $\Pr[\mathsf{Exp}^{\mathsf{deceive}}_{\mathcal{A},\mathcal{V},\mathsf{t_{rf}}}(\lambda) = 1] \leq \varepsilon_{\mathrm{verif}}$ *and* $\mathsf{Adv}^{\mathsf{rf},\mathsf{t_{rf}},\beta}_{\mathcal{A},\mathcal{V}}(1^\lambda) \leq \varepsilon_{\mathrm{ZK}} + lq(n-\mathsf{t_{rf}})(2\varepsilon_{\mathrm{tcca}} + \varepsilon_{\mathsf{sxdh}})$. *Here, $l$ is the bit-length of the vote domain, and $\varepsilon_{\mathrm{ZK}}$, $\varepsilon_{\mathrm{sxdh}}$, $\varepsilon_{\mathrm{verif}}$, $\varepsilon_{\mathrm{tcca}}$ bound the adversarial advantage against the ZK proof systems, the SXDH (Symmetric eXternal Diffie-Hellman) assumption, verifiability, and* TCCA*-security of* TREnc*, respectively; $q$ is the number of ballot-append queries.*

*Proof.* We sketch the proof for MiniMix; the case of HomoRand is analogous. Threshold receipt-freeness is defined via two experiments.

In the first, $\mathsf{Exp}^{\mathsf{deceive}}_{\mathcal{A},\mathcal{V},\mathsf{t_{rf}}}(\lambda)$ [9], security follows from TREnc's TCCA security, verifiability, and the correctness of the secret sharing scheme, as shown in [9].

In the second, $\mathsf{Exp}^{\mathsf{rf},\mathsf{t_{rf}},\beta}_{\mathcal{A},\mathcal{V}}(\lambda)$, the adversary must provide two valid ballots with matching traces. We define a sequence of hybrid games starting from $\beta = 0$ (honest setup) and ending at $\beta = 1$ (simulated setup). Let $\mathcal{A}$ query ballot pairs $\mathsf{B}_0, \mathsf{B}_1$ with identical traces to $\mathsf{BB}_0$ and $\mathsf{BB}_1$ respectively. We progressively replace the processed ballot shares of $\mathsf{B}_0$ with those of $\mathsf{B}_1$ in $\mathsf{BB}_0$, relying on the TCCA security of TREnc to preserve indistinguishability at each step, incurring at most $\varepsilon_{\mathrm{tcca}}$ advantage per swap. This may introduce inconsistencies in the tally since this changes the underlying plaintext. However, since the vote intent of $\mathsf{B}_0$ is known from TREnc.Dec's correctness, the zero-knowledge simulator can emulate the decryption step (Algorithm 1, line 20) to match the expected result of pre-tally phase. Thus, $\mathcal{A}$ cannot distinguish whether $\mathsf{B}_0$ or $\mathsf{B}_1$ was processed, even when $\mathsf{B}_0$ encodes an invalid vote, except with an error bounded by $\varepsilon_{\mathrm{ZK}}$. A hybrid argument over $q$ queries yields $\mathsf{Adv}^{\mathsf{rf},\mathsf{t_{rf}},\beta}_{\mathcal{A},\mathcal{V}}(1^\lambda) \leq \varepsilon_{\mathrm{ZK}} + q(n - \mathsf{t_{rf}})\varepsilon_{\mathrm{tcca}}$.

**Theorem 6.2 (Threshold Correctness).** *Let* TREnc *be traceable and verifiable, and assume that the underlying NIZK proof systems are correct. Then both constructions achieve threshold correctness with $\mathsf{t_{corr}} = n - t$ under a $t$-out-of-$n$ secret sharing scheme. More precisely, for any efficient adversary $\mathcal{A}$ making $q$ ballot-append queries,* $\Pr[\mathsf{Exp}^{\mathsf{corr},\mathsf{t_{corr}}}_{\mathcal{A},\mathcal{V}}(\lambda) = 1] \leq qn\varepsilon_{\mathrm{trace}} + \varepsilon_{\mathrm{corr}}$ *for* MiniMix *and* $\Pr[\mathsf{Exp}^{\mathsf{corr},\mathsf{t_{corr}}}_{\mathcal{A},\mathcal{V}}(\lambda) = 1] \leq 2qln\varepsilon_{\mathrm{trace}} + \varepsilon_{\mathrm{corr}}$ *for* HomoRand*, where $l$ is the vote bit-length, and $\varepsilon_{\mathrm{trace}}$, $\varepsilon_{\mathrm{corr}}$ bound $\mathcal{A}$'s advantage against the traceability of* TREnc *and the correctness error of the underlying MPC protocol, respectively.*

*Proof.* In the threshold correctness experiment, denoted $\mathsf{Exp}^{\mathsf{corr},\mathsf{t}_{\mathsf{corr}}}_{\mathcal{A},\mathcal{V}}(\lambda)$ [9], $\mathcal{A}$ submits $q$ pairs of valid ballots to two election views, each omitting at most $\mathsf{t}_{\mathsf{corr}}$ ballot shares. As all ballots derive from honestly generated ones encoded as TREnc ciphertexts, traceability ensures $\mathcal{A}$ cannot alter the vote intent without changing trace values, except with negligible probability. This yields a bound of $qn\varepsilon_{\text{trace}}$ for MiniMix, and $2qln\varepsilon_{\text{trace}}$ for HomoRand. Once ballots are posted to the PBB, the correctness of the secret sharing scheme guarantees that the inputs to the pre-tally phase correctly reflect the original vote intent. The correctness of the underlying NIZK proofs in MPC-MiniMix and MPC-HomoRand then ensures that both views yield identical outputs for valid votes.

**Verifiability.** Our schemes ensure both *individual* and *universal verifiability*. Since the voter's voting process in MiniMix follows the protocol of [9], it inherits individual verifiability via the traceability of TREnc. In HomoRand, voters additionally provide consistency proofs across the two TREnc ciphertexts in each ballot share, without affecting traceability. Hence, no adversary can alter the vote intent without detection. For universal verifiability, both constructions employ publicly verifiable ZKPoKs in pre-tally and tally phases. The soundness of these proofs ensures that all accepted ballots encode valid votes and that decryption is performed correctly, thereby guaranteeing a trustworthy tally outcome.

## 7 Conclusion

We propose a new direction for validating encrypted ballots in threshold receipt-free voting systems, where ballots are non-interactively rerandomized by multiple independent ballot processing servers. Our approach introduces a pre-tally validation phase executed in a multiparty computation style, allowing authorities to verify the validity of encrypted votes without decrypting them or requiring voter-supplied validity proofs.

We develop two constructions that achieve threshold receipt-freeness and verifiability while addressing privacy risks present in prior mixnet-based systems. Both schemes support homomorphic or mixnet-based tallying, depending on vote range constraints, and crucially reveal only the validity status of a ballot, nothing more. To our knowledge, this is the first threshold receipt-free solution to achieve better privacy independently of the tallying technique.

Our efficiency analysis recommends using MiniMix when voter-side efficiency is critical and the number of valid vote options is small or moderate, while HomoRand is better suited for large or complex vote domains. In addition to our core designs, we also discuss how existing validity-check techniques can be adapted to fit our pre-tally framework. An open direction is to reduce the computational and communication complexity of the pre-tally process, or to devise alternative mechanisms such as randomizable validity proofs that can be verified in a single-pass setting, even in the presence of fully malicious randomizers.

**Acknowledgments.** Thomas Peters is a research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in part by the Walloon Region through the project CyberExcellence (convention number 2110186).

## References

1. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_17
2. Benaloh, J., Tuinstra, D.: Receipt-free secret ballot elections. In: Proceedings of the 26th ACM Symposium on Theory of Computing (STOC), pp. 544–553. ACM (1994)
3. Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on randomizable ciphertexts. In: Public Key Cryptography–PKC, pp. 403–422. Springer (2011)
4. Chaidos, P., Cortier, V., Fuchsbauer, G., Galindo, D.: BeleniosRF: a non-interactive receipt-free electronic voting scheme. In: Proceedings of ACM CCS, pp. 1614–1625. ACM (2016)
5. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: EUROCRYPT '97, pp. 103–118. Springer (1997)
6. Devillez, H., Pereira, O., Peters, T.: Traceable receipt-free encryption. In: Proceedings ASIACRYPT 2022, pp. 273–303. Springer (2022)
7. Devillez, H., Pereira, O., Peters, T., Yang, Q.: Can we cast a ballot as intended and be receipt free? In: Proceedings of IEEE S&P 2024, pp. 3440–3457. IEEE (2024)
8. Doan, T.V.T., Pereira, O., Peters, T.: Encryption mechanisms for receipt-free and perfectly private verifiable elections. In: Proceedings ACNS, pp. 257–287. Springer (2024)
9. Doan, T.V.T., Pereira, O., Peters, T.: Threshold receipt-free single-pass e-voting. In: Proceedings E-Vote-ID 2024, pp. 20–36. Springer (2024)
10. Doan, T.V.T., Pereira, O., Peters, T.: Threshold receipt-free voting with server-side vote validation. Cryptology ePrint Archive, Paper 2025/1321 (2025). https://eprint.iacr.org/2025/1321
11. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_21
12. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_24
13. Hirt, M.: Multi Party Computation: Efficient Protocols, General Adversaries, and Voting (2001)
14. McMurtry, E., Pereira, O., Teague, V.: When is a test not a proof? In: Chen, L., Li, N., Liang, K., Schneider, S. (eds.) ESORICS 2020. LNCS, vol. 12309, pp. 23–41. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59013-0_2
15. Ozdemir, A., Boneh, D.: Experimenting with collaborative zk-SNARKs: zero-knowledge proofs for distributed secrets. In: Proceedings of USENIX Security 2022, pp. 4291–4308. USENIX Association (2022)
16. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
17. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: Africacrypt, pp. 100–113. Springer (2010)

# End-To-End Verifiable Internet Voting with Partially Private Bulletin Boards

Valeh Farzaliyev and Jan Willemson(✉)

Cybernetica, Narva mnt 20, Tartu 51009, Estonia
{valeh.farzaliyev,jan.willemson}@cyber.ee

**Abstract.** In 2024, Harrison and Haines examined the applicability of STARKs in the context of homomorphically tallied elections. While their work ensures the Recorded-as-Cast and Tallied-as-Recorded properties of a voting system, it lacks Cast-as-Intended verification and does not provide a coercion mitigation mechanism. In this work, we address these challenges and propose an updated voting protocol that achieves all three verification properties, at the same time providing coercion resistance by allowing re-voting. Our approach leverages vector commitment schemes with update mechanisms. We implement our protocol and provide comparative benchmarks to the Harrison and Haines solution. Our approach significantly outperforms the latter, allowing processing a considerably larger number of votes within the same hardware limits.

**Keywords:** Electronic voting · zero-knowledge proofs · vector commitments

## 1 Introduction

Voting is the primary method used to delegate power in the democratic societies. The technical solutions to cast and count votes have gone through a long evolution, starting from raising one's hand to filling in paper ballots in the polling booths.

However, in the increasingly mobile world, gathering all the eligible voters in one place during a short period of time has become increasingly challenging. Thus, the need for reliable remote vote casting solutions has become more and more evident.

Practically speaking, there are two main alternatives for remote voting – casting the votes by paper mail, or over the Internet. Postal voting has the benefit of being familiar to the voters (the ballot can be exactly the same as for the in-person voting) and not requiring complex technical apparatus. However, in its classical form, postal voting is susceptible to coercion attacks, and it is hard to give integrity or secrecy guarantees for the postal channel. It has recently been even semi-formally shown by Vakarjuk *et al.* that postal voting has strictly weaker security guarantees than casting the votes over Internet [22].


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 73–89, 2026.
https://doi.org/10.1007/978-3-032-05036-6_5

Of course, Internet voting needs to satisfy all the requirements set for democratic elections, too, and meeting them is far from being trivial. In this paper, we are going to concentrate on two main classes of such requirements: integrity of the elections, and voting freedom.

By election integrity we mean that the end result adequately represents the combination of individual preferences. In order to ensure this, all the main steps of the elections must be independently verifiable. In particular, the following verifiability properties are important to us.

- **Cast-as-Intended** verifiability ensures that every ballot corresponds to the voter preference.
- **Recorded-as-Cast** verifiability ensures that the way preferences are stored in a (digital or physical) ballot box corresponds to the way the ballots were submitted by the voters (i.e. no ballot has been altered in, deleted from or added to the ballot box illegitimately).
- **Tallied-as-Recorded** verifiability ensures that the contents of the ballot box is converted to the end result without alterations in the process.

The first two properties are sometimes commonly known as *individual verifiability*, and the latter one as *universal verifiability*. The three properties together are often called *end-to-end (E2E) verifiability*. However, we note that there is no common definition of E2E verifiability, and it may vary significantly between different sources; see [7] for an overview and comparison of different verifiability notions.

Voting freedom, on the other hand, means that the voters are able to cast their votes according to their true preferences, i.e. without coercion. There are several possible ways proposed to achieve coercion resistance. Typical requirements include *voter privacy* and *vote secrecy*, with the rationale that if the coercer can not learn the voter's preference, it will be hard for him to actually achieve the goal of coercion.

In case of remote voting, voter privacy may be hard to ensure. The main threat here is over-the-shoulder coercion, where another person (say, a family member) attempts to find out how the voter voted with the aim of influencing her choice and making sure she complied with the influence.

Lack of privacy can, to a significant extent, be compensated by the option of re-casting one's vote later, with the last vote getting tallied. In order for the anti-coercion measures to be effective, the voting scheme should be *receipt-free*, i.e. it should be impossible for the voter to get a strong proof of her tallied vote to show to the coercer.

We note that the requirements of E2E verifiability and receipt-freeness are tricky to achieve at the same time. From the verifiability point of view, we would like to give the voter some evidence that her vote was tallied as intended. At the same time, this evidence should not be strong enough to allow for proving the content of the vote to an external party.

**Related Works.** There are numerous academic works trying to find a good trade-off between E2E verifiability and receipt-freeness. Cramer *et al.* [8] described the first efficient homomorphic tallying based multi-authority election system with universal verifiability and vote secrecy by using non-interactive zero-knowledge proof of ballot correctness together with homomorphic secret sharing. Adding cast-as-intended verifiability makes the scheme coercible, since all the ballot encryptions are stored on a public bulletin board. To address this problem, the use of secret bulletin boards were introduced in [9] where a new cryptographic primitive, *commitment consistent encryption*, was introduced. The secret bulletin board stores only encrypted ballots, while the public bulletin board appends consistently extracted commitments for each ballot. The authors also proposed efficient constructions for both homomorphically tallied and mix-net based election systems. Universal verifiability is achieved through *Perfectly Private Audit Trail* (PPAT). Furthermore, the authors suggest using re-voting to enhance coercion resistance. Unfortunately, the protocol is not equipped with a cast-as-intended verification mechanism to be considered as end-to-end verifiable.

When the secret bulletin boards and re-voting are simultaneously used, E2E verifiability requires verification that only the last vote by the voter is included in the final tally. However, in practice it is usually guaranteed by trusting election authority. Recently, a new paradigm – deniable, yet verifiable vote updating mechanism – has emerged [12,18,20]. Particularly, DeVoS [20] is fully compatible with PPAT by introducing a new trusted party.

**Our Contribution.** In this work, we demonstrate an online voting protocol that satisfies E2E verifiability, while providing coercion-resistance by allowing verifiable vote updating. To do so, we follow the approach by Harrison and Haines presented in [13] and extend their protocol design. The main idea is to replace the trust in the components of the voting scheme with well-defined zero-knowledge proofs. In the original work, Harrison and Haines utilize *authenticated data structures* and proof of homomorphic tallying. The intent of their work is to show applicability of modern zero-knowledge proofs systems in the context of voting, achieving recorded-as-cast, tallied-as-recorded, vote secrecy and everlasting voter privacy properties.

Our contributions extend the state of the art by adding the following features.

- Adding a cast-as-intended verification mechanism at the auditing phase.
- Allowing deniable yet verifiable re-voting to mitigate coercion.
- Introducing new proof relations for every stage of the voting process.
- Showing how to modify the scheme for mix-net based voting systems.
- Proof of concept implementation for small-scale elections[1].

[1] Available at https://github.com/Valeh2012/starkevoteid.

**Organization.** The rest of the paper is structured as follows. We start with explaining the mathematical notation, cryptographic primitives and the protocol by Harrison and Haines in Sect. 2. Next, we formally define our protocol in Sect. 3. Section 4 discusses possible modifications to the protocol design, implementation, benchmarking results and further optimizations. Finally, conclusions and future work are given in Sect. 5.

## 2 Preliminaries

### 2.1 Notation and Primitives

Let $\lambda$ be the security parameter. Let $n$ denote the number of voters, and let each voter be represented by an index $i = 1, \ldots, n$. If $\chi$ is a probability distribution over a set $D$, both $x \xleftarrow{\chi} D$ and $x \leftarrow \chi$ mean that $x$ is sampled from $D$ with respect to $\chi$. When $x$ is sampled uniformly, we us the notation $x \xleftarrow{\$} D$.

**Encryption Schemes.** A public-key encryption scheme consists of key generation, encryption and decryption algorithms:

- $\mathcal{E}.\mathrm{KeyGen}(1^\lambda)$: On the input of security parameter $\lambda$, return the secret key $sk$ and public key $pk$.
- $\mathcal{E}.\mathrm{Enc}(pk, m)$: Using the public key $pk$, compute and output the ciphertext $c$.
- $\mathcal{E}.\mathrm{Dec}(sk, c)$: Recover the plaintext message $m$ using the secret key.

In this work, we consider only CPA-secure encryption schemes, i.e. there should be no efficient probabilistic polynomial time algorithm that, given a ciphertext $ct$ and two messages $m_0, m_1$, can decide if $ct = \mathcal{E}.\mathrm{Enc}(pk, m_0)$ or $ct = \mathcal{E}.\mathrm{Enc}(pk, m_1)$. We also require the additive homomorphic property, i.e. that for all $sk, pk, m_0, m_1$

$$\mathcal{E}.\mathrm{Dec}(sk, \mathcal{E}.\mathrm{Enc}(pk, m_0) + \mathcal{E}.\mathrm{Enc}(pk, m_1)) = m_0 + m_1$$

holds.

Informally, CPA security implies a randomized encryption algorithm. Therefore, we sometimes use the notation $\mathcal{E}.\mathrm{Enc}(pk, m; r)$ to explicitly specify that $r$ is the used randomness, being sampled from an appropriate probability distribution. Moreover, we define re-randommization and special decryption algorithms that do not depend on the secret key.

- $\mathcal{E}.\mathrm{ReRand}(pk, ct, r')$: updates the randomness by adding $\mathcal{E}Enc(pk, 0; r')$ to the given ciphertect $ct$.
- $\mathcal{E}.\mathrm{SpecDec}(ct, r)$: having access to the encryption randomness $r$ (but not necessarily the private key!) extract the plaintext (e.g. by full inspection on the plaintext space in case it is small).

**Digital Signatures.** Similarly, a digital signature scheme is a collection of three algorithms:

- $\mathcal{S}$.KeyGen($1^\lambda$): On the input of security parameter $\lambda$, return the private signing key $sk$ and public verification key $pk$.
- $\mathcal{S}$.Sign($sk, m$): Return the signature $\sigma$ of the message $m$ using the signing key $sk$.
- $\mathcal{S}$.Verify($pk, m, \sigma$): Output 1 if $\sigma$ is the signature on $m$ being signed using the signing key that corresponds to the public key $pk$; otherwise output 0.

**Vector Commitment Schemes.** Vector Commitments ($\mathcal{VC}$) were first introduced in 2013 by Catalano and Fiore [6]. A $\mathcal{VC}$ scheme allows to commit to many elements, represented as a finite-dimensional vector, at once. Moreover, there should be an efficient opening to a message at a specific index, an update mechanism and a verification method for such updates. Formally, a Vector Commitment scheme $\mathcal{VC}$ is a set of five efficient algorithms KeyGen, Commit, Open, Verify, Update and ProofUpdate.

- $\mathcal{VC}$.KeyGen($1^\lambda$, n): Given the security parameter $\lambda$ and a vector of size $n$, output the public parameters $pp$.
- $\mathcal{VC}$.Commit($pp, m_1, m_2, \ldots m_n$): On the input of public parameters $pp$ and a vector of $n$ elements, output the commitment string $C$. It is equivalent to the notation $\mathcal{VC}$.Commit($pp, S$) where $S$ is a vector of $n$ elements.
- $\mathcal{VC}$.Open($pp, C, m_i, i$): Produce a proof $\Lambda_i$ that a given message $m_i$ is the $i$-th committed message.
- $\mathcal{VC}$.Verify($pp, C, m_i, i, \Lambda_i$): Return 1 if $\Lambda_i$ is a valid proof that $C$ opens to $m_i$ at position $i$; otherwise return 0.
- $\mathcal{VC}$.Update($pp, C, m_i, m'_i, i$): The committer who produced $C$ updates the commitment string by replacing the $i$-th message $m_i$ with $m'_i$ and returns $C'$.
- $\mathcal{VC}$.ProofUpdate($pp, C, \Lambda_j, m', i$): Update the commitment string $C$ and opening proof $\Lambda_j$ for some message $m'$ at position $i$. The output of this method is a new commitment string $C'$ and a new opening proof $\Lambda'_i$.

A vector commitment scheme $\mathcal{VC}$ is said to be *correct* if for all opening proofs $\Lambda_i$, either generated by $\mathcal{VC}$.Open or $\mathcal{VC}$.ProofUpdate, $pp \leftarrow \mathcal{VC}$.KeyGen($1^\lambda, n$), and all commitment strings $C$ that commit to a vector message with $i$-th element being $m_i$, $\mathcal{VC}$.Verify($pp, C, m_i, i, \Lambda_i$) is 1 (with overwhelming probability).

The scheme is *position binding* if no polynomial time adversary with the knowledge of $pp$ can find a commitment string $C$ that opens to two distinct messages at any position. Moreover, the scheme is *concise* if the commitment and opening proof lengths are independent of $n$. We also require *hiding* schemes, meaning that an adversary cannot distinguish whether $C$ is a commitment to $m_1, m_2, \ldots, m_n$ or $m'_1, m'_2, \ldots, m'_n$ even after seeing some openings.

In [13], the authors give a definition of *Authenticated Data Structures* (ADS) which overlaps with the definition of Vector Commitments above, except for the update mechanism.

**Zero-Knowledge Protocols.** Let $R = \{(x; w) \mid x \in \mathcal{L}\}$ be a relation for an NP-language $\mathcal{L}$. Here, $x$ is called a *statement* and $w$ is the *witness*. A zero-knowledge proof system is a multi-round interactive protocol between a prover and a verifier which satisfies three properties.

- *Completeness*: Honest verifier accepts an honestly generated proof if $x \in \mathcal{L}$.
- *Soundness*: Cheating prover cannot convince the verifier if $x \notin \mathcal{L}$ .
- *Zero-knowledge*: Verifier learns nothing other than the fact that $x \in \mathcal{L}$.

These properties have *perfect*, *statistical* and *computational* variations. Simply speaking, perfect and statistical versions say that the requirements are satisfied with probability 1 or statistically close to 1, respectively, while the computational condition restricts the adversary to be computationally efficient. For soundness, a stronger notion *knowledge-soundness* asserts that if the prover can convince the verifier, then it must know the witness. The zero-knowledge property can be proved by constructing a simulator $\mathcal{S}$ (potentially more powerful than the verifier) which can generate an accepting transcript without the knowledge of $w$. In case the proof system is only computationally sound, it is called an *argument*.

If there is only a single round of communication, then we speak about a non-interactive zero-knowledge (NIZK) proof. All public coin interactive proofs can be turned into NIZK proofs using Fiat-Shamir transformation [10]. Furthermore, when the proof size is at most polylogarithmic in the witness size, we call it *succinct*.

Recently, many interesting succinct non-interactive arguments of knowledge (SNARK [11,17,19]) and succinct transparent arguments of knowledge (STARK [2]) have been proposed. STARKs are quantum-resistant by construction; however they require more computational power than SNARKs. There exist constant-size SNARKs, but all the known STARKs have logarithmic proof size. On the other hand, STARK verification outperforms the most efficient SNARK verifiers in their runtime. Unless explicitly said otherwise, we assume all SNARKs (STARKs) are zero-knowledge instead of denoting them zkSNARKs (zkSTARKs) for distinction.

### 2.2 The Protocol by Harrison and Haines

In [13], Harrison and Haines build a simple e-voting protocol upon the ideas from [8]. Protocol participants are the Election Authority (EA), the voters $V_1, V_2, \ldots, V_n$ and the talliers $T_1, T_2, \ldots, T_m$. There are two append-only bulletin boards: the public bulletin board $\mathcal{PB}$ and the secret bulletin board $\mathcal{SB}$. The talliers also have read-only access to $\mathcal{SB}$.

The EA is responsible for organizing the election. It provides public information such as the list of candidates, the public signature verification keys of eligible voters, the public signature verification keys of valid talliers, and the public voting parameters. It is assumed that an authentic copy of the EA's public signature verification key ($pk^{EA}$) is known to all participants.

The authors of [13] chose ElGamal encryption scheme over elliptic curves to ensure ballot privacy. The encryption secret key $sk$ is jointly generated by the set of talliers using standard secret sharing techniques and the public key is computed as $PK = \text{sk} \cdot G$ (where $G$ is the generator point of the elliptic curve group). Moreover, the talliers keep their own shares of the secret key.

During the voting phase, each voter first verifies public voting parameters, and after that encrypts her vote. The ballot message is a pair consisting of the ciphertext and a NIZK proof of ballot correctness. At last, the voter sends the signed ballot message $\sigma = \mathcal{S}.\text{Sign}(sk, (c, \pi))$ to $\mathcal{SB}$ through a private channel (here $sk$ is the secret signing key, $c = \mathcal{E}.\text{Enc}(PK, v)$ and $\pi$ is the ballot correctness proof for the vote $v$).

After the voting phase ends, the talliers start to count the collected votes. They iterate over the submitted ballots, verify signatures using the public signature verification keys of the corresponding voters, and check that the ballot correctness proofs pass. Next, they add ciphertexts to get the encrypted result $C_j = \sum_{i=1}^{n} c_i$. Now, the results can be decrypted by the talliers using their secret key shares. In addition to this, each tallier commits to the list of voters' public signature keys and added ciphertexts. Following the notation of [13], let $r_p = \mathcal{VC}.\text{Commit}(pp, pk_1, pk_2, \ldots, pk_n)$ and $r_c = \mathcal{VC}.\text{Commit}(pp, c_1, c_2, \ldots, c_n)$. The ciphertext commitment is blinded with a randomness factor $\eta \in \mathbb{Z}_q$. At this point, a STARK proof is generated for the relation

$$R = \left\{ \begin{array}{c|c} (r_p, r'_c, C); & r_p = \mathcal{VC}.\text{Commit}(pp, pk_1, \ldots, pk_n) \\ (pk_1, \ldots, pk_n, c_1, \ldots, c_n, & \wedge\, r'_c = \mathcal{VC}.\text{Commit}(pp, c_1, \ldots, c_n) \cdot \eta \\ \sigma_1, \ldots, \sigma_n, \pi_1, \ldots, \pi_n, \eta) & \wedge\, \mathcal{S}.\text{Verify}(pk_j, \sigma_j, c_j) = 1 \\ & \wedge\, Verify(c_j, \pi_j, PK) = 1\ \forall j = 1, \ldots, n \\ & \wedge\, C = \prod_{i=1}^{n} c_i \end{array} \right\}$$

with the public statement $(r_p, r'_c, C_j)$ and private witness $(pk_1, \ldots, pk_n, c_1, \ldots, c_n, \sigma_1, \ldots, \sigma_n, \pi_1, \ldots, \pi_n, \eta)$. Finally, each tallier signs and appends the decryption result and the STARK proof on the public bulletin board $\mathcal{PB}$.

The last stage of this e-voting protocol is called EXTRACT, where the EA verifies all the signatures and included proofs written on the $\mathcal{PB}$, and computes the final result as the sum of extracted partial decryptions. Finally, the EA signs the final result and reveals it.

For the implementation, Harrison and Haines choose ElGamal over STARK elliptic curve [21], Pedersen hash function [15, Section 5.4.1.7], Elliptic Curve Digital Signature Algorithm (ECDSA) [16], Merkle Trees as the vector commitment, and ElectionGuard's ballot correctness proof [3].

The authors mention that the contribution is not the protocol definition itself, but presentation and evaluation of the STARK proof as an efficient alternative to verify talliers' work without publicly exposing the submitted ballots. Therefore, they do not prove the informal claim that, assuming the vector commitment is secure, binding and hiding, and the non-interactive proofs are complete, sound and zero-knowledge, the proposed e-voting protocol satisfies everlasting privacy, ballot privacy, recorded-as-cast and tallied-as-recorded properties.

## 3 Our Vector Commitment-Based Voting Protocol

In this section, we formally define our online voting protocol based on vector commitments, arguing that the protocol is end-to-end verifiable and coercion resistant. For the latter property, the ballots are stored privately on a private bulletin board, and re-voting is allowed. Assuming the coercer does not block the voters from submitting their choices, but only demands a proof that the submitted ballots encode the coercer's wish, the voters can generate such proofs at the Cast-as-Intended phase, and later vote again. If the coercer does not have access to the submitted ballots, he cannot tell if the voter has voted again, and hence the coercive strategy would fail.

However, having a private bulletin board raises concerns about the correct processing of the recorded ballots. Therefore we propose a second append-only bulletin board which is public and contains auxiliary information about each submitted ballot. This information must be verifiable by anyone for universal verifiability. It also has a purpose of linking two consecutively received ballots. Furthermore, the last entry on the public bulletin board guarantees the order of ballot history and is used at the later stages of the election to guarantee that the ballots are recorded-as-cast.

Before describing the proposed protocol explicitly, we first give a brief list of the used methods.

### 3.1 Building Blocks

Let $\mathcal{E}$ be a CPA-secure additively homomorphic encryption scheme to encrypt the votes, let $\mathcal{S}$ be a digital signature scheme to sign the ballots, and let $\mathcal{VC}$ be a position binding vector commitment scheme to commit to the submitted ballots. Let $n$ be the number of eligible voters and let $PK^{\mathcal{S}} = \{pk_1^{\mathcal{S}}, pk_2^{\mathcal{S}}, \ldots, pk_n^{\mathcal{S}}\}$ be the set of the voters' public keys. Let $\Upsilon$ be the set of possible choices; e.g., for simple yes/no elections the set $\Upsilon = \{0, 1\}$ can be used.

**Ballot Correctness Proof.** A ballot correctness proof, or simply a ballot proof, is a non-interactive zero-knowledge proof of knowledge of intent, generated by each voter to show that their choice under encryption satisfies election rules. Let $c = \mathcal{E}.\mathrm{Enc}(pk^{\mathcal{E}}, v)$ be an encrypted ballot where $v \in \Upsilon$. A ballot proof $\Pi_{BP}$ is a NIZK for the following relation:

$$R_{BP} = \left\{(c, pk^{\mathcal{E}}, \Upsilon); (v, r) \;\middle|\; c = \mathcal{E}.\mathrm{Enc}(pk^{\mathcal{E}}, v; r) \wedge v \in \Upsilon\right\}.$$

**Update Proof.** Let $C = \{c_1, c_2, \ldots, c_j, \ldots, c_n\}$ be the set of ballots submitted by the voters at some moments in time, and let the voter $j$ submit a new signed ballot $\sigma = \mathcal{S}.\mathrm{Sign}(sk_j^{\mathcal{S}}, (c'_j, \Pi_{PB}))$. Furthermore, let $\Lambda_j = \mathcal{VC}.\mathrm{Open}(pp, VC, c, j)$ and $(\Lambda'_j, VC') = \mathcal{VC}.\mathrm{ProofUpdate}(pp, VC, \Lambda_j, c', j)$ be opening proofs to the ballots before and after this submission at position $j$. Update proof $\Pi_U$ is a NIZK for the following relation:

$$R_U = \left\{ \begin{array}{c} \\ (pp, VC, VC', \\ PK^{\mathcal{S}}, \Upsilon); (j, c_j, \sigma, \\ c'_j, \Pi_{PB}, pk_j^{\mathcal{S}}, \Lambda_j, \Lambda'_j) \\ \\ \end{array} \middle| \begin{array}{c} \Pi_{BP}.\text{Verify}(c'_j, pk^{\mathcal{E}}, \Upsilon) = 1 \ \wedge \ pk_j^{\mathcal{S}} \in PK^{\mathcal{S}} \\ \wedge \ \mathcal{S}.\text{Verify}(pk_j^{\mathcal{S}}, (c'_j, \Pi_{BP}), \sigma) = 1 \\ \wedge \ \mathcal{VC}.\text{Verify}(pp, VC, c_j, j, \Lambda_j) = 1 \\ \wedge \ \mathcal{VC}.\text{Verify}(pp, VC', c'_j, j, \Lambda'_j) = 1 \ \wedge \\ (\Lambda'_j, VC') = \mathcal{VC}.\text{ProofUpdate}(pp, VC, \Lambda_j, c', j) \end{array} \right\}.$$

**Opening Proof.** While $\mathcal{VC}$.Open method efficiently generates a proof for an opening at a particular position, generating such proofs for every position and verifying them would take a substantial amount of time at the beginning of the election. Instead, a NIZK proof for the relation $(VC; C)$ where $VC = \mathcal{VC}.\text{Commit}(pp, C)$ can be generated so that verifying it would take only marginal amount of time. This relation can be extended with additional constraints. For example, it may be desirable to let $C$ be an empty set, a set of zeros, or zero-encryptions. For the latter, we formally define the opening proof relation as

$$R_{\perp} = \left\{ (pp, pk^{\mathcal{E}}, VC); C \ \middle| \ \begin{array}{c} VC = \mathcal{VC}.\text{Commit}(pp, C) \\ \wedge \ c = \mathcal{E}.\text{Enc}(pk^{\mathcal{E}}, 0) \quad \forall c \in C \end{array} \right\}.$$

**Accumulation Proof.** Again, consider that $VC$ is a vector commitment to $C = \{c_1, c_2, \ldots, c_n\}$ before tallying starts. In case of homomorphically tallied elections, ciphertexts are added before the decryption phase. For that, we define accumulation proof $\Pi_{Acc}$ for the relation

$$R_{acc} = \left\{ (pp, VC, c_{acc}); C \ \middle| \ VC = \mathcal{VC}.\text{Commit}(pp, C) \ \wedge c_{acc} = \sum_{c \in C} c \right\}.$$

**Decryption Proof.** The final NIZK proof, $\Pi_{Dec}$, guarantees that the plaintext $m$ is a correct decryption of $c$ under the key $sk^{\mathcal{E}}$. We can write the relation formally as

$$R_{Dec} = \left\{ (c, m, pk^{\mathcal{E}}); sk^{\mathcal{E}} \ \middle| \ m = \mathcal{E}.\text{Dec}(sk^{\mathcal{E}}, c) \wedge (pk^{\mathcal{E}}, sk^{\mathcal{E}}) \leftarrow \mathcal{E}.KeyGen(1^{\lambda}) \right\}.$$

### 3.2 Full Protocol

Our voting protocol consists of the following components:

- *Announcement*, *Registration*, *Setup*, *Casting* and *Tallying* phases;
- **Election Setup**, **KeyGen**, **Vote**, **Verify** and **Tally** protocols;
- append-only databases called Private Bulletin Board $\mathcal{SB}$, and Public Bulletin Board $\mathcal{PB}$;

– participants running the protocol: the Election Authority EA, Voters $V_j$, Vote Collector Server (VCS), and Decryption Authority $O^{\mathcal{D}}$.

The protocol proceeds as follows.

– *Announcement:* First, the EA decides upon a CPA-secure asymmetric encryption scheme $\mathcal{E}$, digital signature scheme $\mathcal{S}$ and vector commitment scheme $\mathcal{VC}$ to use, initializes $\mathcal{PB}$ and $\mathcal{SB}$, announces the election parameters and the chosen cryptographic suite.
– *Registration:* Every potential voter $V_j$ generates her own pair of public and private signing keys $(pk_j^{\mathcal{S}}, sk_j^{\mathcal{S}}) = \mathcal{S}.\text{KeyGen}(1^\lambda)$, and registers herself by sending the $pk_j^S$ to the $\mathcal{PB}$. Upon receiving the public signing key from the voter, $\mathcal{PB}$ adds it to the list of public keys $PK^{\mathcal{S}} = PK^{\mathcal{S}} \cup \{pk_j^{\mathcal{S}}\}$ and increments the number of voters $n$. Thus, at the end of the registration phase, we have $n = |PK^{\mathcal{S}}|$.
– *Setup:* After the Registration phase ends, $O^{\mathcal{D}}$ generates the encryption-decryption key pair $(pk^{\mathcal{E}}, sk^{\mathcal{E}}) = \mathcal{E}.\text{KeyGen}(1^\lambda)$, and sends the public encryption key to EA. EA runs the vector commitment scheme setup $pp = \mathcal{VC}.\text{Setup}(1^\lambda, n)$, and appends an empty commitment $VC = \mathcal{VC}.\text{Com}(pp, C^0)$ where $C^0 = (c_i^0)_{i=1}^n$ and $c_i^0$ is an encryption of zero message for all $i = 1, \ldots, n$, along with a proof of opening $\Pi_\perp$ to $\mathcal{PB}$.
– *Casting:* The voter $V_j$
  - samples randomness $r \xleftarrow{\$} R$ (where $R$ is the randomness space chosen as a part of the public parameters),
  - encrypts her vote $v \in \Upsilon$ using randomness $r$ and election public encryption key as $c' = \mathcal{E}.\text{Enc}(pk^{\mathcal{E}}, v; r)$,
  - generates the ballot correctness proof $\Pi_{BP}$ for the relation $((c', pk^{\mathcal{E}}, \Upsilon); (v, r))$,
  - signs the ciphertext using her secret signing key as $\sigma = \mathcal{S}.\text{Sign}(sk_j^{\mathcal{S}}, (c', \Pi_{BP}))$,
  - submits the ballot $(\sigma, c', \Pi_{BP})$ to the Vote Collector Server.

  When VCS receives the ballot $(\sigma, c', \Pi_{BP})$, it checks that the validity of the signature and the ballot proof as the first step. If they pass, VCS
  - updates the vector commitment $VC' = \mathcal{VC}.\text{Update}(pp, VC, c, c', j)$ at position $j$ with the new value $c'$, and
  - generates the opening proof $\Lambda' = \mathcal{VC}.\text{Open}(pp, VC', c', j)$.
  - Assuming $VC, \Lambda$ and $c$ are the previous vector commitment, an opening proof and its opening at the same position, respectively, VCS generates the proof of valid commitment update $\Pi_U$ of $((pp, VC, VC', PK^{\mathcal{S}}); (j, c, \sigma, c', \Pi_{BP}, pk_j^{\mathcal{S}}, \Lambda, \Lambda')) \in R_U$.
  - Last, VCS appends $(\sigma, c', \Pi_{BP}, \Lambda')$ to $\mathcal{SB}$, and $(\Pi_U, VC')$ to $\mathcal{PB}$.
  - Returns unique vote reference $vr$ to the voter.
– *(Homomorphic) Tallying:* At the final phase, EA runs the Tally protocol on the input of all update proofs $(\Pi_U^{(i)})_{i=1}^{\nu}$ and history of the vector commitments $(VC^{(i)})_{i=1}^{\nu}$ and committed ciphertexts $(c_j)_{j=1}^n$. Within the Tally protocol, first all update proofs are verified, checking that indeed $VC^{\nu}$ is the correct

vector commitment to $(c_j)_{j=1}^n$. Then, all the ciphertexts are added as $c_{acc} = \sum_{j=1}^n c_j$, and a proof of accumulation $\Pi_{Acc}$ is produced. Last, EA queries the Decryption Authority $O^{\mathcal{D}}$ to decrypt $c_{acc}$. $O^{\mathcal{D}}$ returns the decrypted tally and proof of correct decryption next to it. EA publishes the tally and all supporting proofs to $\mathcal{PB}$.

### 3.3 Individual Verifiability

Up to this point, our protocol achieves recorded-as-cast and tallied-as-recorded properties, assuming that the vector commitment is correct and positional binding, proofs are knowledge sound and zero-knowledge. The former follows directly from the soundness of update proofs and positional binding property of the vector commitment scheme, while the latter is satisfied due to the accumulation proof being sound. Vote secrecy follows from the CPA-security of the encryption scheme. Ballot signatures confirm eligibility of the voters. Finally, voter anonymity is preserved as the update proofs are zero-knowledge.

However, in the presented form, there is no mechanism to verify the cast-as-intended property (that is, we are not yet doing better than [13]). To add this functionality, we expand the protocol with the *Audit* phase, applying cast-and-audit strategy. That is, the voters can query the submitted ballot using a separate audit device and verify that the encrypted ballot contains their vote after they have finished voting (following the approach proposed by Heiberg and Willemson [14]). In addition, we assume that once the VCS returns the a unique vote reference $vr$ to the voter, that reference is not leaked. The voting device used by the voter then displays this vote reference together with the encryption randomness (e.g. as a QR code).

Let $c'$ be encryption of the vote $v$ using randomness $r$, i.e., $c' = \mathcal{E}.\text{Enc}(pk^{\mathcal{E}}, v; r)$, and let the VCS return the vote reference $vr$. The voter imports $vr$ and $r$ into the audit device (for example, by scanning the QR code). Then the audit device queries VCS to get the encrypted ballot $c'$, proof of update $\Pi_U$, and the proof of opening $\Lambda'$. Next, it verifies $\Pi_U$ and checks that $\mathcal{VC}.\text{Verify}(pp, VC', c', i, \Lambda') = 1$. Note that the position $i$ can be provided by VCS within $\Lambda$ or obtained by the voter herself simply by looking at the index of her public signing key in the list $PK^S$. If both verification checks pass, the audit device attempts to extract the vote $v^*$ using special decryption algorithm as $v^* = \mathcal{E}.SpecDec(c', r)$. The voter now can verify if $v = v^*$. If VCS is honest, $v = \mathcal{E}.SpecDec(\mathcal{E}.\text{Enc}(pk^{\mathcal{E}}, v; r), r) = v^*$ holds due to correctness of the encryption algorithm, and the voter accepts the verification result. If $v \neq v^*$, the voter rejects. As a result, our proposed voting protocol achieves the cast-as-intended property. Moreover, as $(\Pi_U, VC')$ pair is on $\mathcal{PB}$, and assuming the zero-knowledge proof is sound and the vector commitment scheme is complete, the voter is convinced that her vote is cast-as-intended and recorded-as-cast. Combining this with the tallied-as-recorded property, we get end-to-end verifiability.

We observe that our cast-as-intended and recorded-as-cast verification forms a receipt of how the voter voted. However, due to allowing for re-voting, the risk of coercion is mitigated. Assuming the coercer has no access to the communication between the voters and the VCS, or cannot read the contents of the private bulletin board, it is not possible for him to to tell whether the voter has re-voted or not. In addition, the update proofs do not leak any information regarding the voter. Therefore the coercer can not simply see whether a voter has voted or not, so the protocol also provides protection against forced abstention attacks.

## 4 Discussion

The protocol described in Sect. 3 can be easily augmented to comply with different election requirements. For example, currently there is a single authority who holds the private encryption key, and the authority is modeled as an oracle. This is a threat to vote secrecy, because the tallier may query the oracle arbitrarily many times to learn the contents of each ballot. The trivial solution would be limiting access to the oracle to only one query; however, in practice, it would be hard to enforce this rule. Malicious tallier and decryption oracle can collude to decrypt any submitted ballot. The standard solution is to split the private decryption key into a number of shares and distribute them among several registered talliers. Since each tallier can keep their share of the private key, there is no need for a separate decryption oracle. In the modified tallying phase, each tallier will run the Tally protocol, decrypting the accumulated ciphertext and generating the proof of correct decryption by themselves, in parallel. As a result, each tallier will publish their share of decryption. If at least one tallier is honest and does not collude with others, vote secrecy will be retained.

We can also consider more general algorithms for computing election result. For example, a mix-net version of the protocol given in Sect. 3.2 is relatively straightforward to implement. We can add a *Mixing* phase before *Tallying*, and a Mixer will run the **Mixing** protocol at this phase.

– The *Mixing* phase will consist of the following steps:
  - Verify all the previous proofs.
  - Get a list of ciphertexts from $\mathcal{SB}$.
  - Re-randomize each ciphertext with fresh randomness.
  - Shuffle the ciphertexts using a secret permutation, commit to them.
  - Generate NIZK proof of correct shuffle, $\Pi_{shuffle}$.
  - Publish ciphertexts on $\mathcal{PB}$ in the shuffled order together with $\Pi_{shuffle}$,

where $\Pi_{shuffle}$ is a proof for the shuffle relation $R_{shuffle}$:

$$R_{shuffle} = \left\{ \begin{array}{c} ((pp, pk^{\mathcal{E}}, VC, \\ c'_1, \ldots, c'_n); \\ (\pi, c_1, \ldots, c_n, \\ r_1, \ldots, r_n)) \end{array} \middle| \begin{array}{c} VC = \mathcal{VC}.\text{Commit}(pp, c_1, \ldots, c_n) \wedge \\ c'_{\pi(i)} = \mathcal{E}.\text{ReRand}(pk^{\mathcal{E}}, c_i, r_i)\ \forall i = 1, \ldots, n \end{array} \right\}$$

In this version, the talliers do not need to add ciphertexts and produce the accumulation proof. However, they have to decrypt all the ciphertexts one by one and produce the accompanying proofs of correct decryption. Note that because the Mixer gets input directly from the private bulletin board, this input is not part of the statement, but rather part of the witness in the shuffle proof. Regular mix-nets, however, treat incoming and outgoing ciphertexts as public values. Finally, one can add many regular mix-nets after the Mixer to further reduce the link between the voter and her vote. All such modifications do not change the main properties of the voting system.

Finally, it is important to mention that who can generate valid signatures on behalf of the voter, can also submit ballots that would go undetected by the voter since the update proofs do not leak voter identity. Therefore, we assume a trusted signing device that hides private signing keys, and require confirmation from the voter every time she has to sign a document. In practice, this can be realized by separate hardware (smart cards[2]) or software (digital wallets).

### 4.1 Implementation and Benchmarks

The implementation details of the original Harrison and Haines scheme were presented in Sect. 2.2. We use the same components to instantiate our protocol. That is, we consider simple YES/NO election with one question, ElGamal encryption scheme and corresponding NIZK ballot proof, ECDSA digital signature, Merkle Tree with Pedersen hash function as the vector commitment scheme and STARK proof system. All these schemes are defined over the STARK curve. We perform membership checks for public signature keys using another Merkle Tree.

We have extended the source code provided in [13], and added circuits for relations defined in Sect. 3.1. All the circuits are written in Cario 0 language, and proofs are generated using the STONE prover. Our test device has a 4-core (8-threads) Intel(R) Core(TM) i5-82500 CPU running at 1.60 GHz, and 16 GB of RAM. The operating system is Ubuntu 24.04.2 LTS.

First, we simulated the vote casting phase with up to $2^{32}$ voters. In all the experiments, the (average) times required to generate and verify a single update proof remained constant, being 0.7 **core-minutes** and 0.15 **core-seconds**, respectively. As the proof file contains public input, its size grew linearly from 980 KB to 1.0 MB. Peak memory usage was consistently less than 1 GB.

Next, we measured the performance of the accumulation circuit for different numbers of voters. Our implementation reached its memory limit with 1024 voters. In this case, it takes 20.8 **core-minutes** to generate the 1.8 MB accumulate proof, and only 0.21 **core-seconds** to verify the proof with peak memory usage 11.82 GB. In comparison, Harrison and Haines were only able to accommodate up to 64 voters at the similar RAM usage, but slightly longer proof generation time. We summarize all the experiment results in Table 1, with projections for

[2] In case of smart cards, voters are advised to use smart card readers with keypad to avoid PIN caching attacks.

bigger inputs given in gray. Our work noticeably outperforms [13] in all the input sizes.

For a fair comparison, we considered the performance of our protocol with $2^{24}$ voters. In [13], the authors estimate the total proof size to be 1.54 GB, and the verification to take 5.62 **core-minutes** (after adjusting for clock frequency) if batching is applied (512 batches with $2^{15}$ batch size). With this configuration, the total proof size for accumulation proofs using our protocol would be 1.14 GB and the verification would take 2.13 **core-minutes**. Moreover, less than 0.5 TB RAM is enough in contrast to the assumed 6 TB for the tallier machine in [13]. The downside of our proposed scheme is that minimum $2^{24}$ MB $\approx$ 16 TB disk space is needed to store the individual update proofs, and verifying them sequentially requires 41 **core-days**. This is the cost of having verifiable vote updating.

**Table 1.** Comparing performance of accumulate proof generated by the tallier. Entries in gray are projected results. Units for Proving, Verification, Size and Peak RAM columns are in core-minutes, core-seconds, MB and GB, respectively.

| Input | $\Pi_U$ | | | | Harrison and Haines [13] | | | |
|---|---|---|---|---|---|---|---|---|
| Votes | Proving | Verification | Size | Peak RAM | Proving | Verification | Size | Peak RAM |
| 4 | 0.19 | 0.11 | 0.542 | 0.57 | 1.77 | 0.18 | 1.009 | 0.72 |
| 8 | 0.22 | 0.14 | 0.901 | 0.62 | 3.23 | 0.16 | 1.062 | 1.38 |
| 16 | 0.35 | 0.14 | 0.923 | 0.73 | 6.46 | 0.17 | 1.148 | 2.82 |
| 32 | 0.64 | 0.15 | 1.014 | 0.95 | 12.91 | 0.17 | 1.317 | 5.50 |
| 64 | 1.15 | 0.15 | 1.047 | 1.38 | 26.25 | 0.22 | 1.620 | 11.0 |
| 128 | 2.37 | 0.16 | 1.113 | 1.84 | 52.35 | 0.23 | 1.824 | 21.96 |
| 256 | 4.78 | 0.17 | 1.227 | 3.97 | 104.73 | 0.24 | 1.972 | 43.88 |
| 512 | 10.16 | 0.18 | 1.431 | 7.42 | 209.49 | 0.25 | 2.120 | 87.73 |
| 1024 | 20.8 | 0.21 | 1.835 | 11.82 | 419.01 | 0.27 | 2.267 | 175.41 |
| 2048 | 41.36 | 0.21 | 1.737 | 24.00 | 838.04 | 0.28 | 2.415 | 350.79 |
| 4096 | 82.80 | 0.22 | 1.861 | 47.33 | 1676.10 | 0.29 | 2.563 | 701.53 |
| 8192 | 165.68 | 0.23 | 1.987 | 93.99 | 3352.22 | 0.31 | 2.711 | 1403.03 |
| 16384 | 331.4 | 0.24 | 2.111 | 187.30 | 6704.46 | 0.33 | 2.855 | 2803.02 |
| 32768 | 662.95 | 0.25 | 2.235 | 373.93 | 13408.94 | 0.34 | 3.004 | 5612.01 |

### 4.2 Post-quantum Instantiation

Unfortunately, lattice-based encryption and digital signature schemes are not STARK-friendly. Therefore, we choose Labrador [4] proof system and Falcon digital signatures, whose verification has been implemented in [1]. Next, BGV [5]

without bootstrapping is a good choice for an additively homomorphic encryption scheme. Finally, Merkle Trees with any post-quantum secure hash function stays as the vector commitment scheme. Choosing parameters to satisfy e.g. the 128-bit security level is currently an open problem, and it is not a straightforward task due to non-trivial interdependencies between the parameters.

### 4.3 Further Optimizations

Even though the proposed proof systems have extremely fast verification times, for mid- and large-scale elections, downloading and verifying update proofs may take several hours. Luckily, there is an elegant solution – Incrementally Verifiable Computation (IVC) [23], a recent breakthrough in zero-knowledge proof systems. The main idea behind IVC is to recursively generate a new proof for a certain relation and verify the previous proof as a subtask inside the new proof. If we think of generating the proof of update as an incremental function $F$ that takes the input statement and the witness, we can generate a single succinct proof at the end of the voting phase. This will drastically reduce the workload of external verifiers. In practice, this increases memory usage and proof generation time at the prover's side. In other words, the Vote Collector Server should be very powerful in order to provide a seamless voting experience.

## 5 Conclusion

This paper presents a possible approach to designing an end-to-end verifiable and coercion-resistant voting protocol with cast-and-audit individual verification and re-voting. Our work is inspired by [13]; however, our construction is based on a verifiably updatable vector commitment scheme to bind all submitted ballots to the election outcome. All the voting protocol participants produce associated zero-knowledge proofs which are verifiable by anyone. A simple cast-as-intended verification ensures voters that their interaction with the election is not tampered with. We also allow for re-voting to mitigate coercion attacks, while promising that the last vote is included in the tally. We achieve these results by assuming voters generate digital signatures using a trusted device. Simple proof of concept implementation shows that the scheme is deployable in practice. With further optimizations, it is possible to reduce the number of produced artifacts, so the workload of external verifiers. We leave such optimizations and post-quantum implementation a possible direction for future works.

**Acknowledgments.** The paper has been supported by the Estonian Research Council under the grant number PRG2177.

## References


1. Aardal, M.A., Aranha, D.F., Boudgoust, K., Kolby, S., Takahashi, A.: Aggregating falcon signatures with LaBRADOR. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024, pp. 71–106. Springer, Cham (2024)
2. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 701–732. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_23
3. Benaloh, J., Naehrig, M.: Electionguard design specification (version 2.0.0), Technical report, Microsoft Research (2023)
4. Beullens, W., Seiler, G.: LaBRADOR: compact proofs for R1CS from module-SIS. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023, pp. 518–548. Springer, Cham (2023)
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. Electron. Colloquium Comput. Complex. **TR11** (2011). https://api.semanticscholar.org/CorpusID:2182541
6. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_5
7. Cortier, V., Galindo, D., Küsters, R., Müller, J., Truderung, T.: SoK: verifiability notions for e-voting protocols. In: IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22–26, 2016, pp. 779–798. IEEE Computer Society (2016). https://doi.org/10.1109/SP.2016.52
8. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. Eur. Trans. Telecommun. **8**(5), 481–490 (1997). https://doi.org/10.1002/ETT.4460080506
9. Cuvelier, É., Pereira, O., Peters, T.: Election verifiability or ballot privacy: do we need to choose? In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 481–498. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40203-6_27
10. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology – CRYPTO' 86, pp. 186–194. Springer, Berlin, Heidelberg (1987)
11. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_37
12. Haines, T., Müller, J., Querejeta-Azurmendi, I.: Scalable coercion-resistant e-voting under weaker trust assumptions. In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, pp. 1576–1584. SAC '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3555776.3578730
13. Harrison, M., Haines, T.: On the applicability of starks to counted-as-collected verification in existing homomorphic e-voting systems. In: Financial Cryptography and Data Security. FC 2024 International Workshops, pp. 50–65. Springer, Cham (2025)
14. Heiberg, S., Willemson, J.: Verifiable Internet voting in Estonia. In: Krimmer, R., Volkamer, M. (eds.) 6th International Conference on Electronic Voting: Verifying the Vote, EVOTE 2014, Lochau / Bregenz, Austria, October 29–31, 2014, pp. 1–8. IEEE (2014). https://doi.org/10.1109/EVOTE.2014.7001135

15. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification 2022.3.8 [nu5], Technical report, Electric Coin Company (2022)
16. Johnson, D., Menezes, A., Vanstone, S.A.: The elliptic curve digital signature algorithm (ECDSA). Int. J. Inf. Sec. **1**(1), 36–63 (2001). https://doi.org/10.1007/S102070100002
17. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing, pp. 723–732. STOC '92, Association for Computing Machinery, New York, NY, USA (1992). https://doi.org/10.1145/129712.129782
18. Lueks, W., Querejeta-Azurmendi, I., Troncoso, C.: VoteAgain: a scalable coercion-resistant voting system. In: 29th USENIX Security Symposium (USENIX Security 20), pp. 1553–1570. USENIX Association (2020). https://www.usenix.org/conference/usenixsecurity20/presentation/lueks
19. Micali, S.: Cs proofs. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 436–453 (1994). https://doi.org/10.1109/SFCS.1994.365746
20. Mueller, J., Pejo, B., Pryvalov, I.: DeVoS: deniable yet verifiable vote updating. Cryptology ePrint Archive, Paper 2023/1616 (2023). https://eprint.iacr.org/2023/1616
21. StarkEx: STARK curve (2025). https://docs.starkware.co/starkex/crypto/stark-curve.html
22. Vakarjuk, J., Snetkov, N., Willemson, J.: Comparing security levels of postal and Internet voting. Inf. Secur. J. Glob. Perspect. **34**(4), 265–285 (2024). https://doi.org/10.1080/19393555.2024.2410332
23. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) Theory of Cryptography, pp. 1–18. Springer, Berlin, Heidelberg (2008)



# Attack Once, Compromise All? On the Scalability of Attacks

Eva Hetzel[1(✉)], Marc Nemes[2], and Jörn Müller-Quade[1]

[1] KASTEL Security Research Labs, Karlsruhe Institute of Technology, Karlsruhe, Germany
{eva.hetzel,joern.mueller-quade}@kit.edu
[2] FZI Research Center for Information Technology, Karlsruhe, Germany
nemes@fzi.de

**Abstract.** Electronic voting schemes are often criticized for being insecure, on the grounds that a successful attack would allow an adversary to manipulate all votes at once. It is argued that attacks therefore have a higher impact at lower adversary costs compared to paper-based schemes, where attacks are cumbersome. In this paper, we propose a framework to quantify how prone different protocols are to attacks that scale well. For this purpose, we introduce the notion of *scalability of attacks*. We give the adversary access to an oracle which can break common cryptographic building blocks and assumptions and analyze how many inputs of a (multiparty computation) protocol they can learn or manipulate for each oracle access. The more inputs are affected, the more susceptible the protocol is to attacks that scale well. We compare several pairs of protocols solving the same problem in different ways in three examples and analyze the scalability of attacks on each protocol. We find that some protocols have a fatal breakdown, i.e. all inputs are affected with only one access to the oracle, while other protocols scale linearly or have a threshold, where the number of affected inputs increases drastically from one access to the other. Our framework provides strong arguments in favoring one voting scheme over another. It enables voting authorities to compare schemes that appear equally secure at first glance, and to consider the scalability of attacks when deciding on a scheme.

**Keywords:** Scalability · Risk Assessment · Quantification of Security · E-Voting

## 1 Introduction

Electronic voting schemes are often criticized for being insecure. One point that is mentioned frequently is that if an adversary can hack a voting server, they can manipulate all votes at once. Therefore, an attack on an electronic voting scheme has better scalability than an attack on a paper-based voting scheme. Others argue that electronic voting schemes provide cryptographic proofs of correctness


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 90–106, 2026.
https://doi.org/10.1007/978-3-032-05036-6_6

which are virtually impossible to forge, nullifying this scalability point of the discussion.

The security of electronic voting schemes and other cryptographic protocols is always based on assumptions. These assumptions are based on presumably (unproven) difficult mathematical problems (e.g., integer factorization) or trust assumptions (e.g., the majority of all participants is honest) among others. If a single assumption is no longer met once, cryptographers generally consider the entire protocol to be insecure. Two different protocols with unmet assumptions are then considered as being equally insecure, as in, both protocols are 100% insecure. In this paper, we analyze what happens if some assumptions are no longer met only in a few instances, e.g., when a state-sponsored adversary can find a collision in a weak cryptographic hash function at great expense and within a month's time (using a lot of conventional computers or a quantum computer). Our goal is to quantify whether the adversary can manipulate the outcome of an entire election or only manipulate the vote of a single voter. When comparing this scenario to another cryptographic protocol, where one can easily find collisions for all used hashes, we hypothesize that the former protocol is more resilient against widespread attacks than the latter.

### 1.1 Our Contribution

Our contributions are the following:

- We introduce the notion of *scalability of attacks*. Intuitively, the scalability describes how the adversary's cost and the damage of the attack evolve with multiple executions.
- We give a formal definition of scalability using oracles that can be used by the adversary to break assumptions.
- We compare the scalability of attacks on the confidentiality and integrity of several exemplary cryptographic protocols, e.g., voting schemes.

### 1.2 Related Work

In the following, an overview of related work regarding the notion of scalability of attacks and related cryptographic topics is given.

In 2010, Herley [4] introduced a distinction between scalable and non-scalable attacks. He considers an attack to be scalable, if the adversary's cost grows more slowly than linearly in the number of affected instances, or in his case, internet users. However, he considers the personalization of scalable attacks, e.g., through the personalization of phishing emails, not to be profitable, which might no longer hold true in times of artificial intelligence. Furthermore, we consider the term scalability to have more nuances than linear vs. faster growth of the adversary's cost, some of which we present in Table 1 and Sect. 4.

Another approach by Lopuhaä-Zwakenberg and Stoelinga [6] discussing the relation of the adversary's cost with the damage caused by an attack is using

attack trees. However, this approach lacks an analysis of how the cost evolves after multiple executions of an attack.

Next to these explicit studies of the scalability of attacks, there are related topics in cryptography. While in a classical black-box adversarial model, the adversary does not learn intermediate results or other local data of parties involved in a cryptographic protocol, *leakage-resilient cryptography* focuses on protocols that remain secure even if some local, secret data is communicated to the adversary by a side-channel or other vulnerabilities [5]. In a similar fashion, our work looks at the consequences of some secret data being known by the adversary.

*Quantum annoying* protocols are protocols that can be broken by quantum computers but require additional effort like solving discrete logarithms. So far, this property has only been defined for password-authenticated key exchange protocols (PAKEs) [3,10], while a broader definition applicable to other types of protocols is expected to be published soon. A PAKE is quantum-annoying, if a classical adversary has to solve one discrete logarithm for each password guess when breaking the protocol. To test the quantum annoying property of a protocol, the adversary has access to a discrete logarithm oracle, similar to the oracle we define for scalability in this paper. Both properties consider attacks in the context of broken assumptions and evaluate the security that is left. Therefore, results concerning one notion can be translated to the other: If a PAKE is quantum annoying, attacks on the protocol scale badly, since the oracle needs to be queried for every password.

The rest of the paper is structured as follows: Sect. 2 introduces preliminaries. The formal definition including oracles that break (cryptographic or other) assumptions is given in Sect. 3. Section 4 presents three exemplary protocol comparisons regarding the scalability of attacks on their confidentiality and integrity. Section 5 concludes the paper.

## 2 Preliminaries

Before a formal definition of attack scalability can be given, this section presents the necessary preliminaries and assumptions. The focus of this paper lies on the scalability of attacks on (cryptographic) protocols with multiple parties participating in the protocol execution. The goal of the adversary is to either learn some new information about the parties' inputs or to manipulate the output of the protocol without being detected. This corresponds to attacks on the confidentiality and integrity of the protocol, respectively. While we do not consider other security properties like availability in our analysis, the scalability of attacks on those properties can be considered in a similar manner.

The scalability of an attack quantifies how the adversary's cost and the damage of the attack relate to one another when the protocol is attacked multiple times. To determine the cost, we introduce an oracle. This oracle is capable of breaking various assumptions that the security of the protocol relies on. The

adversary is allowed to access this oracle before, during or after the execution of the protocol and the cost of the attack is measured in oracle accesses. Furthermore, the adversary may be one of the parties carrying out the protocol. In order to compare different protocols, we need to determine how many units, e.g., secret inputs are affected per oracle access for both protocols. This corresponds to the damage of an attack. This number of affected units per oracle access can be a constant value or scale differently, e.g., linearly, exponentially, in a logarithmic manner etc. We show example protocols with different attack scalabilities in Sect. 4.

Plotting the number of oracle accesses against the number of affected units results in a scaling curve. All scaling curves are weakly monotonically increasing, because additional oracle accesses can never decrease the information gained or damage caused by the adversary.

The scalability of an attack on a specific protocol does not have to be a continuous function. If, for example, the first $k$ oracle accesses do not provide any usable information to the adversary, but with the information gained from a $(k+1)$th oracle access the adversary learns all inputs, then we get a discontinuous function which jumps from one constant to another. Alternatively, the curve could grow linearly for the first $k$ oracle accesses and then begin to grow quadratically for each additional oracle access.

In this paper, we look at assumptions that might be wrong, e.g., based on unproven mathematical assumptions or human error, as well as assumptions based on unconditional or information theoretical security. While the latter would be impossible to break using a computationally bounded oracle, we extend the notion of using an oracle to actions like breaking in and stealing a secret. We also disregard other attacks that might be possible without an oracle while having a similar impact and therefore be the greater risk in the real world. Further, we do not estimate how much effort it would be to build or run such an oracle, we just assume the existence of one.

We do not compare the severity of the attack made possible by the oracle, other than the number of affected units. For example, we do not determine whether learning the vote of a single voter is better or worse than changing a single vote to a desired candidate or learning which candidate a specific voter certainly did not vote for. We justify this decision as follows: In the United States, about a third of the voters disclose openly which candidate they are voting for [7] but especially in swing states, every vote counts. Therefore, integrity seems to be an important security property for US elections. Meanwhile, Germany has a 5% electoral threshold for its federal election and no swing states, making the confidentiality of the ballot possibly more important than the integrity of a few votes which do not change the outcome of the election anyway. Which protocol is better with the same scalability of attacks therefore also depends on the voters. Accordingly, we analyze attacks on both the confidentiality and the integrity of protocols separately.

## 3 Formal Definition

In this section we formally define the scalability of attacks. Let $\mathcal{O}$ be an oracle that can break any computationally secure cryptographic building block, e.g., performing integer factorization, finding a discrete logarithm for any group, extracting a private key from a public key, finding a pre-image for a given hash or generating the required randomness to unveil a computationally binding commitment to any desired message. We also allow the oracle to break unconditionally or information-theoretically secure cryptographic building blocks like decrypting a one-time-pad or unveiling a perfectly hiding commitment scheme.

Let $I \in \mathbb{N}_0$ be the number of inputs or pieces of sensitive data, e.g., a voter's vote or inputs for a multiparty computation. Let $k \in \mathbb{N}_0$ be how many times the adversary accesses the oracle $\mathcal{O}$ at any point.

We introduce the notion of attack scalability of a security property of a protocol, i.e. how well attacks on the property scale when some assumptions are no longer fulfilled. Let

$$\begin{aligned} S : \mathbb{N}_0 &\to \mathbb{N}_0 \\ k &\mapsto (i)_{0 \leq i \leq I} \end{aligned}$$

be the scalability of attacks on a protocol using the oracle $\mathcal{O}$ $k$ times.

We define attacking the confidentiality as learning at least one bit of new information about an input before (e.g., during the setup), during or after the execution of the protocol. We define attacking the integrity as changing at least one bit of an input without being detected until the execution of the protocol is finished or changing at least one bit of an input which cannot be recovered.

We use indices to denote the respective protocol and attack dimension, e.g., $S^{\text{Integrity}}_{\text{BingoVoting}}$ denotes the scalability of attacks on the integrity of the Bingo Voting protocol. We also define $S = \max(S^{\text{Integrity}}, S^{\text{Confidentiality}})$.

The higher $S$ is, the better attacks scale. In this context, "good" scalability means good in the eyes of the adversary. If an attack scales well, the adversary only needs to put a small amount of effort into their attack to affect many inputs. A person choosing a protocol for implementation should therefore always aim for the scheme with the worse scalability. However, as the following section shows, which scheme has the worse attack scalability is not always obvious.

**Table 1.** Examples of the scalability of attacks depending on the number of oracle accesses $k$ given an oracle $\mathcal{O}$

| | |
|---|---|
| $S(1) = I$ | Accessing $\mathcal{O}$ once, the adversary can affect (manipulate or learn) all inputs. |
| $S(k) = \lfloor \frac{k}{2} \rfloor$ | For every other oracle access, the adversary can affect an input. |
| $S(k) = \begin{cases} 0 & k < 5 \\ I & \text{otherwise} \end{cases}$ | Until accessing the oracle five times, the adversary cannot affect a single input. Once they reach the threshold of five accesses, they can affect all inputs. |

Table 1 shows some example values for scalability. As mentioned in the previous section, $S(k) \geq S(k'), k > k'$ holds.

## 4 Examples

In this section we show three examples to demonstrate different scalabilities of attacks. In each example, we compare two protocols solving the same problem or very similar problems in different ways and show the impact of an oracle on the confidentiality and the integrity of the respective protocol. We demonstrate the best known (to us) attacks for each protocol. Future research may uncover new attacks on the protocols shown and not shown here and therefore change the scalability of (known) attacks, similar to the discovery of new attacks on (symmetric) encryption schemes.

### 4.1 Public Key Infrastructure Vs. Web of Trust

First, we compare the scalability of attacks on a certificate authority (CA) in a public key infrastructure (PKI) with a web of trust (WoT).

A certificate authority uses their private key in order to sign certificates which map a public key to a website, person or institution. The public key can belong to another CA which results in a tree of signed certificates, with a root CA at the top. People trusting the root CA can use the verified chain of signed certificates in order to trust a certificate signed by a lower CA. Due to the high number of people trusting the root CA, keeping its private key secret is an important task. If an adversary obtains the private key of a root CA (e.g., by accessing the oracle once), they can affect a large number of people.

In a web of trust, each participant signs the public keys of other participants they have met after personally verifying their identity. If a participant wants to contact a participant they have never encountered, they can check whether there is a transitive connection to the target starting with trusted and verified peers. The trustworthiness of each public key and other thresholds can be parametrized and used to limit the number of hops and therefore the number of public keys that are considered authentic.

In this example, we chose $I$ to be the number of people that want to securely communicate with each other. More precisely, we look at the following workflow that the adversary aims to attack:

1. Alice wants to send a message $m$ to Bob.
2. Alice visits a public key server and searches for the public key of Bob.
3. The public key server sends Alice Bob's public key $pk_{\text{Bob}}$ as well as a certificate $C_{\text{Bob}} = (pk_{\text{Bob}}, \text{Bob}, \{\sigma_1, ..., \sigma_n\})$ which includes Bob's name and a list of signatures $\sigma_i$, $i = 1, ..., n$, which show that this public key belongs to Bob.
4. Alice verifies the signatures of $C_{\text{Bob}}$ and aborts if there are too few valid signatures. Otherwise, Alice continues.

5. Alice generates a symmetric key $K_s$ and encrypts the message $m$ using $K_s$. She then encrypts the symmetric key $K_s$ using Bob's public key $pk_{\text{Bob}}$ and signs the message using her private key $sk_{\text{Alice}}$. Alice then sends the signed, encrypted message, the encrypted symmetric key, and a certificate $C_{\text{Alice}} = (pk_{\text{Alice}}, \text{Alice}, \{\sigma_1^*, ..., \sigma_m^*\})$ to Bob (*hybrid encryption*).
6. Bob checks the validity of $C_{\text{Alice}}$ and the validity of the signature, then decrypts the symmetric key using his secret key $sk_{\text{Bob}}$ and then decrypts the encrypted message using the symmetric key $K_s$.

In order to verify the receiver's public key before sending a message, the users could establish a PKI or a WoT. In the case of a PKI, the set of signatures $\sigma_i$ in the certificates contains a single element (signed by the certificate authority). We allow the adversary to intercept messages that are sent in the network, because an adversary with access to an oracle could be considered powerful enough to control parts of the network.

**PKI.** To attack the confidentiality and integrity of Alice's message to Bob in a PKI, the adversary intercepts the message from the public key server to Alice and replaces Bob's public key with $pk_{\text{Adversary}}$.[1] Additionally, the adversary creates a new certificate and signs it (using the CA's secret key which they received from the oracle) which convincingly shows that $pk_{\text{Adversary}}$ belongs to Bob. Next, the adversary intercepts the message that Alice sends to Bob and decrypts it using $sk_{\text{Adversary}}$. The adversary can read the message $m$, manipulate it before encrypting and signing it using $sk_{\text{Adversary}}$ and create a new certificate $C'_{\text{Alice}} = (pk_{\text{Adversary}}, \text{Alice}, \sigma')$ that shows that $pk_{\text{Adversary}}$ belongs to Alice. After attaching the new certificate, they send the new message to Bob, who considers the message to be authentic. With all users trusting the certificates issued by the root CA we get $S_{\text{PKI}}(1) = S_{\text{PKI}}^{\text{Integrity}}(1) = S_{\text{PKI}}^{\text{Confidentiality}}(1) = I$.

**WoT.** If an adversary manages to get the private key of any participant of the web of trust (also by accessing the oracle once), they can affect all participants who strongly trust this participant and therefore indirectly some additional participants. The adversary can use this private key to issue certificates and make other participants trust in the authenticity of $pk_{\text{Adversary}}$, just as in the case of a PKI. However, it is unlikely that there is a single person in a web of trust that could affect a majority of the participants. Ulrich et al. [11] as well as Richters and Peixoto [8] analyzed the OpenPGP web of trust graph. According to them, authenticated communication in a WoT is only possible within strongly connected components. The largest strongly connected component in the OpenPGP WoT graph consisted of 39796 nodes, i.e. keys that have been signed or are used for signing other participants' keys, in 2009 [8]. The indegree of a node shows how many users confirm the authenticity of the respective key by signing it. The highest indegree of a single node in the largest strongly

[1] For simplicity's sake, we assume the communication with the public key server to be unencrypted.

connected component was 965 [8]. By learning the private key of this node, the adversary could directly affect 965 participants, assuming these participants also put high trust in signatures of this participant and ignoring indirectly affected participants in order to get a rough estimate.[2] The average indegree was 7.58 [8], which is due to a large majority of the nodes having a very low indegree of 1 or 2. Since the information about the network topology, including the best connected nodes is available at the keyservers, we assume that the adversary picks the nodes with the highest indegrees first. Therefore, with the first oracle access, the scalability results in $S_{\mathrm{WoT}}(1) = \frac{965}{39796} I \approx 0.024I$.

Figure 1 illustrates the portion of the participants affected by the first 100 oracle accesses. To get an upper bound on the scalability, we assume that the number of affected participants increases with every oracle access according to the indegree of the node chosen in this step, disregarding that some of the participants might already have been affected in previous attack steps. While an attack on the root CA in a PKI affects every certificate issued by it, the web of trust uses a decentralized approach which leads to an approximately logarithmic scalability behavior.

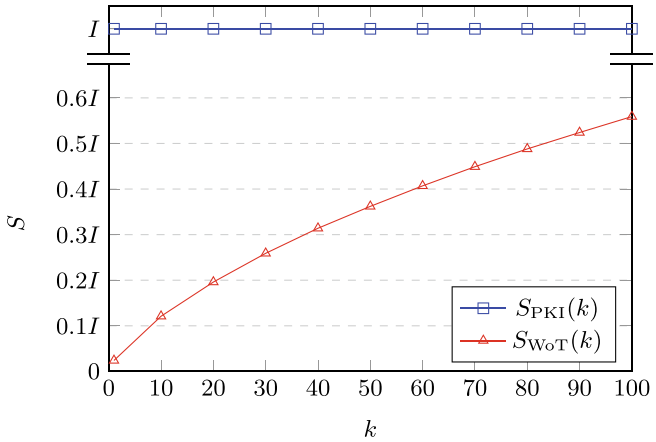


**Fig. 1.** Scalability of attacks on authentication infrastructures. The scaling curves for attacks on the confidentiality and integrity coincide for both the PKI and the WoT.

In this scenario, confidentiality and integrity are both compromised with the presented attack. By intercepting and decrypting messages encrypted with the adversary's public key $pk_{\mathrm{Adversary}}$, the adversary gets to read confidential messages. If they go even further by manipulating the message and attaching a forged signature, the integrity of the message is broken as well.

[2] We also treat each key as representing a different participant, which is a simplification since some participants register multiple keys. This affects the scalability by a small factor but does not undermine the argument.

Figure 1 considers the adversary not to be the CA or part of the WoT. However, it is not unlikely that the adversary participates in the WoT and that other participants have trust in their key. If this was the case, the curve should be shifted to the left by one oracle access, since the adversary knows their own private key from the beginning. Allowing the adversary to be the CA in a PKI, on the other hand, would instantly break the whole authentication infrastructure. Without accessing the oracle, the adversary would possess the CA's private key, making the scheme meaningless.

### 4.2 Mix-Net Voting Vs. Bingo Voting

We now compare two voting schemes: Mix-net Voting and Bingo Voting. We allow the adversary to be part of the voting authority, because an adversary with access to an oracle could be considered powerful enough to infiltrate the voting authority. We chose $I$ to be the number of votes. While the integrity of some votes may be irrelevant in a winner-take-all system, the manipulation of a single vote might be non-negligible, for example, if a candidate receives additional funding based on the number of votes they received in their last election.

**Mix-Net Voting.** Homomorphic encryption combined with re-encryption mix-nets can be used for anonymous voting [2]. Several distrusting parties jointly generate a secret key / public key pair for an ElGamal encryption scheme, so that no party knows the entire secret key, and publish the public key. Voters encrypt their vote using this public key and post the encrypted ballot on a public bulletin board. The encrypted ballots are then sent through a mix-net consisting of several mix-servers run by also distrusting parties. The mix-servers shuffle and rerandomize the votes before publishing them. Lastly, the secret key is recombined using enough of the key shares and the votes are decrypted and counted. In order to prove the correctness, each mix-server also publishes a proof that it did not alter the contents of any ballot.

To manipulate the tally, any mix-server can alter the content of a ballot and then forge the proof of correctness by using the oracle to solve the Decisional Diffie-Hellman-Problem in the Chaum-Pedersen Protocol used for the proofs. Therefore, $S^{\text{Integrity}}_{\text{Mix-netVoting}}(k) = k$.

Since the public key is public, a single oracle access is enough to learn the complete secret key, even though the secret key is shared among several distrusting parties. The adversary can then decrypt all public votes on the bulletin board and learn the vote of all voters before the ballots are shuffled. Therefore, $S^{\text{Confidentiality}}_{\text{Mix-netVoting}}(1) = I$ and therefore $S_{\text{Mix-netVoting}}(1) = I$.

**Bingo Voting.** We briefly explain the Bingo Voting protocol by Bohli et al. [1]. Bingo Voting consists of three phases:

**Phase 1: Pre-voting Phase.** Let $l$ denote the number of candidates. For every candidate $P_i$, $n$ random numbers are generated, where $n$ is the number of eligible voters. They are combined into tuples $(N^i_j, P_i)$ and act as dummy

votes for the upcoming voting phase. Unconditionally hiding commitments to the tuples are created using fresh randomness for each commitment and published on a public bulletin board. Furthermore, the voting authority proves that the dummy votes are equally distributed among all candidates.

**Phase 2: Voting Phase.** The voter presses the respective button on the voting machine in order to vote for their preferred candidate. A trusted random number generator generates a new number (which is indistinguishable from the random numbers used for the dummy votes), displays it to the voter and transfers it to the voting machine, which assigns it to the chosen candidate. For each other candidate, the voting machine picks a dummy vote from the pool of dummy votes for this candidate. Then, the voting machine prints a receipt for the voter, which lists all candidates next to a random number. The voter verifies that the number next to their chosen candidate matches the number on the display of the random number generator.

**Phase 3: Post-voting Phase.** After counting all votes, the voting authority publishes the following information:

- The final result of the tally.
- All receipts printed by the voting machine during the voting phase.
- All unused commitments with their respective unveil-information. These are dummy votes that were not used because a voter voted for this candidate or because of voter abstention.
- Proofs of correctness, e.g., for each receipt, prove that it contains a proper dummy vote for each candidate except for the selected candidate.

In order to prove the correctness, the voting authority performs the following steps for each ballot:

1. A new tuple with the chosen candidate and the random value from the random number generator is created.
2. A commitment to this new tuple is published together with the $l - 1$ commitments to the dummy votes of the ballot is computed. Therefore, $l$ commitments are published, one of them is a new commitment and all other commitments can be found in the list of all dummy votes. This list is called $C_{left}$.
3. $C_{left}$ is rerandomized, shuffled and published as $C_{middle}$.
4. $C_{middle}$ is rerandomized, shuffled and published as $C_{right}$.
5. All commitments in $C_{right}$ are unveiled. Therefore, everyone can verify that the contents of the commitments are exactly the contents of the receipt, including the actual fresh random number (indistinguishable from the rest). The only difference is the order of the candidates, due to the shuffle.
6. Using an (external) coin toss, either the random value from rerandomizing and shuffling to get from $C_{left}$ to $C_{right}$ or from $C_{middle}$ to $C_{right}$ is revealed (the voting authority may not predict the coin toss). If the voting authority tries to cheat, they will be detected with a probability of 50% per manipulated proof. Since the association between $C_{middle}$ and the third, unused list is not revealed, the association between $C_{left}$ and $C_{right}$ stays hidden.

If the adversary is part of the voting authority and uses the oracle to break the binding property of the commitments, they can manipulate the tally: Suppose they want candidate $A$ to win the election. When a voter votes for a candidate other than $A$, instead of using a dummy ballot meant for $A$, they pick a dummy ballot meant for the chosen candidate. When unveiling, they use the oracle to generate randomness which unveils the commitment to the value they wish. Therefore, for every oracle access, they can manipulate a vote without being detected, resulting in $S^{Integrity}_{\text{BingoVoting}}(k) = k$.

If the adversary uses the oracle to unveil a commitment, they can compare the unveiled random number from the commitment with the random numbers on voters' ballots and potentially find out which candidate a voter did not vote for, which also leads to $S^{Confidentiality}_{\text{BingoVoting}}(k) = k$. Therefore, $S_{\text{BingoVoting}}(k) = k$.

Figure 2 illustrates the scalabilities of attacks on the two voting schemes. While attacks on the integrity of both schemes scale linearly, there is a big difference regarding attacks on the confidentiality. The secret key used to decrypt the votes in Mix-net Voting is a single point of failure of the entire scheme, if an adversary is powerful enough to compute it from the public key. Bingo Voting, on the other hand, does not have such a vulnerability and attacks on the confidentiality scale linearly as well. Attacks on the Mix-net Voting protocol therefore have a better scalability than attacks on the Bingo Voting protocol.

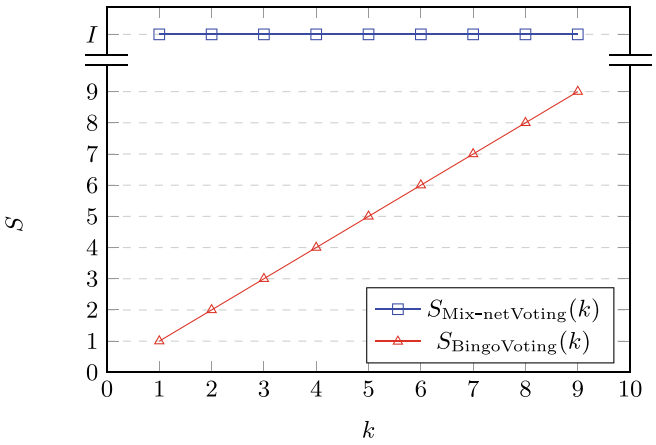


**Fig. 2.** Scalability of attacks on voting schemes. For Mix-net Voting, the scalability of the attack on the confidentiality is displayed, which is also the overall (maximum) scalability of the scheme. For Bingo Voting, the scalability of attacks on the confidentiality and integrity coincide.

### 4.3 Secret Sharing

In this section, different ways of storing a secret on one or multiple servers are compared. This secret can later be retrieved, e.g., to use it as a backup if the local

copy of the secret is lost. The secret can be a decryption key, for example. The data to be protected is the secret itself. Since we also consider partial leakage of the secret, we divide the secret into information units, which are the substrings that the secret comprises. For the following examples, the secret is assumed to be divided into five substrings and therefore, $I = 5$.

**One Server.** The trivial option for sharing a secret is to store it on a server unencrypted. In this case, $\mathcal{O}$ can be understood as an adversary, e.g., a secret agent, stealing the secret from the server. Here, $S^{\text{Confidentiality}}_{\text{OneServer}}(1) = I$, since the entire secret can be learned with only one oracle access.

The same holds, if the adversary wants to target the integrity of the secret instead of the confidentiality. With one oracle access, the secret can be changed at the server, therefore $S^{\text{Integrity}}_{\text{OneServer}}(1) = I$ and $S_{\text{OneServer}}(1) = I$. We assume that the manipulation of the secret is not detected until after the protocol execution.[3]

**Substrings.** Another option is to divide the secret into substrings and send them to several servers. The substrings correspond to the five information units that make up $I$. To guarantee being able to retrieve the secret even if some of the servers are unresponsive, each share is sent to two servers. Since the secret is divided into five substrings, where each is shared with two servers, this leads to ten servers used in total. Since the adversary does not know which servers store the same shares and which ones could give them new information, the adversary has to guess the shares they want to get from the oracle. $S^{\text{Confidentiality}}_{\text{Substrings}}(1) = 1$ for the first oracle access, since one substring, i.e., one information unit of the secret is learned, independent of the server that was attacked. $\mathcal{O}$ is the same oracle as in the single server case: It can be used by the adversary to steal the secret or a share of the secret from one server at a time. For the second oracle access, with a probability of $\frac{1}{9}$, nothing new is learned, because the server with the same share as the first one was chosen. With a probability of $\frac{8}{9}$, however, a new share containing one substring of the secret is discovered. This leads to $E[S^{\text{Confidentiality}}_{\text{Substrings}}(2)] = 1 + \frac{1}{9} \cdot 0 + \frac{8}{9} \cdot 1 = \frac{17}{9}$. Figure 3 shows the expected portion of the secret learned after every oracle access.

If the adversary's goal is not to learn the secret but to change the secret, the attack scales differently. If the integrity of the secret is targeted, the adversary can use the oracle to manipulate the shares stored at the servers. However, the adversary does not know which servers will be queried later to reconstruct the secret and which ones therefore have an impact on the integrity of the whole secret if their shares are manipulated. With every oracle access, one server can be attacked. This results in $E[S^{\text{Integrity}}_{\text{Substrings}}(k)] = 0.5 \cdot k$, since the manipulated

[3] At the end of the protocol execution, the manipulated secret should be indistinguishable from the real one. However, if e.g., the secret was a decryption key, the decryption with the manipulated key might either fail or lead to a different plaintext. In this case, mechanisms could be implemented to recover the real secret, which are out of the scope of this work.

share is only chosen with probability $\frac{1}{2}$ when reconstructing the secret and only one substring of the secret can be manipulated at a time. If one substring of the secret is manipulated, it would still be possible to recover the remaining four substrings of the secret by the end of the protocol.

Looking at the second oracle access, there are four possibilities: With the first oracle access, either a share was picked that will be used when reconstructing the secret or one that will not be used. If the share will be used, the second oracle access can either lead to an additional share that will also be used for reconstruction with probability $\frac{4}{9}$, or one that will not be used with probability $\frac{5}{9}$. The same happens if the first pick was a miss: With the next oracle access, the adversary can pick a share that will be selected for reconstruction and therefore have an impact on the integrity of the secret with probability $\frac{5}{9}$, or one that will not be selected with probability $\frac{4}{9}$. This leads to a scalability of $E[S^{\text{Integrity}}_{\text{Substrings}}(2)] = \frac{1}{2} + \frac{1}{2} \cdot \frac{4}{9} \cdot 1 + \frac{1}{2} \cdot \frac{5}{9} \cdot 1 = 1$ in the second oracle access. For the following oracle accesses, a similar behavior can be observed. The resulting linear curve is shown in Fig. 3.

**Shamir's Secret Sharing.** A more advanced approach is the well-known secret sharing scheme by Shamir [9]. The idea of Shamir's secret sharing is to send shares of the secret to $n$ servers and only $t$, $t \leq n$ servers are necessary to reconstruct the secret. While there was a part of the secret leaked to every server in the previous approach, any $t-1$ servers working together cannot learn anything about the secret when using Shamir's secret sharing. With this approach, the shares do not correspond to information units anymore. Therefore, the adversary must steal the shares of $t$ servers to learn the secret. However, in this approach, it does not matter which servers the adversary steals shares from, since the shares all differ from each other and any $t$ shares are enough to reconstruct the secret. The scalability can therefore be described as $S^{\text{Confidentiality}}_{\text{Shamir'sSecretSharing}}(k) = \begin{cases} 0 & k < t \\ I & \text{otherwise} \end{cases}$.

To compare the two secret sharing approaches using multiple servers in Fig. 3, we choose $n = 10$ and $t = 5$. Values of $t$ closer to $n$ might however be more appropriate in practice.

Targeting the integrity of the secret instead of its confidentiality, the adversary already succeeds with probability $\frac{1}{2}$ at the first oracle access. This results from choosing five out of the ten servers when reconstructing the secret. The compromised share is chosen with probability $\binom{9}{4}/\binom{10}{5} = 0.5$ and this results in $E[S^{\text{Integrity}}_{\text{Shamir'sSecretSharing}}(1)] = 0.5 \cdot I$. With Shamir's secret sharing scheme, the entire secret is compromised if one of the shares is manipulated. At the second oracle access, the probability that at least one of the compromised shares is used to reconstruct the secret is $(\binom{10}{5} - \binom{8}{5})/\binom{10}{5}$. Therefore, $E[S^{\text{Integrity}}_{\text{Shamir'sSecretSharing}}(2)] \approx 0.78 \cdot I$. Figure 3 shows the expected value of the scalability of the attack on the integrity of Shamir's secret sharing scheme.

The solid lines in Fig. 3 show that while the adversary does not learn anything about the secret until the fifth oracle query when using Shamir's secret sharing,

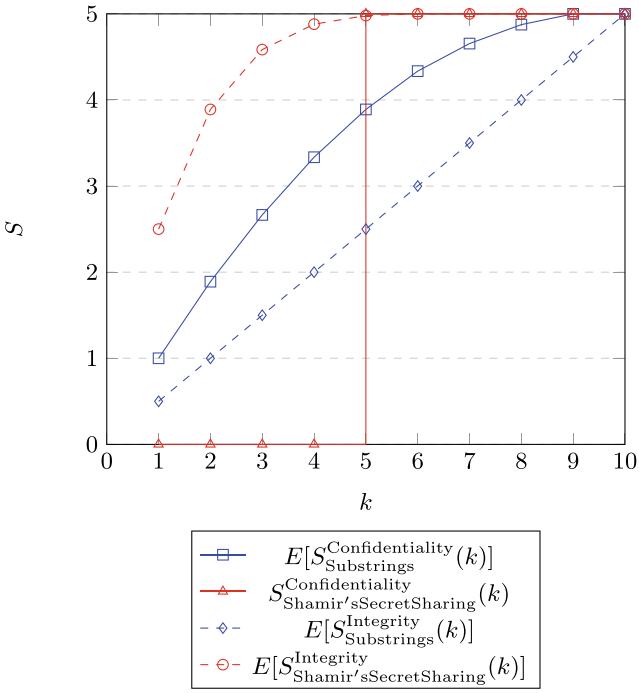


**Fig. 3.** Scalability of attacks on secret sharing schemes when using ten servers. Attacks on confidentiality as well as integrity are considered. For the substring approach and the integrity of Shamir's secret sharing scheme, expected values are displayed.

they are expected to learn between $\frac{3}{5}$ and $\frac{4}{5}$ of the secret with the same number of queries when substrings were sent to the servers instead. If the adversary is lucky and picks servers guarding different substrings with every oracle access, they might even be able to recover the entire secret within five queries. Depending on how valuable parts of the secret already are to the adversary, they might find the scalability of the substring approach favorable. The substrings might e.g., be meaningful secrets by themselves, or the secret might be a key and with every substring gained, brute-forcing becomes less complex. However, we assume that the final goal of the adversary is to learn the entire secret. Which scheme should be implemented therefore strongly depends on the application. In different scenarios, an adversary might prefer the scalability behavior of Shamir's secret sharing scheme, since they can be sure to know the whole secret after five oracle accesses. Here it should be noted, that if a higher value for $t$ relative to the number of servers $n$ is used, Shamir's secret sharing scheme provides stronger guarantees.

Comparing the scalability of the attacks on the integrity of the secret, the dashed lines in Fig. 3 show very different results. Since picking one compromised share when reconstructing the secret with Shamir's secret sharing scheme leads

to an entirely different secret, it scales much better from the first oracle access (in the adversary's view). Using the approach with the shared substrings, only parts of the secret can be manipulated with every access. Depending on the application, there might be a trade-off between the scalability of attacks on the confidentiality and the integrity of the schemes.

The plots in Fig. 3 refer to the case that the adversary is not the owner of any of the servers. If the adversary were to be one of the parties receiving shares, they would start with some information without having to access the oracle. Looking at the curves in Fig. 3, this would lead to them being shifted by one oracle access to the left.

## 5 Conclusion and Future Work

To sum up, we introduced the notion of scalability of attacks by giving a formal definition and comparing the scalability of attacks on the confidentiality and integrity of three different types of protocols. The notion of attack scalability can be used to quantify how the damage of an attack grows with multiple executions, also considering the adversary's cost. Since it is the adversary's goal to have a high impact with low costs, attacks that scale well in the adversary's view are more likely to occur. Investigating the scalability of attacks on a protocol can therefore reveal vulnerabilities and the notion presented in this paper is a valuable tool in risk assessment.

While our work confirmed that a single point of failure is a big disadvantage in protocols, the scalability of attacks is not a simple scalar value. When comparing two protocols, it might not even be directly clear which scalability is better. It depends on the application, the circumstances and the properties of the compared protocols. Voting authorities and other decision makers can use our work to compare contemplable protocols in their specific situation in order to make a more sophisticated decision.

In this paper, we focused on the integrity and confidentiality while giving the adversary a single oracle. For a more detailed comparison between two schemes, one could extend the notion of integrity to adding a ballot, removing a ballot, changing a ballot to a specific value or forcing the voter to vote for a random candidate. Confidentiality could also be further divided into learning a vote, learning which candidate a voter did certainly not vote for or which candidate they voted for with a specific probability. That way, one can better compare two schemes which both have linearly scaling attacks on their integrity at a first glance, but in reality, one is better protected against ballot stuffing. Denial-of-service attacks may be more difficult to formalize but could also be considered. Another dimension can be added by differentiating between oracles, e.g., one oracle that can only solve computational problems but cannot break information theoretically secure building blocks or another oracle that can attack supply chains (in order to replace a trusted random number generator with a predictable one).

**Acknowledgments.** This work was funded by the Topic Engineering Secure Systems of the Helmholtz Association (HGF) and supported by KASTEL Security Research Labs, Karlsruhe.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Bohli, J.M., Müller-Quade, J., Röhrich, S.: Bingo voting: secure and coercion-free voting using a trusted random number generator. In: E-Voting and Identity: First International Conference, VOTE-ID 2007, Bochum, Germany, October 4–5, 2007, Revised Selected Papers 1, pp. 111–124. Springer (2007)
2. Canetti, R., Hohenberger, S.: Special topics in cryptography, lecture 19: verifiable mix-net voting (2004)
3. Eaton, E., Stebila, D.: The quantum annoying property of password-authenticated key exchange protocols. In: Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20–22, 2021, Proceedings 12, pp. 154–173. Springer (2021)
4. Herley, C.: The plight of the targeted attacker in a world of scale. In: WEIS (2010)
5. Kalai, Y.T., Reyzin, L.: A survey of leakage-resilient cryptography. In: Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali, pp. 727–794. Association for Computing Machinery (2019)
6. Lopuhaä-Zwakenberg, M., Stoelinga, M.: Cost-damage analysis of attack trees. In: 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 545–558. IEEE (2023)
7. Maive, S.: Political campaign lawn sign survey - lawnstarter ranking (2024)
8. Richters, O., Peixoto, T.P.: Trust transitivity in social networks. PLoS ONE **6**(4), e18384 (2011)
9. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
10. Tiepelt, M., Eaton, E., Stebila, D.: Making an asymmetric PAKE quantum-annoying by hiding group elements. In: European Symposium on Research in Computer Security, pp. 168–188. Springer (2023)
11. Ulrich, A., Holz, R., Hauck, P., Carle, G.: Investigating the OpenPGP web of trust. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 489–507. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23822-2_27

# Development and Expert Evaluation of an Informative Video Concerning Verifiable Internet Voting

Tobias Hilt[1(✉)], Florian Moser[2], Philipp Matheis[1], and Melanie Volkamer[1]

[1] SECUSO, Karlsruhe Institute of Technology, Karlsruhe, Germany
{tobias.hilt,philipp.matheis,melanie.volkamer}@kit.edu
[2] Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France
florian.moser@inria.fr

**Abstract.** As digitalization advances, online elections are becoming increasingly prevalent. State-of-the-art internet voting systems implement verifiability, which allows to observe the election result to be correct, while safeguarding the secrecy of the election. However, the continued use of unverifiable 'black-box' systems suggests that election organizers may be unaware of the security challenges in internet voting and the mitigation strategies that have been developed.

To address this gap, we developed an informative video on the topic for election organizers who are non-experts in internet voting. To ensure that the simplifications made for our target audience do not lead to misunderstandings, 19 German-speaking internet voting experts evaluated the video. Based on their feedback, we consider improvements to the video to enhance its correctness, clarity, and completeness. Further, developing the video and then performing the expert evaluation provided valuable experiences and lessons learned we want to share with similar endeavours trying to simplify complex topics for non-expert audiences.

**Keywords:** Online Elections · Verifiable Internet Voting · Informational Video · Expert Evaluation

## 1 Introduction

Elections are increasingly held digitally, as can be, for example, observed for national elections in Estonia [8] or Switzerland [4]. In contrast to these examples, however, online elections often rely on so-called "black-box" systems, where the integrity and confidentiality of the voting process are entirely in the hands of a single system with no details public about its internal workings [17].

Using a black-box system presents risks, as potential manipulations or neglect of vote secrecy cannot be (dis)proven by an independent party. An attacker could, for example, compromise the election server from a remote location, altering votes undetected or breaching vote secrecy, which has been shown to be feasible [25]. This in turn may undermine trust in the electoral process and therefore


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 107–123, 2026.
https://doi.org/10.1007/978-3-032-05036-6_7

in democracy itself, as voters did not receive any concrete assurance that their votes are accurately recorded and counted and that their choices remain secret. Moreover, unsatisfied voters or malicious actors might claim electoral fraud, and neither the election organizers nor the system providers would be able to conclusively disprove such allegations.

To mitigate this risk, there are approaches that enable voters to verify that their votes have been cast as intended. In addition, election observers may verify that all cast votes are tallied correctly, notably while safeguarding vote secrecy. These approaches still require trust in some part of the system, but clearly disclose where this trust is required. This allows the election organizer to make an informed decision whether a given system is appropriate in their election context. Such systems are well-known in the academic literature (e.g., [2]), and some have also been developed by industry and are in active use (e.g., [9,21]).

The continued use of "black-box" systems [17] implies that many election organizers are not fully aware of the risks of internet voting, and at the same time, not aware of the different available approaches that aim to fix these issues. Lastly, they are likely unaware that there is no single perfect solution, but rather all approaches include trade-offs, and the election organizers must make an informed decision which approach is appropriate in their election setting.

*Our Contribution.* To introduce election organizers to internet voting, we developed an informational video on the subject. This video not only highlights the security challenges of internet voting, but also explains possible solutions for mitigating risks using a concrete internet voting system as a running example. It also communicates the open nature of the challenge, with each solution approach delivering their unique advantages, disadvantages and trust assumptions.

As we aim at a non-expert audience, and through the inherent limits of content transferable in a medium-length video, it is crucial to explain the topic in a simplified manner. However, simplification must not lead to incorrect statements, as this would risk misinforming the election organizers. Therefore, we evaluated the video with 19 German-speaking internet voting experts.

In this work, we document both the video and its evaluation results, but also the approach used. By documenting and evaluating the approach, we hope to support similar projects where an informational video aimed at a non-expert audience explains a complex topic in a simplified, but nonetheless correct, manner. By the production and evaluation of the video itself, we hope to contribute meaningfully to the secure advancement of electronic voting, particularly in German-speaking countries. The main contributions of this work are therefore:

- Development of a video about risks of internet voting, and approaches to mitigate that risk, popularized for non-expert election organizers.
- Evaluation of the correctness of this video by 19 German-speaking internet voting experts and the derivation of improvements to enhance the clarity.
- Documentation of the methodology used to develop a popularized video on a complex topic and the evaluation of the said video by experts.
- Lessons learned about the development and evaluation process and recommendations for future work.

## 2 Related Work

Previous work has already explored how e-voting systems can be made comprehensible to a broader audience, including potential voters and election organizers.

Some studies implicitly explain the characteristics of e-voting systems to their study participants. For example, Kulyk et al. investigated user interactions with successive versions of a voting system [13]. Although the primary goal was to assess usability and security trade-offs, participants gained an implicit understanding of these features, through their interactions and short explanations of the differences. Furthermore, studies on verifiability (e.g., [11,22,23]) use manipulated voting systems to help participants understand verification properties, although the primary focus is again not on education.

Further, some work also explicitly explains characteristics of e-voting systems, but again usually focuses on a concrete system or a concrete aspect of e-voting systems. Examples include research from Storer et al., which used videotaped material in a lab study to explain the mCESG voting system used in the UK [20], and Llewellyn et al., which provided oral explanations and illustrated figures to explain the "split mechanism" of the Prêt-à-Voter voting system [15]. Research by Gharadaghy and Volkamer offers detailed descriptions of various verifiability forms in e-voting systems using illustrated figures and text [5], while Schürmann et al. report on the use of e-learning modules and videos to improve the cybersecurity awareness of election organizers in the context of the 2019 European Parliament elections [19]. Moreover, short explanatory videos on concrete instances of e-voting systems produced by industry exist, e.g. about the system by Swiss Post[1] which is in use in Switzerland.

Our work extends upon these contributions by presenting a more holistic view of the security challenges of and potential solutions to mitigate risks in internet voting in general, focusing less on specific characteristics and concrete systems. While we also use a concrete system as a running example, we do not go into great detail on how it works. Instead, the video focuses on the general concepts which transform well to other systems.

## 3 Distributed Code Voting with Confirmation Codes

Here, we detail the voting system we took as a basis for the video. It is similar to existing proposals (e.g., [1,6]), while it remains unproven, which is however sufficient for our purposes.

*Setup Phase.* In the setup phase, the distributed servers agree on an encryption key $ek$ with a distributed decryption key (in spite the fact that decryption requires the involvement of all servers). The election encryption key $ek$ is sent to the printer. For each voter, the printer then selects for each candidate $c$ a random voting code $v_c$ and, per server $i$, a random partial confirmation code $r_c^i$.

[1] https://www.youtube.com/watch?v=j4X5iyIKhGg, last accessed 27.06.2025.

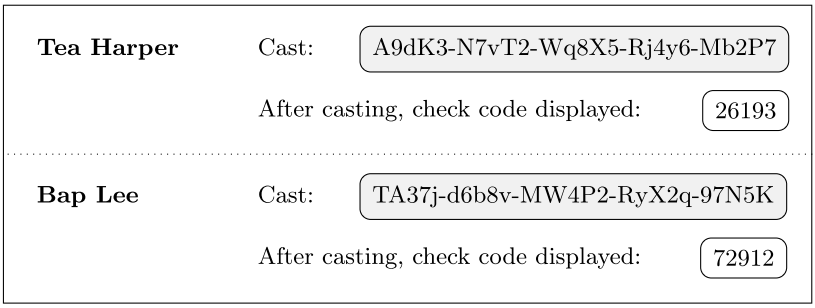


**Fig. 1.** Code sheet for a two-candidate election.

The printer then responds to each server $i$ for each candidate $c$ with an encryption of the candidate $e_c = enc(ek, c)$, a hash of the voting code $hash(v_c)$, as well as the respective partial confirmation code $r_c^i$. Furthermore, the printer prints the personal vote sheet of the voter, which for each candidate $c$ lists the corresponding voting code $v_c$, as well as the aggregation of all partial confirmation codes of that candidate $r_c = aggr(\forall i.r_c^i)$.

*Voting Phase.* When the voting period starts, the voter receives their personal code sheet (see fig. 1). On their voting device (e.g., laptop or phone), the voter opens the voting interface (e.g., webpage or app), and enters the voting code $v_c$ of the preferred candidate $c$ as printed on the code sheet. This voting code $v_c$ is sent to the servers, which confirm to each other using digital signatures that this is the first valid vote of that voter. Then, each server marks the corresponding encryption $e_c$ ready for tally and responds with the corresponding partial confirmation code $r_c^i$. The partial confirmation codes are aggregated, and the resulting confirmation code $r_c = aggr(\forall i.r_c^i)$ is displayed to the voter. The voter then checks whether the displayed confirmation code $r_c$ matches the confirmation code as printed on the code sheet, or otherwise complains to the authorities.

*Tally Phase.* After the voting phase, each server has a ciphertext for each voter who cast their vote, and corresponding signatures from all other servers. They collectively perform a verifiable shuffle (e.g., [24]), which transforms the ciphertext to destroy the link to the casting voter, while provably not altering its content. Afterwards, the servers collectively decrypt the list of transformed ciphertext, and then count the votes.

*Informal Security Statement.* Assuming the printer, the channel from the printer to the voter, and one of the servers are honest (but notably not the voter's device), this protocol guards the secrecy and the integrity of the votes. Intuitively, the voting code hides the selected candidate to achieve secrecy, and the confirmation code ensures that all servers have received the authenticated unmodified vote to achieve integrity. Finally, the verifiable shuffle and distributed

decryption guard the properties in the tally phase. Notably, some of the steps are verifiable by independent third parties (e.g., the proofs output by the verifiable shuffle).

## 4 Methodology

As described in Sect. 2, to the best of our knowledge, there are no existing endeavours that present a holistic view of the risks and the mitigation approaches to a non-expert audience. We therefore document in this section in detail how we approached the project, and the reasoning behind it. In Sect. 6, we discuss the strengths and weaknesses of this chosen approach.

Our approach is based on the principles of design science [10]. The primary objective is to inform election organizers about the security challenges associated with conducting elections online, outline existing methods to mitigate risks, and highlight the remaining issues. The main artifact we developed through this approach is an informational video created through iterative refinement. In line with the design science methodology, we then subjected the artifact to an evaluation. Section 4.1 and Sect. 4.2 provide further details on these components.

### 4.1 Video

We chose video as our medium because studies indicate that videos can efficiently communicate complex information by combining visual and auditory elements [18]. In addition, we decided to keep it at a length of ten minutes, which is within the recommended duration for informational videos [16].

*Defining the Story Structure.* The structure and story of the video will be highlighted shortly, but for a thorough inspection, we refer the interested reader to the video with English subtitles and translated script[2]. Based on previous experiences made while consulting election organizers on this topic, we decided that the overall theme of the video should follow an exploratory approach.

The story opens by motivating internet voting through parallels with other digitized services, such as online banking or appointment scheduling. It then highlights essential election requirements, such as fairness, vote secrecy, and vote integrity, alongside associated challenges and common misconceptions. Particular emphasis is placed on the significant security risks posed by voters using untrusted personal devices, given the numerous potential vulnerabilities across device types, operating systems, applications, and browsers (see Fig. 2).

After that, we present a proposal for an e-voting system that addresses these challenges and discuss its (dis)advantages. The technical details described in Sect. 3 are not part of the video, instead, the approach is explained more visually and in easier-to-understand terms (see Fig. 3).

The video further explains that alternative e-voting approaches exist, each potentially addressing certain challenges of the proposed system but introducing

[2] https://secuso.org/2025-07-E-Voting-Explanation-Video/.

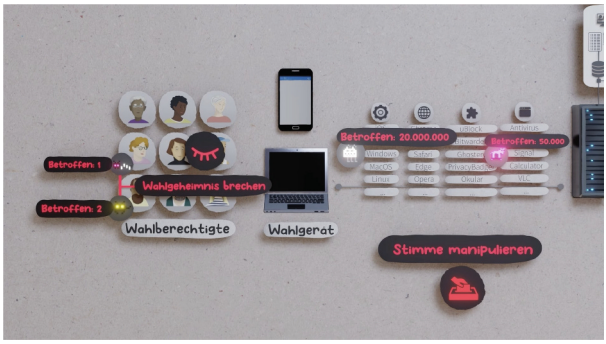


**Fig. 2.** The video visualizes the diversity of devices and software in use, which altogether pose a large attack surface. Malware might therefore be able to infiltrate some of these systems, and then break vote secrecy or manipulate the vote.

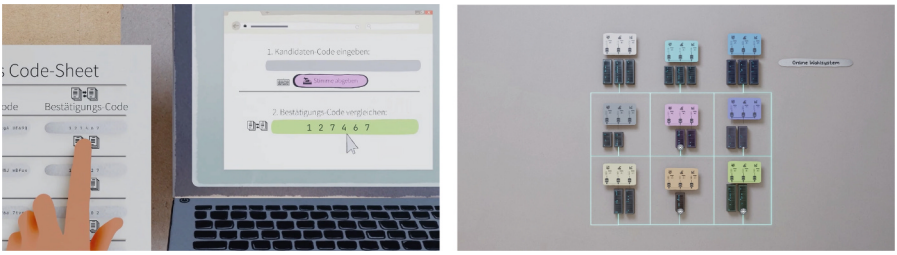


**Fig. 3.** The video explains how a voter verifies their confirmation codes, and how the servers communicate with each other to implement a digital "four-eyes principle".

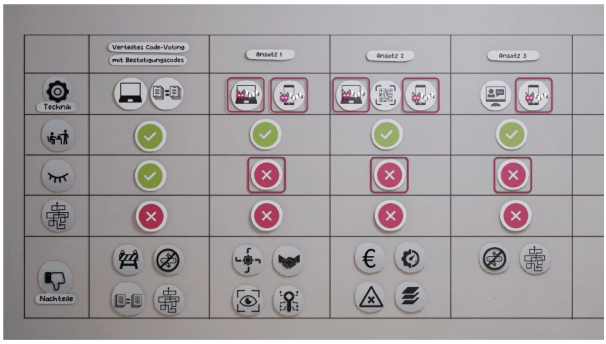


**Fig. 4.** The video uses a table to illustrate how different approaches exist, with each having individual characteristics and (dis)advantages. Notably, we discovered in the expert evaluation that the visualisation of this comparison is confusing, which we discuss in Sect. 5, and will improve in the next version of the video.

new issues or performing worse in other aspects. It emphasizes the need for continued research, as a single, perfect solution that resolves all challenges without creating new ones does not yet exist (see Fig. 4).

Finally, the story concludes that many open questions remain, for example, whether such a complex system would be fully trusted by voters.

*Choosing a Voting System.* Unfortunately, there is no clear default choice for which voting system to present: State-of-the-art systems are diverse in both the mechanism they employ, as well as the trust assumption they rely upon for their security (e.g., see [17]). To inform the election organizers accurately while at the same time respecting the limits of our medium, we needed a highly secure voting system while keeping its complexity in terms of mechanisms and trust assumptions to a minimum.

The approach we have chosen to present employs voting codes with confirmation codes, paired with distributed servers. This results in a secure voting system that needs only weak trust assumptions, notably none in the voting device or a single server. To cast a vote, a single interaction with the server is sufficient, and the computations and voter tasks are straightforward. Overall, our setting is derived from what is enforced by Swiss legislation for their national elections [3], and the system is similar to recent proposals employing code voting in the Swiss setting [1,6]. The system is described in detail in Sect. 3.

*Producing the Video.* In the beginning, the interdisciplinary research team, consisting of members with a background in cryptography and human factors, and the production team discussed a rough initial concept. This initial concept resulted in a draft of the textual script written by the interdisciplinary research team that was iteratively refined with the production team. After reaching a consensus on the final version of the textual script, the script was recorded. Subsequently, the production team developed a visual script based on the textual script. This visual script outlined in detail how the topic would be presented visually. Similar to the development of the textual script, the visual script underwent iterative revisions with input from the entire development team until a consensus was reached.

Finally, the production team proceeded to produce the video using the visual script and the recording of the textual script. The production phase was also followed by an iterative feedback process, with multiple iterations of the video produced until all parties were satisfied with the final result.

### 4.2 Expert Evaluation

Our intended target audience is election organizers who are in charge of important decisions such as whether an online channel will be offered for an election. However, through the limits in time and complexity that can be conveyed through our chosen short-video approach (see Sect. 4.1), we needed to simplify the topic. To avoid the risk that this simplification introduced inaccuracies, we chose to perform an expert evaluation first, to guarantee the correctness of the simplified content.

*Reaching Out to Experts.* We reached out to 18 German-speaking internet voting experts directly via email. Additionally, we contacted the email distributor for internet voting offered by the Federal Office for Information Security (BSI[3]). We believe that the selection considered most internet voting experts in the research community who understand German.

The invitation mail contained the following questions:

- Is the subject matter presented in the video presented correctly according to your understanding?
- Are there any aspects of the video that are misrepresented or could be misleading?
- What did you (not) like about the video?

With the invitation email we offered a two-week period to respond. Experts could participate by simply answering the email or by filling out the anonymous online questionnaire (linked in the invitation email). A reminder email was sent one week later to encourage participation. In total, we have received 19 answers.

*Analysing the Responses.* The responses we received were forwarded to a different member of the research team who merged and randomized the responses to ensure anonymity and avoid bias in the analysis. The feedback analysis then broke down the experts' responses into their statements which were categorized into three categories by two members of the research team:

- *Issues to Address* contained all the comments in which experts identified potential inaccuracies or misleading aspects of the video. This category highlighted areas that require further review or clarification.
- *Suggestions* captured recommendations for changes that did not necessarily point to inaccuracies but offered ideas for improvements, such as adjustments in content or presentation.
- *General Feedback* included comments on what experts liked or disliked about the video, reflecting their personal preferences and biases. In particular, remarks about inaccuracies were excluded from this category.

Any disagreements on understanding and categorization were resolved through brief discussions among research team members. Subsequently, the *Issues to Address* category was further analysed by two team members. They sub-categorized the feedback into three specific areas: *Correctness*, *Completeness*, and *Clarity*. For each aspect under *Correctness*, the team discussed internally to determine agreement with feedback or different points of view, with these points detailed in Sect. 5.2. Feedback about *Completeness* and *Clarity* was also carefully reviewed and decisions were made on whether and how to address these points in a future revised version of the video.

[3] https://www.bsi.bund.de/DE/Home, Last accessed on 24.01.2025.

## 5 Expert Evaluation

In this Section, we present the analysis to our expert evaluation as described in Sect. 4. We start with the results collected in Sect. 5.1, which we discuss in Sect. 5.2. Finally, the limitations of this analysis are presented in Sect. 5.3.

### 5.1 Results

We received responses from a total of 19 experts. In general, the experts indicate that they enjoy the video, and often give useful suggestions to improve the video further. Nine experts note minor issues, while they express none of the issues as critical. In this evaluation, we focus on statements that we classify as "Issues to address". We received 29 statements in this category which were sub-categorized into nine statements concerning *Correctness*, 11 concerning *Clarity*, and nine concerning *Completeness* (see Table 1). Given the substantial number of responses, we focus on a representative subset of the feedback that encompasses the most significant themes.

**Table 1.** Number of statements that were classified as "Issues to address".

| Category | |
|---|---|
| Correctness | 9 |
| Clarity | 11 |
| Completeness | 9 |

**Correctness.** Some of the statements are related to the potential manipulation of votes by the servers, proposing attacks against the presented internet voting system:

- **Expert 4:** As long as only ONE provider is honest [...] and if a majority (not all) providers manipulate together, then they can enforce their (manipulated) ballots as the true ones [and decide] against the honest providers by majority vote. Or not?
- **Expert 12:** With regard to protection against manipulation, tallied-as-cast [...] is not covered at all, is it?

Other statements are related to authentication:

- **Expert 5:** The fact that the nPA [the German electronic identity card] is not used for general applications or services does not prevent it from being used for voting, does it?
- **Expert 5:** The person voting must still identify themselves to the election server, otherwise anyone can send any code and use brute force to cast the correct vote, which is why the statement that the identity of the person voting is not known by malware is not true.

Additionally, one expert's statement is related to costs; **Expert 18** "would not generally agree that specific voting devices are generally too complex and expensive", responding critically to a statement made in the video that argues that dedicated voting devices delivered to each voter are impractical due to the complexity and cost of such a solution.

**Clarity.** Most of the responses categorized into the *Clarity* category are related to the phrasing used in the video and often contain alternative phrasings proposed by the experts. Two different experts mention that the phrase "mathematical proof" might be misleading. For example, **Expert 3** writes that "When checking the verification codes, I would not speak of "mathematical proof", but rather of a strong indication that the vote must have arrived correctly."

Two experts highlight clarity issues related to the approach:

- **Expert 1:** It is claimed that [...] the vote would be counted later if only at least one of the participants in the online voting system works correctly. This is ultimately not wrong, but misleading, because, in fact, as soon as the participants in the online voting system no longer agree, 'the election is canceled' and no counting takes place. The system as such is therefore not robust.
- **Expert 19:** The candidate code is incredibly long and complicated. Or could you not use much shorter codes?

**Completeness.** With regards to *Completeness*, experts mentioned that we could add different aspects to the video to have a more holistic view. Four of the proposed additions are listed below:

- **Expert 11:** Not wrong or misleading, but one aspect was missing: denial of service.
- **Expert 12:** It should also be mentioned that the postal service provider must be trusted throughout the entire scenario.
- **Expert 19:** The voting machine must also be trusted in this approach to voting secrecy. A manipulated device could simply display: "Enter the name of your candidate".
- **Expert 19:** If you are already proposing approaches "without protection of voting secrecy" (as you call it) you must also mention the Swiss approach [...].

### 5.2 Discussion

We now discuss all the expert responses mentioned in Sect. 5.1, and give an outlook on how this may be addressed in the next version of the video.

**Correctness.** Given the fully-defined system we presented in Sect. 3, we observe that a single honest server is sufficient to guarantee the true cast ballots are tallied, and prevent even a majority of dishonest servers from proposing manipulated ballots. The reason is that the servers confirm to each other using digital signatures that some incoming vote is the first valid vote of that voter. Only if this agreement is successful will the servers then produce the return code. This aspect was not mentioned in the video to reduce the amount of concepts explained, but we will consider including it in the next version, possibly with a suitable simplification.

Responding to the feedback related to the authentication process, we agree that the nPA could, in principle, be used to authenticate electronic ballots, similar to how the eID is used in Estonia to authenticate [9], and we could mention this in the video. However, we note that the nPA is as of yet not broadly used in Germany [14]. Responding to the second feedback related to authentication, which suggests that the voting codes alone are insufficient to authenticate an eligible voter, we note that the voting codes visualized in the video are designed to hold more than 128bit of randomness[4]. This is considered to be secure against random guessing by current standards[5]. We plan to address this in a future version of the video by explicitly stating that the codes are chosen sufficiently long that they cannot be guessed. What we also have to acknowledge is that the phrasing at this point in the video is not very clear about identification, and indeed, as the expert noted, the casting voter can be identified: for example by malware on the voter's device, or by an implementation which may as a hardening measure require the voters to additionally login[6]. However, the secrecy of the votes is still ensured, as the voting code does not reveal the voter's choice, and the tally phase makes sure the encryptions related to the voter's choice are separated from the casting voter before decryption (see Sect. 3).

Regarding costs, we agree that it is hard to make absolute statements concerning which approaches are more expensive than others, as costs depend on several factors, such as the size of the electorate or the frequency of elections. As this is also not the core point of the video, instead of discussing the cost of different voting modalities (e.g., using the Estonian example [12]), we plan to address

[4] We chose the alphabet $A_{57}$ from [7], which holds 5.83 bit per character. Each voting code is 25 characters long, hence each holds 145.75 bits of information. The "spare" bits may be used for other purposes, e.g. validation or error correction.

[5] NIST considers 128bits secure until 2030 and beyond, see https://www.keylength.com/en/4/, last accessed 24.01.2025.

[6] This may help to defend against denial of service attempts, which flood a server with many (invalid) requests, until they are overwhelmed and can no longer respond.

this in a future version of the video by choosing a more open formulation, or not mentioning this point at all.

**Clarity.** It was suggested to avoid using the term "mathematical proof", but instead "strong indication", when describing the verification of the confirmation codes. Indeed, in a scientific publication, one would not use "mathematical proof", as this implies an absolute (true) result, whereas in the proposed system, even if the verification of the confirmation code is successful, the voting code may not reach the tally: For example, the adversary might have prevented the voting code from reaching the servers and guessed the confirmation code. However, this can be made very unlikely (to the extent of practical impossibility) by proper configuration of the system, for example, a sufficient length of the confirmation codes. In a future version of the video, we will strive for an alternative wording that finds a better balance between transporting the desired meaning intuitively, while also being factually correct.

We agree with the expert arguing that "the vote would be counted later if only at least one of the participants in the internet voting system works correctly [...] [is] misleading", as indeed in the design chosen, the election could be forced to be canceled by a single dishonest server, by it simply refusing to participate in the generation of confirmation codes or the decryption in the tally phase[7]. This phrase intended to focus on the strong security guarantees, mentally prefixed with "if the confirmation code is correct, and the election is tallied, then [...]". The next version of the video will communicate this more appropriately.

We further agree with the expert that the voting codes are very long. However, in our running example voting system, the voting codes also serve as the authentication mechanism, and therefore need to be sufficiently long to be hard to guess for the adversary. We could instead separate authentication from the voting codes and, by this, reach very short voting codes in the length of the number of candidates (a single digit would be sufficient for up to 10 candidates), as proposed in [1]. We will reevaluate in the next version of the video whether we can present an approach using shorter voting codes, or whether these suggestions are less appropriate for our purposes, which are strongly focused on simplicity of presentation.[8]

**Completeness.** Some experts notice that not all trust assumptions are explicitly stated in the video. Although a letter is stated to be delivered to the voter, it is not mentioned that this assumes the postal service to be honest. In a similar spirit, the video does not mention that voters are only protected if they use the voting system as intended (even if a malicious component tries to convince

[7] While such active dishonest behaviour is likely attributable, if the adversary does not fear or care about the consequences, it might still manifest.

[8] For example, mapping the fundamental ideal of [1] to our proposal would require the voter to enter two values, concretely one long authentication secret and one short voting code. This might substantially increase the textual and visual burden of the casting procedure in our informational video.

the voter to break protocol). Another expert mentions that relevant context to internet voting is omitted, specifically a discussion of denial of service attacks. While the video can not discuss all aspects of internet voting, due to its target audience and limited duration, we may reevaluate the relative importance of e.g. fully and precisely stating our trust model versus other mentioned aspects of internet voting.

When the video discusses the advantages and disadvantages of the presented voting system, a table visualizes potential trade-offs to other approaches. Based on the feedback received, we realize that the table visually (wrongly) transports the meaning that we compare to other actual approaches. However, our only goal was to argue that, overall, other valid approaches with different trade-offs exist, which we intend to visualize better in a future version of the video.

### 5.3 Limitations

This study has several limitations that should be acknowledged.

- Feedback from the experts may not fully address the practical concerns and informational needs of the intended audience – election organizers. We explicitly chose to evaluate the video with experts before sharing it with election organizers to identify any remaining errors in accuracy.
- The feedback from the experts is qualitative, relying primarily on descriptive assessments rather than quantitative metrics. This limits the possibility of generalizing the findings and their respective importance.
- The video is currently produced in German and was reviewed exclusively by German-speaking experts. This language limitation means that the video's applicability and usefulness have not yet been tested with experts from different linguistic backgrounds.
- The system presented in the video is not in active use. Rather, it has been chosen because of its simplistic design, which is useful for our didactic purposes. However, as such, the video may not fully capture the complexities and challenges of actual deployed verifiable internet voting systems.

## 6 Approach Evaluation & Future Work Recommendations

As described in Sect. 4, the approach applied in this study was based on design science principles, primarily because to the best of our knowledge, there is no comparable and documented endeavour. To support future projects that aim to simplify highly complex topics (which may have significant consequences if communicated incorrectly) to non-expert audiences, we are sharing our insights from this approach alongside the recommendations for future work.

Overall, the adopted approach was successful, as evidenced by the overwhelmingly positive response received for the artifact, and the actionable feedback to improve the video further. We reflect here on the crucial aspects of the approach and discuss possible improvements.

*Establish a Clear Basis Before Simplification:* At the outset, it was invested in clearly establishing what exact (real-world and therefore complex) scenario would serve as the foundation for the project. Having this well-defined base was crucial during the various stages of simplification. It allowed us to continuously refer back to the original scenario and assess whether the simplifications remained valid. This proved particularly beneficial in ensuring that the integrity of the simplified scenario was maintained throughout the project.

*Embrace Interdisciplinary Collaboration:* The interdisciplinary nature of our team was invaluable. By integrating insights from different fields, we were able to simplify complex details without sacrificing accuracy. Concretely, having both experts from cryptography and human factors involved, allowed us to balance necessary detail with understandability. The general success of this collaborative approach was reflected in the expert feedback, which did not highlight any major flaws in the simplifications.

*Be Aware of Challenges in Phrasing:* Throughout development, and during expert evaluations, we encountered the significant challenge of finding universally acceptable phrasing. To address this, we involved the video production team, who were non-experts, from the outset. Their input was invaluable in flagging phrases and concepts that the cryptography and human-factors experts assumed were self-evident but were not clear to a broader audience. Despite numerous iterations and alternative proposals, the e-voting experts we consulted continued to suggest revisions, underscoring how difficult it is to find phrasing that resonates across diverse perspectives and backgrounds.

*Seek Early Expert Review for Textual Script:* A key lesson we have learned is the importance of obtaining expert feedback on the video script *prior* to the production of the visuals. Much of the feedback we received was related to the textual content, rather than the visualizations. This suggests that involving experts earlier in the process could resolve potential issues before the time-consuming visual production begins. However, we note that some visuals also lead to actionable feedback from the experts: Concretely, the comparison table of voting systems was misunderstood to compare to concrete systems, instead of the intended meaning to highlight more generally that all approaches have advantages and disadvantages. This indicates that expert feedback and potential revisions after visual production cannot be fully avoided.

## 7 Conclusion

We produced an informative video through interdisciplinary collaboration, aimed at informing election organizers about the security challenges associated with conducting elections online, outlining existing methods to mitigate risks, and highlighting the remaining issues. We chose a short video as our medium and devoted considerable effort to simplifying and condensing the complex topic. To

ensure the factual correctness of the video, we sought feedback from German-speaking internet voting experts. The feedback is overall positive, and no major issues identified, while it is very helpful to refine the video further. With the final, to be produced, version of this video we hope to contribute to the further advancement of secure internet voting, especially in Germany.

We find that establishing a clear basis before simplification, which can be consulted when developing the presentation of the topic, helps to keep the simplifications accurate and consistent. Further, useful for the simplification is especially interdisciplinary collaboration, which enables one to continuously balance details with understandability. However, this can not fully resolve the challenges of phrasing, which seem to be best addressed by including a large number of reviewers of diverse backgrounds in the project. Finally, besides expert review of the final result, expert review of the textual script can uncover most of the areas for improvement before the time-consuming production of the visuals begins. By sharing these insights, we hope to offer guidance and a starting point for similar endeavours trying to simplify a complex topic for a non-expert audience, without the simplification jeopardizing the correctness of the explanations.

**Acknowledgement.** This work was supported by funding from the topic Engineering Secure Systems, subtopic 46.23.01 Methods for Engineering Secure Systems, of the Helmholtz Association (HGF) and by KASTEL Security Research Labs. Further, this work benefited from funding managed by the French National Research Agency under the France 2030 program with the reference ANR-22-PECY-0006. Finally, we would also like to thank MotionEnsemble (Lena Schall, Alexander Lehmann) for their valuable support in producing the video and helping to refine its content.

## References

1. Cortier, V., Debant, A., Moser, F.: Code voting: when simplicity meets security. In: Computer Security – ESORICS 2024, pp. 410–429. Springer, Cham (2024)
2. Cortier, V., Gaudry, P., Glondu, S.: Belenios: a simple private and verifiable electronic voting system, pp. 214–238. Springer, Cham (2019)
3. Federal Chancellery: Federal chancellery ordinance on electronic voting (2013). https://www.fedlex.admin.ch/eli/cc/2022/336/en
4. Germann, M., Serdült, U.: Internet voting and turnout: evidence from Switzerland. Elect. Stud. **47**, 1–12 (2017)
5. Gharadaghy, R., Volkamer, M.: Verifiability in electronic voting - explanations for non security experts. In: Krimmer, R., Grimm, R. (eds.) Electronic Voting 2010. LNI, vol. P-167, pp. 151–162. GI (2010)
6. Haenni, R., Koenig, R.E., Locher, P.: Private internet voting on untrusted voting devices. In: Financial Cryptography and Data Security. FC 2023 International Workshops, pp. 47–62. Springer, Cham (2024)
7. Haenni, R., Koenig, R.E., Locher, P., Dubuis, E.: CHVote protocol specification. Cryptology ePrint Archive, Paper 2017/325 (2017). https://eprint.iacr.org/2017/325
8. Heiberg, S., Willemson, J.: Verifiable internet voting in Estonia. In: 2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE), pp. 1–8 (2014)

9. Heiberg, S., Willemson, J.: Verifiable internet voting in Estonia. In: International Conference on Electronic Voting (EVOTE) (2014)
10. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004)
11. Hilt, T., Berens, B., Truderung, T., Udovychenko, M., Neumann, S., Volkamer, M.: Systematic user evaluation of a second device based cast-as-intended verifiability approach. In: Voting'24. Springer (2024)
12. Krimmer, R., Duenas-Cid, D., Krivonosova, I., Vinkel, P., Koitmae, A.: How much does an e-vote cost? Cost comparison per vote in multichannel elections in Estonia. In: Electronic Voting, pp. 117–131. Springer, Cham (2018)
13. Kulyk, O., Neumann, S., Budurushi, J., Volkamer, M.: Nothing comes for free: How much usability can you sacrifice for security? IEEE Secur. Priv. **15**(3), 24–29 (2017)
14. Liesbrock, P., Sneiders, E.: Assessing poor adoption of the Eid in Germany. In: Rocha, A., Adeli, H., Dzemyda, G., Moreira, F., Colla, V. (eds.) Information Systems and Technologies, pp. 292–301. Springer, Cham (2024)
15. Llewellyn, M., et al.: Testing voters' understanding of a security mechanism used in verifiable voting. In: 2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13). USENIX Association (2013)
16. Manasrah, A., Masoud, M., Jaradat, Y.: Short videos, or long videos? A study on the ideal video length in online learning. In: 2021 International Conference on Information Technology (ICIT), pp. 366–370 (2021)
17. Moser, F., Kirsten, M., Dörre, F.: SoK: mechanisms used in practice for verifiable internet voting. In: 9th International Joint Conference on Electronic Voting (E-Vote-ID 2024). LNI, Gesellschaft für Informatik (2024)
18. Noetel, M., et al.: Video improves learning in higher education: a systematic review. Rev. Educ. Res. **91**(2), 204–236 (2021)
19. Schürmann, C., Jensen, L.H., Sigbjörnsdóttir, R.M.: Effective cybersecurity awareness training for election officials. In: Electronic Voting, pp. 196–212. Springer, Cham (2020)
20. Storer, T., Little, L., Duncan, I.: An exploratory study of voter attitudes towards a pollsterless remote voting system. In: IaVoSS Workshop on Trustworthy Elections (WOTE 06) Pre-Proceedings, pp. 77–86. Citeseer (2006)
21. Swiss Post Ltd.: Swiss post voting system – system specification (version 1.4.1), Technical report, Swiss Post Ltd. (2024)
22. Thürwächter, P.T., Volkamer, M., Kulyk, O.: Individual verifiability with return codes: manipulation detection efficacy. In: Electronic Voting, pp. 139–156. Springer, Cham (2022)
23. Volkamer, M., Kulyk, O., Ludwig, J., Fuhrberg, N.: Increasing security without decreasing usability: a comparison of various verifiable voting systems. In: Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022), pp. 233–252. USENIX Association (2022)
24. Wikström, D.: A universally composable mix-net. In: Theory of Cryptography, pp. 317–335. Springer, Berlin, Heidelberg (2004)
25. Wolchok, S., Wustrow, E., Isabel, D., Halderman, J.A.: Attacking the Washington, D.C. internet voting system. In: Financial Cryptography and Data Security, pp. 114–128. Springer, Berlin, Heidelberg (2012)

# How to Implement Anywhere Voting: A Case Study of Brazil

Leonardo Kimura[1(✉)], Roberto Araújo[2], and Marcos Simplicio[1]

[1] University of Sao Paulo, São Paulo, Brazil
lkimura@larc.usp.br, mjunior@larc.usp.br
[2] Federal University of Para, Belém, Brazil

**Abstract.** High levels of abstention in elections are often caused by the distance to polling stations. This is particularly prominent for voters who are traveling or have recently relocated. A promising solution to this problem is "anywhere voting", which allows citizens to cast their votes at any polling station. However, existing implementations typically rely on Internet-connected authentication to avoid double voting. Albeit simple, this approach is often seen as impractical in many scenarios, especially in remote regions where Internet access is unreliable or when the risk of denial-of-service attacks is high. In this article, we study how anywhere voting could be adapted to such constrained environments. Using Brazil as a case study—given its vast territory, regional disparities, and infrastructural challenges—we evaluate four potential solutions. Our analysis suggests that the most viable approach involves preventing double voting through secure hardware, while eliminating residual duplicates via mixnets and threshold cryptography.

**Keywords:** in-person voting · electronic voting · secure voting · abstention

## 1 Introduction

Election turnout is essential for democratic legitimacy and representativeness, influencing both the perceived authority of elected governments and the overall political stability of a territory [12,19,22]. Nevertheless, turnout rates have been declining globally [13]. While a significant portion of this decline is often attributed to growing political apathy among citizens [17], various practical aspects also contribute to this issue – including compulsory voting, the complexity of the voting process, and the time required to cast a ballot [7]. These challenges highlight the need for strategies to promote and facilitate voter participation.

One of the key factors contributing to voter abstention is the distance to polling stations. In the Greek 2023 elections, for example, 31% of abstentions related to practical reasons were attributed to travel distance [18]. Similarly, in Canada, 9% of abstentions in 2022 were linked to voters being out of town, while


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 124–140, 2026.
https://doi.org/10.1007/978-3-032-05036-6_8

in Brazil this figure reached 46% in 2024 (almost 10% of the total population) [20,35]. Even though mail-in voting offers one potential solution for improved turnout, it is not suitable for areas with unreliable postal services or where the risk of voter coercion and vote-buying is high [4]. Strategies based on Internet voting face similar criticism, in particular related to vote-buying and the threat of malware [15], since votes are still cast in an unsupervised environment.

Arguably, a more promising alternative to mitigate such concerns is *anywhere voting*, which allows citizens to cast their votes at any polling station [30]. While it still requires in-person participation, this approach significantly improves accessibility – particularly for individuals who are traveling or have recently relocated. Indeed, implementations of anywhere voting mechanisms in parts of the United States and Canada have demonstrated significant increases in turnout following their adoption [8,29]. However, most existing implementations depend on Internet connectivity to authenticate voters and prevent double voting. Therefore, deploying such mechanisms in more constrained environments is a challenge, in particular in locations lacking reliable Internet access (e.g., in remote areas of the Amazon Rainforest). In addition, Internet connections may be vulnerable to cyberattacks, a non-negligible risk given the history of voter suppression attempts in many countries [6,28].

In this work, we explore how to implement anywhere voting in more restrictive environments – specifically, scenarios where constant Internet connectivity is unavailable or unreliable. We focus on Brazil as a case study, given its continental scale, large regional disparities, and infrastructural challenges. Specifically, Brazilian elections are among the largest in the world, with nearly 120 million voters casting their votes on a single day, under the coordination of a centralized electoral authority. However, the country also has a history of electoral fraud, including ballot stuffing, manipulated recounts, and explicit vote-buying practices [26,33]. Therefore, any proposed solution must be capable of operating across a wide range of geographic and socioeconomic conditions, while remaining accessible to a diverse electorate, many of whom have limited formal education. At the same time, it must offer strong security guarantees against both internal and external adversaries.

In this work, we discuss four potential solutions for enabling anywhere voting in Brazil: Internet-connected authentication, single-line voting, double-vote removal, and secure hardware. We evaluate each solution by examining its advantages, challenges, and overall feasibility, focusing on factors such as security, practicality, and implementation complexity. Finally, we identify the most promising approaches and outline directions for future research.

## 2 Background

In this section, we first examine how other countries typically implement anywhere voting. After that, we present the Brazilian Elections, beginning with a brief overview of their social and historical context. Then, we describe the current electoral process in Brazil, including the setup and distribution of voting machines, as well as procedures for voting and tallying.

### 2.1 Anywhere Voting in Other Countries

Anywhere voting is already deployed in some countries [30]. Even though specific implementation details vary in each country's voting procedures, two primary strategies are commonly adopted: (1) the use of internet-connected electronic poll books (EPB), and (2) mechanisms to detect and penalize double voting.

**Internet-Connected EPB.** The first and most straightforward approach for anywhere voting is to use Electronic Poll Books (EPBs) connected to the Internet [34]. In this scenario, voters who cast their ballots are recorded in an EPB, which is synchronized with other EPBs to prevent double voting. For example, in several U.S. states, EPBs typically consist of conventional laptops connected to the Internet [34], placed in the so-called "voting centers". These centers, larger than conventional polling stations (though fewer in number), are strategically located in areas with high visibility, such as near shopping centers and schools. The introduction of voting centers has led to a notable increase in voter turnout [29].

Albeit promising, Internet-connected EPBs introduce significant availability risks [11]. For instance, during elections in Los Angeles, technical issues with EPBs led to considerable delays [36]. As a result, reports typically recommend that internet-connected EPBs incorporate backup mechanisms, such as using multiple Internet providers or employing physical poll books [11]. While these measures may be sufficient in some countries, they are likely inadequate as the main solution for countries where infrastructure availability is very diverse, like Brazil. We explore these challenges in more detail in Section 3.1.

**Detect and Punish.** The second approach consists of identifying voters who cast multiple votes and defining a punitive treatment that is considered enough to dissuade potential attackers. In Australia, for example, electors who cast their votes have their names marked on a local electoral list. At the end of election day, all electoral lists are cross-verified, and voters suspected of casting multiple votes may face penalties, including monetary fines or imprisonment. Although this approach does not eliminate double votes from the final tally, it appears to be effective in countries where it is implemented. For example, in the 2022 Australian federal elections, only 0.013% of votes were identified as multiple votes (an average of 13 double votes per 100,000 voters), a negligible figure according to the Australian Electoral Authority [24].

However, the effectiveness of this approach in an adversarial environment like Brazil is questionable. First, one cannot dismiss the risk of attackers impersonating legitimate voters to avoid detection, as well as unfairly punishing honest voters. This is a concern in Brazil, as, despite the adoption of relatively strict authentication procedures—such as the presentation of official identification and the collection of fingerprints for biometric verification—impersonation attacks remain a risk [5]. Second, attackers could exploit vulnerable voters, deceiving them into casting multiple votes with the false promise of impunity. This type of misbehavior is likely to coincide with other forms of deception like vote-buying, which remains a significant issue in Brazil, particularly among illiterate

or poorly educated populations [23]. Finally, this solution only detects, rather than removes, multiple votes. This limitation can be especially problematic in smaller elections, where even a few fraudulent votes could significantly affect or invalidate the final results.

### 2.2 Brazilian Elections

Brazilian elections have a long history of electoral fraud. Before 1996, elections were conducted using paper ballots, which were cast in ballot boxes and manually counted later. This facilitated widespread fraudulent practices, such as ballot box substitution, chain voting (in which an attacker fills out a ballot and hands it to a voter, who then deposits it and returns an empty ballot to the attacker), and scrutineers filling out blank votes [33]. For instance, the 1994 general elections in Rio de Janeiro were annulled due to numerous fraud incidents.

Motivated by this issue, in 1996 Brazil introduced a fully electronic voting system, using Direct Recording Electronic (DRE) voting machines. These machines significantly reduced electoral fraud, minimized human intervention, and increased the efficiency of vote counting and tallying. As a result, voting machines were rapidly adopted across the country, with 100% of votes being cast electronically by 2000. One source of criticism, however, is that DRE machines lack a software-independent audit trail [25]. Initially, concerns about this limitation were primarily raised by security researchers, based on (arguably) reasonable technical arguments [1,2]. However, over the years, these concerns assumed a more political nature. In particular, they gained more attention in the Brazilian 2018 Presidential Elections: at that time, declarations against the country's electronic electoral system were commonplace in the campaign of Jair Bolsonaro [14,21], who was elected president that year. This political movement led to a surge in accusations against the electoral system, many of which lacked any technical basis. In response, the Election Authority launched campaigns to defend the integrity of the electoral system and its security features [31]. Despite these efforts, trust in the voting machines decreased noticeably among a significant portion of the population [3].

**Procedures.** Elections in Brazil are managed by a centralized Electoral Authority, the TSE (*Tribunal Superior Eleitoral* or Supreme Electoral Court) [32]. Voters cast their ballots using Direct Recording Electronic (DRE) voting machines (Fig. 1), which consist of two main components: a poll-worker terminal for voter authentication, using an identification document and the voter's fingerprint biometrics; and a voter terminal, where votes are cast through a numerical keypad. The machines also include a printer, removable media for storing electoral data, and a USB port—currently disabled, but useful for potential future expansions. Security in these voting machines is ensured through an internal Trusted Platform Module (TPM), a tamper-resistant module responsible for executing critical cryptographic operations, such as verifying software authenticity and signing electoral data.

**Fig. 1.** (a) Brazilian voting machine, comprising the voter terminal and the poll-worker terminal. (b) Currently disabled USB port in poll-worker terminal for potential expansion.

The voting machines are prepared a few days before the election. During this process, the machines are loaded with the official software for the election. Since the voting machines do not have networking capabilities, they are also preloaded with information specific to the assigned polling station. This includes (1) the list of candidates to be displayed, (2) the list of voters assigned to that station, and (3) the biometric data of voters for authentication. Once prepared, the voting machines are distributed to their respective polling stations.

On election day, each polling station—typically a room in a school or public building—receives one voting machine, along with three poll workers (usually volunteers from the community) responsible for overseeing the electoral process. When a voter arrives to cast their ballot, the poll workers first inspect the voter's ID and visually confirm that the photo matches the individual. Next, they check the voter's registration in a list, enter the voter's ID number into the poll worker terminal, and request fingerprint authentication. If the fingerprint is not recognized (e.g., due to damage), the poll worker can manually exempt the voter by entering the voter's birth year. After authentication, the voter proceeds to the voting terminal to cast their vote.

At the end of the voting period (usually at 5 PM), the poll workers instruct the voting machines to finalize the election. The voting machine then generates the poll tape (*Boletim de Urna - BU*), a document containing the total votes cast on that machine. This poll tape is printed, hand-signed, and fixed in a public location. An election official then removes the external media from the voting machine—with the poll tape, logs, and other election data—and transmits it to the TSE. Then, in the tally, the TSE processes the received poll tapes, periodically updating and publishing partial electoral results. Once a significant percentage of votes have been processed (usually around 9–10 PM), the winner is determined and announced.

## 3 Solutions

In this section, we introduce four solutions for voting anywhere using the Brazilian elections as a case study. Table 1 summarizes the main challenges of each solution.

**Table 1.** Comparison between anywhere voting solutions

| Solution | Benefits | Main Challenges |
|---|---|---|
| Internet-connected EPBs | Easy understanding, few changes to the process | Risk of denial of service (DoS) |
| Single line | Easy understanding | High cost and big changes in the voting process |
| Duplicates Identification | Robust against attacks | Hard to explain to the population |
| Secure hardware | Easy understanding | Distribution costs |

### 3.1 Internet-Connected EPBs

The most straightforward solution is to use an Internet-connected electronic poll book to authenticate users, as employed in the USA and other countries. One simple approach is to directly connect the poll worker terminal to the Internet, since it is already responsible for voter verification. However, this approach could raise significant trust concerns, as it would effectively compromise the offline nature of the voting machines—a feature the TSE has emphasized as a key security guarantee. A more likely alternative would be to introduce a separate, internet-connected device dedicated to additional voter identification. This equipment could be either a conventional laptop to minimize costs or custom-designed hardware to reduce the risk of malware.

**Challenges.** We consider this solution inadequate for the Brazilian context. First, not all polling stations have reliable internet access. Many polling places, particularly in remote regions such as the Amazon Rainforest, experience unstable or even nonexistent connectivity. As a result, these areas would need to rely on satellite-based internet, which is currently both costly and unreliable, making it unsuitable as a continuous, critical component of the election process.

Additionally, as discussed in Sect. 2.1, internet-connected EPBs pose significant availability risks, particularly due to their vulnerability to Denial of Service (DoS) attacks. For example, malicious actors could disrupt the election process by blocking internet access in targeted regions. Although this attack seems to be rare in other countries, it is a concern in the Brazilian case. As an example, in the 2022 elections, some authorities have been accused of intentionally

hindering access to voting polls in strategic locations [28]. In addition, the use of Jammers to disrupt connectivity is common, as seen in cases like cargo theft in trucks [27]. Given this context, introducing a relatively simple and untraceable method for selectively disrupting the election process would likely be exploited by attackers, potentially compromising the election results to an unacceptable degree.

### 3.2 Single Line

This solution introduces an extra authentication step at the entrance of the polling place. In Brazil, polling places are typically public buildings such as schools or colleges, with polling stations located in separate rooms, each serving its own line of voters. Under this solution, voters form a single line at the entrance of the polling place, authenticate, and then go to one of the available stations within the building. This approach aims to reduce disparities in station workload and prevent certain stations from becoming significantly more congested than others. This solution proposes a gradual implementation of anywhere voting, starting within individual polling places, then extending to the city or state level, and ultimately to the entire country.

Figure 2 illustrates the operation of the proposed solution. To cast a vote, voters first join a common line at the entrance of the polling place. They then proceed to an Authentication Terminal (AT), a dedicated device responsible for verifying the voter's identity, including biometric authentication. After verification, the AT issues a paper-based token containing a QR code and the assigned polling station. The voter proceeds to the designated station and presents the token, which is scanned by the voting machine. The machine then performs a second biometric authentication before the voter is permitted to cast their vote.

To expand this solution to support anywhere voting in other places, Authentication Terminals (ATs) could be synchronized with a central system via the Internet prior to election day (e.g., one week in advance). Thus, voters would be able to select their preferred polling location up to this day, and the corresponding ATs would be updated with the necessary voter data. If a voter does not express a preference, it is assumed that they intend to vote in their original polling place, so that only the corresponding AT would be updated. This approach offers a reasonable compromise between flexibility and security: voters gain the ability to choose their polling location up to one week before the election, while election equipment remains offline on election day, mitigating risks associated with continuous Internet connectivity.

**Challenges.** This solution introduces significant complexity to the voting process. First, rather than going directly to their assigned polling station as they traditionally do, voters must first authenticate at the Authentication Terminal (AT) and obtain a token. This change is likely to cause confusion, particularly among voters accustomed to using the same polling station year after year. Second, since the tokens are paper-based, they are susceptible to being

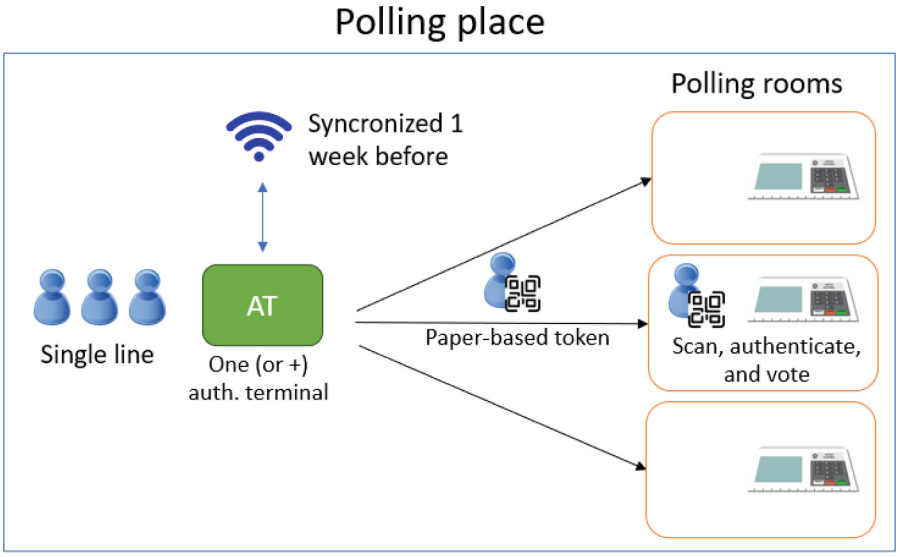


**Fig. 2.** Overview of the single-line solution

crumpled, torn, or lost, which would require the voter to return to the AT to generate a new (identical) token. Third, this process requires voters to undergo biometric authentication twice—once at the AT and again at the voting machine. While this redundancy is a necessary safeguard to prevent the transfer of tokens between individuals, it significantly increases the overall voting time. Finally, this solution would substantially increase the election costs, due to the introduction of new security-critical AT equipment, the need for additional staff to operate them, and the increased logistical and storage costs.

### 3.3 Duplicates Identification

Rather than preventing double voting, this solution focuses on detecting and removing duplicate votes during the election tally. By doing so, it avoids the challenges associated with continuous Internet connectivity during the voting phase. To preserve voter privacy, the system uses anonymization techniques such as mixnets and threshold cryptography. Additionally, to guarantee the vote integrity, it incorporates end-to-end (E2E) verifiability, adapted to the specific context of Brazilian elections [9].

Figure 3 shows an overview of the proposed solution. Essentially, each voter receives a token that allows them to vote in any polling station. This token is scanned by the voting machine, which records the token ID alongside the encrypted vote. After the election ends, all votes are aggregated, and duplicate votes are identified by verifying the token IDs. These duplicates are then selected, shuffled using a mixnet, threshold decrypted, and removed from the final election results.

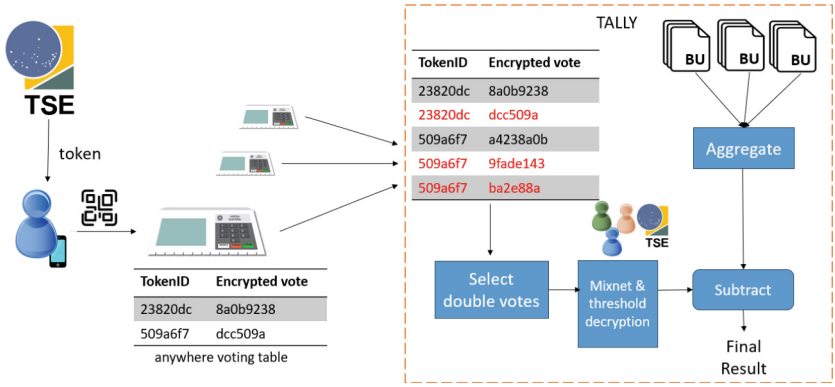


**Fig. 3.** Overview of the duplicates identification solution

One advantage of this solution is that it introduces almost no modifications to the existing voting process. Voters who are not interested in anywhere voting can continue voting as usual—by presenting their IDs and casting their vote at their pre-assigned polling station. Meanwhile, voters who opt for anywhere voting can do so using only a smartphone with a QR code token. In addition, the hardware requirements are minimal: only the addition of a QR code scanner.

**Details.** This solution can be divided into three main phases: token generation, voting, and tally. Figure 4 shows a high-level overview of the protocol.

During a setup phase, a set of trustees (or a threshold of them) and the Electoral Court (TSE) generate election public parameters. That is, the trustees (in cooperation) generate the public key for vote encryption, while the Electoral Court (TSE) generates the seed for pseudorandom $tokenID$. This data is then loaded into the voting machines.

In the token generation phase, voters who wish to participate in anywhere voting must download an official e-voting-ID application. This app authenticates the voter—e.g., via facial biometrics—and requests a token from the TSE. The TSE then generates the token, which essentially is a pseudorandom $tokenID$ and an encrypted biometric data, and sends it to the app. If the voter has already requested a token previously, the TSE must return the same token. Note that only voters opting into anywhere voting must generate this token. Therefore, voters with no interest in this feature may continue voting as usual, at their original polling station, using only their official ID. In addition, even voters who have generated the token can vote in their original polling station without their smartphones, as a fallback. To enable this, each voting machine must have the biometric data and the $tokenID$ of all voters assigned to that location.

In the voting phase, voters go to any polling station to cast their vote. The voting machine scans the token's QR code and checks its digital signature. It then extracts the biometric data from the token and authenticates the voter. Once authenticated, the voter cast her vote. The voting machine stores the

**Setup**

1. Generate public key for vote encryption using threshold
2. Generate seed for pseudorandom $tokenID$ generation
3. Insert the seed and encryption key in the voting machines

**Token generation**

1. The voter downloads the voting-ID app and authenticates (e.g., facial biometry). The voter requests a new voting token
2. If the voter has not yet generated a token, the TSE:
   - generates a new pseudorandom $tokenID$ (using seed and voter information such as ID)
   - select and encrypts voter's fingerprint data, $biometry$
   - $token \leftarrow (tokenID, |biometry|)$
   - signs and sends the token to the voter
3. If the voter has already generated a token, send the same token

**Voting**

1. The voting machine receives the token (through QR code scanning)
2. The voting machine verifies the token, extracts the voter's biometry, and authenticates the voter's fingerprint
3. The voter cast a vote
4. The voting machine encrypts and stores $any\_vote \leftarrow (tokenID, ||vote||, metadata)$.
5. At the end of the election, the voting machine generates the $poll\_tape$ (the sum of all received votes). It transmit the $poll\_tape$ and the $any\_votes$ to TSE

**Tally**

1. TSE sum all $poll\_tapes$ and generates the $raw\_election\_results$
2. TSE aggregates all $any\_votes$ to one single table.
3. TSE identify the double votes by comparing the $tokenID$. The double votes are selected based on their metadata (e.g., timestamp)
4. The TSE retrieves, from double votes, the encrypted $||vote||$. The TSE suffles them through Mixnet.
5. The output of the Mixnet is threshold decrypted.
6. The revealed double votes are subtracted from $raw\_election\_results$. The TSE publishes the final election result.

**Fig. 4.** Voting protocol in the duplicates identification solution

resulting *any_vote*, which consists of the encrypted vote along with the voter's $tokenID$. At the end of the election, each voting machine produces a poll tape, an aggregate of all votes cast on that machine. Both the individual *any_votes* and the corresponding poll tape are then transmitted to the TSE.

Finally, in the tally phase, the TSE receives and processes all poll tapes as usual, generating the initial raw electoral results. Then, the TSE aggregates all *any_votes* into one single table and identifies any double votes by comparing their $tokenID$. The encrypted $||vote||$ from double votes are then extracted and shuffled through a mixnet. The output of the mixnet is then threshold decrypted. Finally, the revealed double votes are subtracted from the raw results, producing the final electoral results.

Note that double votes should be selected using a pre-defined and publicly verifiable criterion. For example, a possible approach could be to consider only the first vote as valid. However, methods relying on timestamps can be vulnerable to manipulation. An attacker, for example, could impersonate the voter at the

beginning of the election to invalidate the legitimate vote. This risk is further amplified by the fact that biometric authentication is optional: if the fingerprint authentication fails, voters are still allowed to vote by providing their birth year. One potential solution to address this issue is to select double votes randomly (e.g., by taking their hash and considering its alphanumeric order). In addition, votes with successful biometric authentication could be prioritized: if there is one vote authenticated biometrically and another without authentication, the unauthenticated vote would be considered as fraudulent and discarded.

**Challenges.** Although this solution protects against double voting and keeps the vote privacy, its implementation in a real-world election presents significant challenges.

First, it is hard to explain this solution to the population. The electoral authority has been frequently accused of electoral fraud, leading to a general lack of public trust. Thus, any change that appears to increase the TSE's power is likely to be met with public skepticism. For example, in this solution, the electoral authority must select which double votes to remove. Even though this selection is pre-defined and publicly verifiable, the mere perception that the TSE is "removing votes" tends to be negatively received by the public.

To further illustrate this problem, consider that the TSE periodically publishes the partial election results as poll tapes are processed. Thus, it is common for candidates and the population to constantly follow the results, learning the number of received votes up to then and trying to guess the probable winners. Under the this approach, however, the number of received votes from some candidates could decrease as invalid votes are removed. While this process is entirely legitimate, the visible reduction in vote totals may create hard feelings among candidates and their supporters, increasing distrust on TSE. In addition, in some cities, political parties commonly collect and sum the printed poll tapes attached in polling places, allowing them to anticipate results ahead of the official count. In such cases, the removal of duplicate votes – resulting in discrepancies between poll tape totals and the final tally – can create the impression that votes are being "lost". In both cases, confusion and reputation damage may occur before the TSE has an opportunity to clarify the process.

Second, since this solution relies on threshold encryption to provide vote secrecy, it is of critical importance to choose the trustees properly. However, since Brazilian elections are conducted solely by one entity, it is not trivial to determine these trustees. One option is to assign a key share to the Federal Court of Accounts (TCU), which oversees governmental organizations, or to the Armed Forces, which participated in supervising the 2022 elections. However, involving these institutions could be considered a deviation from their primary roles and would likely encounter significant political resistance. Alternatively, key shares could be distributed among political parties. However, with more than 30 registered parties in Brazil, selecting only a few to fulfill this role could raise concerns about fairness and representativeness.

This challenge is further exacerbated by the fact that each key share is a critical component of the electoral process. As such, the selection of trustees introduces a significant risk of electoral disruption if any of them refuse, or are unable, to participate in the decryption phase. For example, a political party might "accidentally" lose its decryption key. Thus, it is likely that the TSE would be unwilling to assume this risk and would instead choose to control all key shares internally or distribute them only among its regional branches (TREs).

Third, an attacker could steal tokens through coercion or malware to cast a vote in place of voters. Although this is already possible currently (e.g., attackers could steal IDs from voters), this solution could intensify the risk. For example, with anywhere voting, attackers could use all tokens in the same corrupted polling station, increasing the attacker's scalability and reducing their costs. Nevertheless, this attack is easy to detect, since (1) if there are legitimate votes cast with biometric authentication, the attacker's votes would be removed, and (2) a high number of double votes and biometric liberation from staff would likely raise suspicion.

Finally, this solution requires the implementation of advanced cryptographic techniques. As a result, the software must be developed with extreme care, as even minor implementation errors could introduce security vulnerabilities that compromise the system's integrity and damage the election's reputation. To enhance software quality and transparency, the TSE could adopt measures such as open-source code, to facilitate public inspection, and bug bounty programs, as seen in the SwissPost e-voting system [10,16].

### 3.4 Secure Hardware

This solution tries to prevent duplicate votes without excessively impacting the voting process. Essentially, each voter receives a smartcard or a USB security token, configured by the TSE to allow only one vote per election. This device communicates with the voting machine during the voting process and is marked as "used" once the vote is cast, preventing any further voting attempts.

One possible protocol could work as follows. First, the voter visits a TSC center in person and receives a secure hardware containing their personal information (e.g., an ID number and encrypted biometric data). This hardware is configured to enable voting in the election by performing the following steps: (1) privately generating a public and secret key pair; (2) sending the public key to TSE; and (3) the TSE generating and sending a certificate for the token, which includes the voter's information, a random token ID, and the election ID. When the secure hardware is connected to the voting machine, it follows these steps (Fig. 5): (1) verifies the certificate; (2) retrieves the biometric data and authenticates the voter; (3) securely deletes the private key from the hardware; and (4) accepts the vote. In the next election, the secure hardware can be reconfigured remotely.

However, Brazil does not have yet a widespread adoption of secure hardware. Currently, the most popular initiative involving secure hardware is the

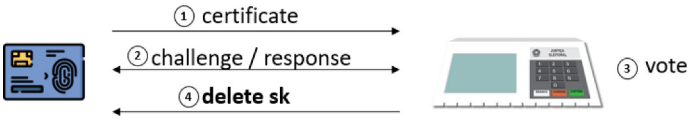


**Fig. 5.** Simplified flow of the voting procedure using secure hardware

e-CPF/CNPJ, which is used to digitally sign official documents and is primarily utilized by businesses and specific individuals, such as lawyers. Thus, one potential solution is to integrate the voting functionality into the e-CPF/CNPJ secure hardware. This would not only encourage the adoption of e-CPF/CNPJ initiative but also increase the hardware's functionality, encouraging better care and maintenance of these devices.

**Challenges.** While this solution mitigates the risk of double voting, it does not entirely eliminate it. Smartcard technologies are not completely invulnerable to attacks, and while cryptographic key extraction or logic subversion typically results in the destruction of the device, there remains a small probability that such attacks could succeed. Thus, attackers with substantial resources could potentially use and destroy multiple secure hardware devices to extract one or more secret keys, enabling them to cast double votes.

Thus, it is convenient to combine this technique with additional mechanisms to detect or remove double votes. Even if these mechanisms are not activated, their mere existence could serve to discourage potential attackers from investing significant resources to cast double votes. One possibility is to integrate the mixnet and threshold cryptography solution to identify and remove double votes (Sect. 3.3). Alternatively, the *tokenID* of all cast votes could be retrieved, enabling the detection of double votes at the conclusion of the election and imposing penalties on the corresponding voters (e.g., by not notifying them of a lost or stolen device). This approach is similar to the "detect and punish" strategy adopted in Australia (Sect. 2.1).

Another challenge is the potential loss of the device. If a voter loses their secure hardware, they would need to reissue the token to vote. This process could present a significant obstacle, particularly if the loss is discovered shortly before the election.

Finally, a significant challenge is to reliably distribute the secure hardware among the population. Currently, the e-CPF/CNPJ has relatively low adoption, partly due to its cost (approximately 35 US dollars). Since a significant portion of these costs is associated with in-person user validation, they could potentially be reduced by using the TSE's existing infrastructure for voter validation. The remaining costs could be subsidized by the government, particularly for voters in economically vulnerable situations.

## 4 Discussion

From our analysis, we believe that **the most promising approach is to integrate the duplicates identification solution with secure hardware**. While the duplicates identification approach alone is sufficient to protect against double votes, its reputation risks seem to be too high for TSE—especially in terms of explaining to the public how and why votes will be removed. On the other hand, the secure hardware solution alone cannot completely protect against hacking and double votes. Thus, we conclude that the most promising solution is to reduce at most the number of double votes through secure hardware, while using mixnets and threshold cryptography to detect and eliminate any remaining double votes.

With this configuration, we believe the number of double votes will be sufficiently low that their effect in the tally will be nearly imperceptible. Then, after some tests in small-sized elections, the TSE can better evaluate the risks of double voting and potentially relax some restrictions. For example, voters could be allowed to vote in their original sections even without their secure hardware (e.g., in case of losses or thefts), assuming that any resulting double votes would be identified and removed later.

**Costs.** One of the main costs of this solution is the secure hardware distribution. As discussed in Sect. 3.4, these costs could be mitigated by integrating the voting functionality into existing secure hardware solutions (e.g., e-CPF/CNPJ) and by utilizing TSE centers for in-person activation. Since secure hardware for personal identification can have numerous applications beyond electronic voting—such as for online digital signatures or tax declarations—subsidizing the cost of secure hardware for the population appears to be a reasonable approach.

For the voting machines, there is no need for any hardware modification. The only requirement would be some way to communicate with the secure hardware, which could be done through the existing USB port. In addition, there is no need for all voting machines to have this capability. It is enough that only some polling stations be able to receive anywhere voting, as long as they are strategically distributed across cities and neighborhoods. Also, no additional staff is needed to operate the machines.

Therefore, we believe that the biggest impact of the solution would be on the additional procedures and software development required. In particular, the TSE would need to establish procedures and provide services for obtaining, activating, and revoking secure hardware for anywhere voting. In addition, the tally software would need to be expanded to detect and remove double votes. The tally procedure itself would also require modification to incorporate the mixing and threshold decryption operations performed by the trustees.

## 5 Future Works and Conclusion

In this work, we present four potential alternatives to implement anywhere voting in more restricted scenarios, without constant Internet access. Using Brazil

as a case study, we evaluate each alternative, highlighting its advantages and challenges. Our analysis concludes that the most promising approach is to combine mixnet and threshold cryptography to detect and remove double votes at the tally, while using secure hardware to minimize the number of double votes.

Future works include formally describing the protocol, particularly the remove duplicates solution, and establishing its security properties along with formal security proofs. In addition, we plan to implement an experimental version of the solution to evaluate its feasibility and performance in the Brazilian elections.

**Acknowledgments.** This work was in part supported by the Brazilian CNPQ (Research productivity grant 307732/2023-1), and CAPES (Finance Code 001).

**Disclosure of Interests.** Leonardo Kimura and Marcos Simplicio report financial support was provided by Superior Electoral Court. Other authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Aranha, D.F., Barbosa, P.Y., Cardoso, T.N., Araújo, C.L., Matias, P.: The return of software vulnerabilities in the Brazilian voting machine. Comput. Secur. **86**, 335–349 (2019)
2. Aranha, D.F., Karam, M.M., de Miranda, A., Scarel, F.B.: Software vulnerabilities in the Brazilian voting machine. In: Design, Development, and Use of Secure Electronic Voting Systems. IGI global (2014)
3. AtlasIntel: Research - Brazil elections 2022 (in Portuguese). Tech. rep., São Paulo, SP, Brazil (2022). https://bit.ly/3YPEYZ1
4. Benaloh, J., Ryan, P.Y.A., Teague, V.: Verifiable postal voting. In: Christianson, B., Malcolm, J., Stajano, F., Anderson, J., Bonneau, J. (eds.) Security Protocols 2013. LNCS, vol. 8263, pp. 54–65. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41717-7_8
5. Borges, B.: Biometrics does not guarantee election security, experts say (in Portuguese). UOL (2014). https://bit.ly/3PJuhD6
6. Burton, O.V., Eisenstadt, P.: Voting rights in Georgia: A short history. Southern Cultures (2024)
7. Cantoni, E.: A precinct too far: turnout and voting costs. Appl. Econ. Am. Econ. J. **12**, 61–85 (2020)
8. Chief Election Officer: Administrative report - 2014 municipal election review. Tech. rep., City of Vancouver (2015). https://bit.ly/3GIxRvi
9. Cominetti, E., Simplicio, M., Aranha, D.F., Matias, P., Araujo, R.: E2easy: a simple lattice-based in-person end-to-end voting scheme. In: 10th Workshop on Advances in Secure Electronic Voting Schemes (2025)
10. Cortier, V., Debant, A., Gaudry, P.: A privacy attack on the swiss post e-voting system. In: Real World Crypto Symposium (2022)
11. Cortés, E., Howard, L., Norden, L.: Prevent and recover from electronic pollbook failures and outages. Tech. rep., Brennan Center for Justice (2018). https://bit.ly/44NNX0F

12. Dalton, R.J.: Citizen Politics: Public Opinion and Political Parties in Advanced Industrial Democracies. CQ Press (2020)
13. Dassonneville, R., Hooghe, M.: Voter turnout decline and stratification: quasi-experimental and comparative evidence of a growing educational gap. Politics **37**, 184–200 (2017)
14. Funke, D.: In Brazil's presidential election, hoaxes about voter fraud run rampant. Poyntier (2018). https://bit.ly/42cJfbJ
15. Gibson, J.P., Krimmer, R., Teague, V., Pomares, J.: A review of E-voting: the past, present and future. Ann. Telecommun. **71**(7), 279–286 (2016). https://doi.org/10.1007/s12243-016-0525-8
16. Haines, T., Pereira, O., Teague, V.: Running the race: a Swiss voting story. In: Krimmer, R., Volkamer, M., Duenas-Cid, D., Rønne, P., Germann, M. (eds.) International Joint Conference on Electronic Voting. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15911-4_4
17. Harder, J., Krosnick, J.A.: Why do people vote? A psychological analysis of the causes of voter turnout. J. Soc. Issues **64**, 525–549 (2008)
18. Heinrich Böll Fundation: Voter Abstention in Greece - Comparative survey of voters and abstainers in the June 2023 National Election. Tech. rep., Kapa Research (2024). https://bit.ly/4cQtbzy
19. Kuzio, T.: Political culture and democracy: Ukraine as an immobile state. East European Politics and Societies (2011)
20. Labour force survey: Reasons for not voting in the federal election. Tech. rep., The Daily (2022). https://bit.ly/4m79mIG
21. Nicas, J., Milhorance, F., Ionova, A.: How bolsonaro built the myth of stolen elections in Brazil. The New York Times (2022). https://bit.ly/3A8Gywk
22. Norris, P.: Critical Citizens: Global Support for Democratic Government. Oxford University Press (1999)
23. Peterlevitz, T.: Who turns to clientelism? Opportunistic politicians, patronage appointments, and vote buying in brazil. SSRN (2022)
24. Puglisi, L.: Here's the facts about multiple voting - It is negligible in Australia (2023). https://bit.ly/4lVsw3V
25. Rivest, R.: On the notion of 'software independence' in voting systems. Philos. Trans. A Math. Phys. Eng. Sci. **366**, 3759–3767 (2008)
26. Schneider, R.: Free or fair elections? The introduction of electronic voting in Brazil. Economía (2020)
27. SETCESP: Jammer is used in over 90% of cargo theft (in Portuguese) (2023). https://bit.ly/42Wr4pn
28. Stargardter, G.: Brazil highway police blockades fan voter - Suppression fears. Reuters (2022). https://bit.ly/4cVF73b
29. Stein, R.M., Vonnahme, G.: Effect of election day vote centers on voter participation. Election Law J. Rules Politics Policy **11**, 291–301 (2012)
30. The Electoral Comission: Vote anywhere. The Electoral Comission (2023). https://bit.ly/3U5sAT3
31. TSE: Brazil's electoral justice permanent program on countering disinformation (2022). https://bit.ly/3YlTGqF
32. TSE: Practical guide - 2022 Brazilian elections (2022). https://bit.ly/3C1wxlv
33. TSE: Why the voting process is electronic (in Portuguese) (2024). https://bit.ly/43IVZ8s
34. U.S. Election Assistance Comission: Electronic poll book report. Tech. rep. (2023). https://bit.ly/3SbJaP4

35. Xavier, B.: Datafolha - 34% of Brazilians say they would not have gone to vote if it were not mandatory (in Portuguese). Folha de Sao Paulo (2024). https://bit.ly/3EN7svP
36. Zetter, K.: L.A. County has found the cause of its hourslong poll lines. It wasn't the new voting machines. Politico (2020). https://bit.ly/3VGOLA9



# Credential Attacks in Ontario's Online Elections

Eric Klassen[1], James Brunet[2], Nicole Goodman[3], and Aleksander Essex[1(✉)]

[1] Western University, London, Canada
aessex@uwo.ca
[2] Carlton University, Ottawa, Canada
[3] Brock University, St. Catharines, Canada

**Abstract.** We propose two novel voter authentication attacks in the context of the 2022 Ontario Municipal Election, which offered online voting to almost four million voters in over 200 municipalities. One attack exploits a misconfiguration in one of the voting portals used by up to one million voters. It was mitigated through a successful coordinated vulnerability disclosure that we conducted with the affected vendor during the election period. The other attack exploits widespread and insecurely discarded login credentials. This attack affects the vast majority of the deployments examined, and we study and quantify the risk for each city individually. In both cases, the risks were aggravated by unique, context-dependent factors, which we detail. Finally, toward quantifying this risk, and absent the availability of this data elsewhere, we present a comprehensive census of online deployments used in the province.

## 1 Introduction

This paper studies online voter authentication mechanisms used in the 2022 Ontario Municipal Election, involving 417 separate local elections, serving a combined 10.7 million eligible voters [1]. Of these municipalities, 219 offered an online voting option, of which 158 cities eliminated the paper ballot altogether.

Owing to the inherently unsupervised nature of remote online voting, strong digital credential mechanisms are necessary to ensure that a digital ballot reflects the genuine intent of the designated voter. Coercion and vote-selling are two well-studied examples of attacks in this setting (among many such works, see e.g. [2–5]). This paper, however, focuses on a different category of credential attacks—those in which voters are unknowing participants.

Our inquiry began with an observation communicated to us by one municipal election official that voter information letters (containing the voter's login PIN) had been observed discarded in a communal trash bin unopened and hence unused. Based on this, we consider the following research questions: Which municipalities offered online voting in the 2022 election, and how did voters authenticate to these systems? What is the potential insecurely discarded voter login credentials, and how can we quantify the risk for each city? What would be


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 141–157, 2026.
https://doi.org/10.1007/978-3-032-05036-6_9

the attack complexity required to change an election outcome? Were there any municipalities where this attack could be considered 'easy' under our defined risk metric? Could we observe other vulnerabilities in the online voting portals that allow voter login credentials to be exploited?

Our findings suggest weak authentication methods in Ontario are a weak point in most of the observed systems, highlighting the dilemma faced by 1.6 million eligible voters in cities where paper ballots were eliminated: Accept the cyber risks introduced by the design design decisions of these deployments, or, as one judicial decision from a 2019 election challenge blithely put it, accept "voluntary disenfranchisement" [6].

**Findings and Contributions.** The contributions of this work are threefold. First, to facilitate our credential risk analysis, we conducted a comprehensive census of cities and vendors offering online voting. While basic data about online voting adoption was published by the Association of Municipalities of Ontario (AMO) [1], we undertook an intensive supplementary data collection effort to determine which cities used which online voting vendors and how, in each case, they approached voter authentication. Section 2 describes the methods used to collect this data and presents findings about online voting in Ontario, which includes our census of vendors and cities that offered online voting in 2022 and how the landscape has changed since the previous election cycle.

Second, we present a novel effort to quantify the cyber risk of discarded credentials in Sect. 3, correlating each municipality's credential scheme against their electoral outcomes, combined with additional context-dependent factors to compute an overall risk score in each case. Of the 219 Ontario cities offering online voting, we found that the majority (70%) were at high risk of this attack under the proposed metric.

Third, we present a novel cross-site browser framing vulnerability in Sect. 4 that could allow an attacker to misdirect a voter into casting a ballot for an unintended candidate. We highlight additional aggravating factors that increased the attack's severity beyond the general case. Finally, we describe our coordinated vulnerability disclosure effort with the affected vendor (Scytl) during the live election period. They promptly issued a mitigation, affecting as many as one million voters, and issued an advisory to their municipal clients acknowledging our findings.

## 2 Background and Methodology

The election was held on October 14th, 2022, although early voting began in some municipalities at least as early as October 9th. 417 separate municipal elections served a total of 10.7 million eligible voters [1]. While many voting methods were offered, this work focuses on the 219 municipalities offering online voting to a combined 3.8 million eligible voters. Of these, 154 municipalities eliminated paper ballots altogether, offering only a remote electronic voting method (either online or by telephone) to their combined total of 1.7 million eligible voters.

### 2.1 Data Collection Methodology

Comprehensive information about Ontario's online voting landscape is not readily available. Federal or provincial governments do not track the marketplace. Vendors typically decline to volunteer this information and in past instances have even refused our requests for it. The AMO publishes election results and tracks voting methods used by municipalities, but does not track vendor information [1]. The AMO also does not provide this information as a single usable dataset; instead, it provides data about the 444 Ontario municipalities across 444 separate web pages. An AMO representative told us that our request for a dataset could not be fulfilled, so we wrote a script to collect, extract and aggregate this data from their multitude of individual web pages.

To determine which municipalities worked with which vendor, we began with the observation from previous work [7] that vendors tended to follow a consistent URL format for each city's voting portal. For instance, Simply Voting used the URL convention https://{municipality}.simplyvoting.com, while Dominion used https://intvoting.com/{municipality}. We used this method to create an initial list during the early voting period in October. To validate this list and to determine the types of login credentials used by each city, we undertook an intensive process in the months following the election, manually locating and examining a wide range of online sources, including council meeting minutes, municipal staff reports, social media pages, YouTube videos, and the Internet Archive Wayback Machine. This approach allowed us to identify and validate all city/vendor pairs with high confidence.

### 2.2 Vendor Census Results and Dataset

The election results dataset and census form the basis of our analysis of voter authentication vulnerabilities in Sect. 3. Table 1 shows a summary of our vendor census. Our full dataset is available online.[1] In addition to presenting information about which vendors ran online elections for which cities, we combine this information with previous work from the 2018 election [7] showing the growth of online voting adoption in the province.

As the data shows, online voting adoption grew by 25% between the 2018 and 2024 election cycles, with over 50% of municipalities (representing over 3.8 million voters) now offering an online voting option.[2] One interesting finding was that the market share of Dominion Voting Systems declined considerably in 2022. Of Dominion's 49 municipal clients in 2018, 34 switched to a new vendor in 2022, and 8 discontinued online voting altogether. We conjecture this outcome is in large part due to their 2018 election night service disruption due to insufficient bandwidth provisioning [7]. Scytl's market share increased partly because they partnered with Intelivote in 2018, but operated as a separate provider in 2022.

[1] 2022 Ontario Municipal Election Vendor Census. https://github.com/eklass3/ONElectionData_2022.

[2] 219 municipalities represent over 50% since not all 444 municipalities run elections (mainly in the upper-tier/regional municipalities).

Neuvote and Voatz were newcomers to the market, and we anticipate additional new vendors will enter the market in 2026.

**Table 1.** Summary of our Ontario online voting vendor census: 2018 to 2022.

| Year | Dominion | Intelivote | Neuvote | Scytl | Simply Voting | Voatz | Total |
|---|---|---|---|---|---|---|---|
| **Eligible Voters by Vendor** | | | | | | | |
| 2018 | 1,323,194 | 860,985 | n/a | 253,437 | 304,479 | n/a | 2,742,095 |
| 2022 | 996,965 | 679,910 | 1,049 | 1,037,635 | 714,360 | 402,385 | 3,832,304 |
| Change | (326,229) | (181,075) | 1,049 | 784,198 | 409,881 | 402,385 | 1,090,209 |
| **Municipal Clients by Vendor** | | | | | | | |
| 2018 | 49 | 98 | 0 | 2 | 28 | 0 | 177 |
| 2022 | 23 | 96 | 1 | 36 | 49 | 14 | 219 |
| Change | (26) | (2) | 1 | 34 | 21 | 14 | 42 |

### 2.3 Observed Credential Mechanisms

Based on our findings from our vendor census, we found that credentials were overwhelmingly distributed to voters by postal mail. Voters receive a voter information letter (VIL) in their mailbox, which contains a login credential (typically an 8–16 digit PIN) [7] and the URL of the voting site. Voters authenticate to the website by providing the PIN from the VIL along with their date of birth. The risks of insecurely discarded credentials under this approach are explored in Sect. 3. The VIL instructs voters to connect to the voting site on their device by typing a partial URL into their browser's address bar (see Fig. 1).[3] This, along with an additionally observed server configuration vulnerability, provides the basis for the cross-site browser framing attack discussed in Sect. 4.

## 3 Risk Analysis of Insecurely Discarded Credentials

In this section, we examine the feasibility of harvesting discarded login credentials as an attack to maliciously alter an election outcome. Two observations are at the core of the proposed attack. First, harvesting a discarded credential offers a low-detectability way to cast a fraudulent ballot in cities that only offer online voting. Second is the observation that each of the 1.7 million eligible voters

[3] How to Vote Online - 2022 Ontario Municipal and School Board Election. Town of the Blue Mountains. Available: https://www.youtube.com/watch?v=RgJobco9cxs.

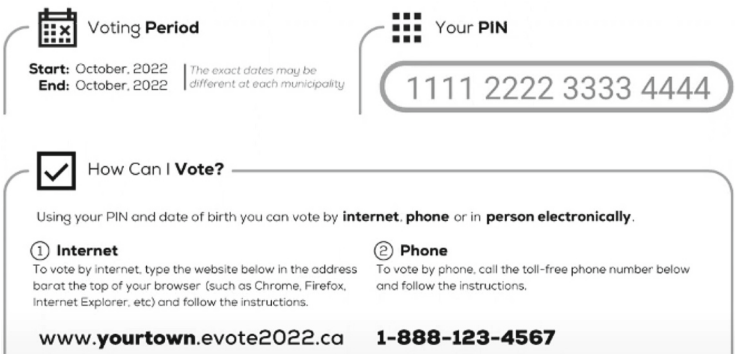


**Fig. 1.** Example Voter Information Letter (VIL) directing user to type a partial URL.[3]

across the 154 online voting-only cities automatically received a Voter Information Letter (VIL) by postal mail containing a login credential (typically an 8–16 digit PIN). Since voter turnout in 2022 averaged 36.3% [1] across the province, we can infer that over a million credentials went unused. Based on the expectation that most credentials were insecurely discarded (as discussed below), this section attempts to quantify the risk of this attack across each city by examining the number of unused credentials relative to the declared victory margins in combination with the secondary date-of-birth credential used (if any).

The impetus to study this issue comes from an observation communicated to us by an Ontario election official (from an unpublished study currently in progress), who was made aware of discarded, unopened VILs in recycling bins at one of their community mailbox locations. Although this example is specific to one municipality, the conditions that facilitate such occurrences—centralized mail delivery and low voter engagement—are shared by most municipalities, suggesting that this issue is likely widespread. Our goal is to provide a simple metric that election officials can apply to their unique cases based on overt risk factors such as the widespread availability of unused VILs and the relative strength of secondary credentials like date of birth (DOB). We note our analysis is based on the reported login credential methods and voter turnout in each city only. Gathering data about the actual prevalence of discarded VILs would have raised complex legal issues and was beyond the scope of this work. We recommend that municipalities explore this question directly themselves

**Attack Complexity.** Our proposed credential harvesting attack is both manual and labor-intensive, so we begin by defining an upper bound on the effort required for this attack to be reasonably considered feasible. Supposing that a legitimate election campaign could knock on one thousand doors during an election period,[4] we define a malicious effort to collect up to 1,000 discarded VILs

[4] E.g., a recent federal candidate claims to have knocked on 5,000 doors. See https://thetyee.ca/News/2024/03/04/One-Man-March-Beat-Poilievre-Own-Riding/.

**Table 2.** Number of races in the 2022 election at risk of the proposed credential-stealing attack.

| Attack Complexity | Victory Margin | Number of Races |
|---|---|---|
| Easy | 1–10 | 24 |
| Medium | 11–100 | 148 |
| Hard | 101–1000 | 528 |

**Table 3.** Survey of date of birth used as a secondary credential in the 2022 election.

| Combination Type | Municipalities (#) | Municipalities (%) |
|---|---|---|
| DOB (D-M-Y) | 87 | 38.7% |
| DOB (M-Y) | 6 | 2.7% |
| DOB (Y) | 22 | 9.8% |
| Combination unknown | 33 | 14.7% |
| DOB use unknown | 75 | 33.3% |
| Does not use DOB | 2 | 0.9% |

as the upper bound of feasibility for the purposes of this metric. We limited our analysis, therefore, to contests with victory margins of fewer than 1,000 votes. We further subdivide the attack complexity across three orders of magnitude: hard (up to 1,000 VILs), medium (up to 100 VILs), and easy (up to 10 VILs). Using this classification, we found that 700 races across approximately 70% of the 154 municipalities met one of these criteria. Table 2 shows the distribution.

### 3.1 Secondary Authentication Practices in Ontario Municipal Elections

Obtaining discarded VILs alone is typically not sufficient to cast a fraudulent vote. Many (but not all) municipalities require a secondary login credential to vote based on the voter's date of birth. We observed various combinations of year (Y), month (M), and day (D). Importantly, these combinations were often set by the municipality, not the vendor, so we had to confirm the specific configuration used in each municipality by referencing public-facing documents. In some instances, information about DOB combination or even if the municipality required a DOB was not publicly available (i.e., unknown). The analysis results are summarized in Table 3.

As has been previously observed [7], date of birth is a weak login credential: It is inherently low-entropy; It cannot be changed; and, although it is treated as private, it is not inherently *secret*. However, unlike many US states, which publish voter lists with full DOB information, Ontario does not make this information widely available—not even to candidates, so a successful attack would require an adversary to obtain this information by other means.

### 3.2 Options for Obtaining Voter DoB Information

Several entities in the election ecosystem have full access to voter DOB information, including Ontario's Municipal Property Assessment Corporation (MPAC), which curates the preliminary voter lists, as well as the e-pollbook service provider (as do many other companies and agencies in Ontario). Notwithstanding these potential insider threats, we consider potential options for an attacker to obtain Ontario voter DOB information through publicly available sources.

**Data Breaches.** Large-scale data breaches involving dates of birth are a regular occurrence [8–10]. As part of our effort to explore data breach availability, we solicited legal advice from a firm in our jurisdiction specializing in cybercrime law. We were advised that downloading and analyzing freely available datasets represented low legal risk, as long as passwords or credit card numbers were not part of the data. We accordingly restricted our search to datasets where we could confirm these conditions were met prior to accessing them.

As a proof of concept, we downloaded and analyzed the 2019 Facebook data breach, which confirmed the existence of exposed Ontario voter DOBs online. Our analysis found that this dataset contained the first name, last name, city, and full dates of birth of 22,308 Ontario residents. We found an additional 28,276 Ontario residents with partial dates of birth (month and day) in this breach. Legal considerations prevented us from examining several larger data breaches as they contained password information, however it was apparent that the totality of exposure greatly exceeds what we were able to directly observe, given our constraints.

**Online DOB Oracles.** The login page of the voting site represents a kind of date of birth oracle with a limited number of guesses. Using a harvested PIN, the attacker can make a guess the voter's DOB. If the guess is correct, the login will be successful. However, several additional online DOB oracles exist in the form of Elections Canada's, Elections Ontario's, and MPAC's voter registration directories. All three services allow voters to check their registration status by entering their name, address, and date of birth (with the first two revealed to the attacker by the VIL). Using a oracle, the number of guesses expected to confirm a DOB varies by the specific year/month/day combination required by the municipality. We did not attempt to determine how many incorrect guesses could be made to each website before triggering a lockout period. To quantify DOB entropy, we used the Ohio voter registry as a proxy [11], which contains complete DOB information. Ohio is demographically similar to Ontario and comparable in population size (approximately 8 million registered voters). The results of our analysis are shown in Table 4.

Of the 22 Ontario municipalities using year of birth as a secondary credential, our results indicate an attacker can find the correct date of birth in approximately 64 guesses on average, and possibly fewer in instances where a voter's name is correlated with their age. Ontario publishes a dataset of baby

**Table 4.** DOB entropy for differing combinations.

| Authentication Level | Entropy (Bits) |
|---|---|
| Year-Month-Day | 14.6 |
| Year-Month | 9.7 |
| Year | 6.1 |

names for the past hundred years, listing each name and the number of newborns registered with that name (for $k \geq 5$).[5] As an illustrative example, consider an attacker-recovered VIL addressed to a voter named 'Ryszard'. This name only appears in the list between 1950–60, suggesting the voter's DOB could be recovered using an oracle in closer to 10 guesses. Of course, in municipalities using no DOB credential, such as the Town of Atikokan [12], the VIL alone is sufficient to cast a ballot.

### 3.3 Credential Attack Risk Metric

To quantify credential attack risks to municipalities, and similar to the parameterization methodology of the Common Vulnerability Scoring System (CVSS), we developed out risk metric by subjectively ranking cases toward identifying constituent risk factors. Our proposed risk metric is as follows:

$$R_{\text{credential\_attack}} = r_{EV} + r_{AV} + r_{VT} + r_{CC} + r_{HCC} + r_{SF}$$

where:

1. **Eligible Voters Risk** ($r_{EV}$): Scored 0–10 based on the size of the eligible voter population. Each point represents a 10th percentile decrease in population, with larger voting populations reducing the likelihood of close elections.
2. **Alternate Vote Risk** ($r_{AV}$): A binary score (0 or 10) based on the availability of voting methods. Municipalities offering additional voting methods score 0, as a discarded VIL could mean a voter intends to cast their ballot through other means. This would raise suspicion if the voter is logged as having already voted. Municipalities with only internet and telephone voting score 10, as this ensures that unopened and discarded VILs could be used to cast a fraudulent vote.
3. **Voter Turnout Risk** ($r_{VT}$): Scored 0–10, based on the percentage of eligible voters participating in the election. Each point denotes a 10th percentile decrease in voter turnout, with higher turnout decreasing the availability of discarded VILs.
4. **Councillor Closeness Risk** ($r_{CC}$): Scored 0–3 for each elected councillor based on their victory margin for that race. Margins of 1–10 votes score 3, 11–100 votes score 2, 101–1000 votes score 1. Greater than 1000 votes score 0. Smaller margins score higher, indicating past elections could have been more easily influenced, assuming this is an approximate predictor of future risk.

[5] https://data.ontario.ca/dataset/ontario-top-baby-names-male.

5. **Head of Council Closeness Risk** ($r_{HCC}$): Scored 0–3, the same as councillor closeness risk ($r_{CC}$) but based on the elected head of council's (typically, mayoral) victory margins.
6. **Secondary Factor Risk** ($r_{SF}$): Scored according to the difficulty of guessing the secondary credential (i.e., date of birth) based on our entropy calculations in Table 4. The scores are as follows: 5 for unknown credential level or unknown date-of-birth format; 2 for day-month-year format; 5 for month-year format; 8 for year-only format; and 10 for no date of birth used. Higher scores indicate greater risk.

We applied our risk metric to each municipality and categorized them into quartiles based on their point distributions. The highest observed score indicated the most extreme risk, with 69% of municipalities in the 2022 election classified as either high or extreme risk.

As one worked example, the Municipality of Sioux Lookout had the highest risk, scoring 40 points: $r_{EV} = 8$ (3,185 eligible voters), $r_{AV} = 10$ (online and telephone voting offered), $r_{VT} = 6$ (33% voter turnout), $r_{CC} = 11$ (one race with 1–10 margin, three with 11–100 margin, and two with 101–1000 margin gives 3+3(2)+2(1)=11), $r_{HCC} = 0$ (the mayor was acclaimed), $r_{SF} = 5$ (unknown DOB configuration). The summary of risk ratings for all the municipalities is given in Table 5. Our full dataset is available online (see supra note 1).

**Table 5.** Distribution of municipalities based on risk scores.

| Risk Category | Municipalities (#) | Municipalities (%) |
|---|---|---|
| Low (0–9 pts) | 12 | 5.5 |
| Medium (10–19 pts) | 54 | 24.7 |
| High (20–29 pts) | 78 | 35.6 |
| Extreme (30–40 pts) | 75 | 34.2 |
| Total | 219 | 100 |

### 3.4 Assumptions About Prevalence of Discarded Credentials

Our risk metric presumes an abundance of credentials thrown out, unused and intact. One question is how the incidence rate of secure destruction of unused VILs should affect the risk score. Let $n_e, n_v$ represent the number of eligible and actual voters in a given city. Let $n_u = n_e - n_v$ be the total number of unused VILs in a given city. Let $0 \leq \sigma \leq 1$ represent the proportion of unused VILs that are securely destroyed. The number of unused VILs available for a credential attack, therefore, is $n_a = n_u(1 - \sigma)$. As a loose upper bound, we argue the risk rating should not be adjusted downward for cities where $n_a > n_v$, i.e., where more VILs were thrown out than cast in the election itself. Making a conservative

assumption that half of the unused VILs were securely destroyed (i.e., $\sigma = 0.5$), we found 86 cities (39%) still met this criterion (i.e., more credentials were insecurely discarded than used). However, if we assume only one in 5 VILs were securely destroyed (i.e., $\sigma = 0.2$), then 184 (84%) still meet this criterion.

This demonstrates the difficulty municipalities would face to ensure insecurely discarded VILs were scarce enough to make this attack infeasible. Although determining actual values of $\sigma$ was beyond the scope of this study, we suspect it was low across most cities. None of the sample VILs we found online contained instructions to securely destroy an unused VIL,[6] although any such instruction would go unseen by a voter who did not open their letter. We also found no evidence of a plan by any city to offer secure document destruction to voters.

## 4 Cross-Site Browser Framing Attack

This section presents findings relating to a cross-site framing vulnerability that we discovered in the 2022 Ontario Municipal Election and the ensuing coordinated disclosure with the affected vendor (Scytl). We performed a coordinated disclosure during the live election and worked with them to correct the issue. Scytl acknowledged the vulnerability and sent an advisory to their 37 municipal clients, improving the security posture of the voting service for over one million eligible voters.

A *cross-site framing attack* involves a malicious website embedding another website inside an HTML `iframe` to misdirect users into performing unintended actions. Typically, the attack is structured around the expectation that the user is aware that they are visiting the outer (malicious) site, but unaware they are performing actions in the inner (framed) site. In the election context, we consider the reverse, where the voter believes they are interacting with the legitimate voting site, and are unaware it is being framed inside a malicious site to misdirect the voter to cast a vote for an unintended candidate.

### 4.1 Attack Assumptions

Although our proposed attack exploits voter authentication gaps at login time, it does not exploit the credentials directly. It does not exploit the Transport Layer Security (TLS) protocol or modify the voting website's underlying logic or network communications. It assumes the voter follows the instructions *as given* in the Voter Information Letter, which directs them to manually type a partial URL into their browser address bar.

The attack only requires the voter not to notice being redirected to a malicious URL. Generally, this is a strong (i.e., unrealistic) assumption. However, Ontario's online voting context presents at least one context-specific aggravating usability factor that makes this outcome more reasonable in practice: voters

[6] See, e.g., Sample Voter Information Letter. City of Markham. https://www.electionsmarkham.ca/media/lnbhp4iq/voter-information-letter-sample-english.pdf.

(and their browsers) encounter the genuine election site URL for the first time when voting.

Our analysis found that most cities offering online services used a new or different URL than in prior years. We identified three causes: Either the city was offering online voting for the first time (e.g., the City of Barrie), the town changed vendors between election cycles (e.g., the City of Belleville), or the vendor changed the election domain name between election cycles (e.g., Intelivote). Of the 1.39 million eligible voters in municipalities that offered online voting as the only voting method, 134 (87%) cities used a new URL compared to the 2018 election. Although a formal user study was outside the scope of this work, we hypothesize that without a prior history of engaging with the genuine voting site's URL, voters are less likely to notice a domain redirection at login.

### 4.2 Additional Aggravating Factors

Our analysis of municipal websites and associated social media found that cities rarely share the URL except in the VIL. We found no examples of VILs containing QR codes. Therefore, we generally expect voters to access the voting website by manually typing the URL into their browser's address bar. Two of the four sample VILs we examined in our study did not include the protocol (`https://`) as part of the URL. Neither of the two municipalities that included a complete URL on the VIL provided explicit instructions to voters to ensure they type "`https://`". For these reasons, we expect most voters would type a partial URL (e.g., `voting-site.com`) into their browser address bar. Browsers generally interpret this to mean the user is requesting an insecure HTTP resource, i.e., http://voting-site.com.

HTTP Strict Transport Security (HSTS) is a policy mechanism preventing the browser from making insecure HTTP requests. However, it is not strictly enforced on first use because the policy is communicated to the browser via an HTTP response header only after the site is accessed. Because the browser is seeing the voting site's URL for the first time, it has no previously cached site security settings to apply. Some browsers may attempt to upgrade the request to secure HTTPS automatically. A network-based attacker in a machine-in-the-middle (MitM) position blocking this request will often result in the browser downgrading to an insecure HTTP to maintain compatibility with legacy HTTP-only websites. Major browsers try to address this gap by maintaining a preloaded list of domains enforcing an HSTS policy, upgrading insecure HTTP connections for any domain on this list. Previous work has noted the relatively widespread prevalence of this gap in the online voting context [13].

### 4.3 Server-Side Attack Detectability

Based on the factors described above, an attacker has a window of opportunity to hijack the voting session during the browser's initial insecure HTTP request. At this point, the attacker could redirect the voter to a malicious site (made to look and function like the genuine voting website) and harvest the login credentials

the voter (unwittingly) types in. However, in this basic approach, the attacker has to cast the malicious vote on the voter's behalf using the voter's harvested credentials. However, in our conversation with one of the vendors, they pointed out that if numerous ballots were cast from a single attacker-controlled IP or were cast from IPs outside expected IP address ranges, suspicion would be raised on the server's side. A more sophisticated version of this attack could reduce its forensic footprint by maintaining the direct interaction between the voter's device (and hence IP) and the voting server. A cross-site browser framing attack, which frames the legitimate voting website inside an outer attacker-controlled domain, would preserve the apparent browser-server interaction.

**Table 6.** Vulnerability Analysis of Voting Services

| Vendor | Insecure Redirects | Site Framing |
|---|---|---|
| Dominion | Yes | No |
| Intelivote | Yes | No |
| Neuvote | Yes | Yes |
| Scytl | Yes | Yes (Fixed) |
| Simply Voting | No | No |
| Voatz | Yes | No |

### 4.4 Vulnerability Discovery

In late September 2022, many of the Ontario municipal voting websites started to go live. We began examining the login pages of the six vendors for any publicly observable issues. In particular, we visited each login page and examined the security headers in the response. Our analysis found that five of the six vendors were vulnerable to insecure redirects when the voter types an incomplete URL for the first time, because they were not on the HSTS preload list. However, in Scytl's case, we observed that their primary domain (`secured.vote`) did not set any security headers.[7] In particular, we observed Scytl did not set the `Strict-Transport-Security` header (requiring HSTS), and consequently, it was not on the HSTS preload list, allowing the attacker to capture and redirect the voter's initial request for the login page. Additionally, the server did not set the `X-Frame-Options` to deny cross-site framing. We also observed Neuvote did not deny cross-site framing. Our findings are shown in Table 6.

### 4.5 Exploit and Payload Testing

We tested our attack by deploying two web servers—one legitimate, one malicious. The sites had differing domain names, and TLS was enabled in both. We

[7] securityheaders.com gave secured.vote a grade of F.

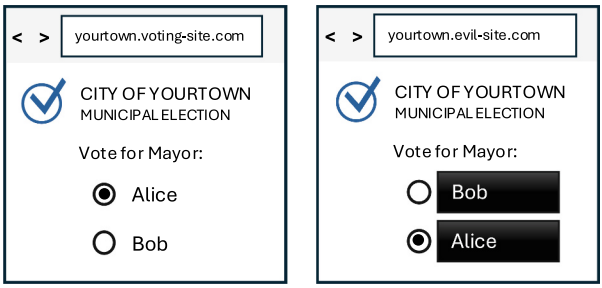


**Fig. 2.** A cross-site framing attack in the election context. **Left**: Legitimate voting website depicting a vote for Alice. **Right**: Voting website framed inside a malicious site applying cross-site overlays (black) to misdirect a voter into voting for Bob instead.

used a web proxy on a local device to simulate the attacker's MitM position and behavior. We then tested typing the partial URL of the legitimate site in Google Chrome and had the proxy block any HTTPS requests to the legitimate site. We observed the browser sending an insecure HTTP request, and configured the proxy to respond with an `HTTP 301` redirect to the malicious site. We confirmed no browser warnings appeared on the voter's end at any point in this process.[8] We then tested and confirmed the ability to frame the legitimate site inside the malicious site, establishing the basic feasibility of the exploit.

The next step was to develop a payload to modify the voter's selections. In the cross-site framing setting, an attacker's options are fortunately limited: The malicious server cannot observe the messages exchanged between the browser and voting server, and cannot directly observe or control the voter's ballot selections in the inner framed site. Based on these limitations, our idea was to apply iframe-based overlays in the malicious outer frame to switch the apparent ordering of candidate names toward misdirecting the voter into casting a ballot for the wrong person. This approach is depicted in Fig. 2. However, the attacker does not directly know which page of the inner-framed voting website the voter is currently accessing, and therefore does not know when to activate the name-swapping iframe overlays. We then considered a simple workaround: The malicious outer site provides the voter with explicit instructions to *double*-click all fields and buttons. We then used additional iframe overlays placed on the page-forward buttons to detect when the voter had advanced to the ballot page. This allowed the malicious site to activate the name-swapping overlays only on the intended page. Future work could improve on this approach. Since the adversary is already in a MitM position between the voter and the voting website, a length-based analysis of the TLS-encrypted traffic could be used to infer which page the voter is currently accessing. Previous work has demonstrated the basic feasibility of this approach in the online voting setting [14]. Putting it all together, the protocol-level view of the attack is given in Fig. 3.

[8] This was tested on Chrome 128.0.6613.114 and Firefox 129.0.2 with default settings on Windows 10.

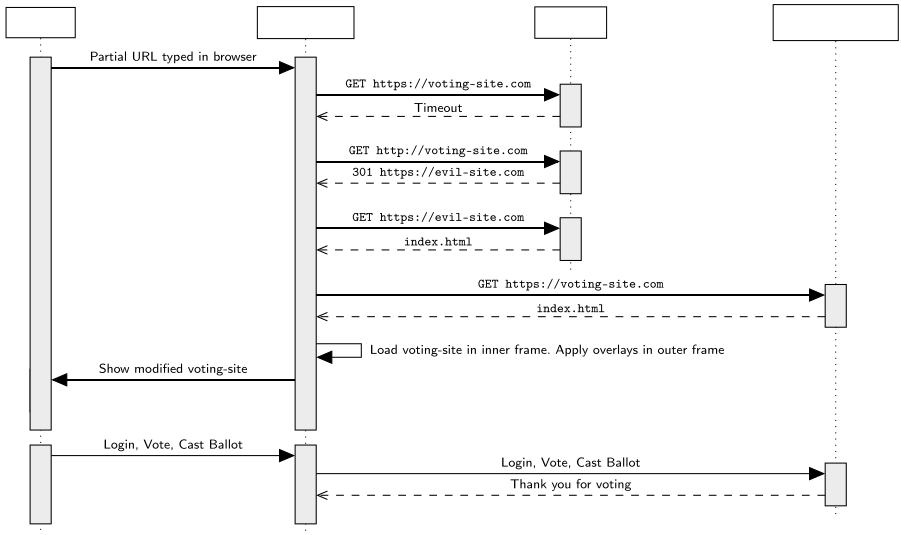


**Fig. 3.** Cross-site framing attack showing a normal voting session (from the voting-site's point of view) despite the site's presentation having been maliciously modified in the voter's browser.

### 4.6 Coordinated Disclosure and Acknowledgement

After confirming our findings, we initiated a coordinated vulnerability disclosure with Scytl. They promptly responded, and we met on October 14th to discuss the attack and possible mitigation strategies. The following day (October 15th), they enabled their server's `X-Frame-Options` headers, preventing future cross-origin framing. The same day, they sent out an advisory to their municipal clients acknowledging our findings and describing the issue and their mitigation efforts and provided us written permission to publish our results. In Neuvote's case, unfortunately we observed the behavior too late to correct it before the election. This is the first time we have presented these findings publicly, and we are unaware of any ongoing litigation efforts in Ontario pertaining to the 2022 election that would be affected by the publication of this work.

## 5 Discussion and Conclusion

We provide several recommendations to Ontario municipalities to reduce the future risk of these attacks. However, these findings may be useful to other jurisdictions as well: Most ballots cast in the 2022 municipal election were processed by companies currently running elections in the US, Europe, and elsewhere.

**Challenges to Secure Web Sessions in the Election Context.** Although the cross-site browser framing vulnerability in Sect. 4 was mitigated by the vendor enabling a single server security header, the overall attack was due to a

combination of multiple security and usability gaps. It was aggravated further by the ephemeral, one-time nature of the voting URL. A standard penetration test would likely have caught the missing security header before the voting period began. Penetration tests are generally not made available to the public. Unfortunately, we do not know if the root cause of the oversight rests with the municipal client, vendor, or pentesting company (if one occurred). The independence of these tests is also an obstacle: Small municipal clients may not have the budget to commission such a test on their own, relying on the vendor to commission their own prior to the procurement process. A recent voluntary national standard for municipal online voting provides guidance on this matter: "In addition to contracted penetration testing, terms and conditions for open-ended adversarial testing of the online voting service should be offered" (see §6.1.7, [15]). The risk of this attack can be further reduced if governments develop a unified and persistent voting portal. Future work should study whether a voter is more likely to notice a URL redirection attack against a domain they have previously used and whether this effect increases over time. At a minimum, reusing domains across election cycles could narrow the attacker's window of opportunity by allowing the voter's browser to enforce previously cached security settings.

**Challenges to Secure Voter Credentials.** The credential harvesting attack presented in Sect. 3 is inherent to the approach of automatically mailing every voter a PIN, and full mitigation will likely require a fundamental redesign of the voter authentication method. Ontario municipalities, however, are constrained in how they communicate with voters. Voter lists only contain a voter's name, address and date of birth, making the latter the sole shared 'secret' between the voter and the city. Unfortunately, there appears to be no clear pathway forward at this time. Both Canada and Ontario have no digital identity infrastructure (like in Estonia). As a near-term mitigation strategy, however, the presence of a credential harvesting attack could be made detectable if municipalities followed up with voters through an independent communication channel, acknowledging when a vote was cast in their name. For example, municipalities could mail a post-election "thank you for voting" letter and instruct the voter to contact the municipality if no such vote was cast by them. Another mitigation option is to require a secondary email-based registration step. In the meantime, we recommend cities apply the metric in Sect. 3 to determine their risk level and identify any aggravating factors in their context. Second, cities should consider ways to reduce the number of unused discarded VILs, starting with estimating the rate of insecurely discarded VILs and integrating the possibility of credential attacks into their overall election security strategy.

**Acknowledgements.** The authors thank Scytl and the anonymous reviewers. This work was supported by NSERC Discovery, SSHRC Grant No. 430-2022-01059, and Brock University's Chancellor's Chair for Research Excellence.

## References


1. Association of Municipalities of Ontario: 2022 Municipal Election Results Website. Accessed 30 Aug 2024. elections2022.amo.on.ca
2. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, pp. 61–70 (2005)
3. Kusters, R., Truderung, T.: An epistemic approach to coercion-resistance for electronic voting protocols. In: 2009 30th IEEE Symposium on Security and Privacy, pp. 251–266. IEEE (2009)
4. Delaune, S., Kremer, S., Ryan, M.: Coercion-resistance and receipt-freeness in electronic voting. In: 19th IEEE Computer Security Foundations Workshop (CSFW 2006), p. 12. IEEE (2006)
5. Clark, J., Hengartner, U.: Selections: internet voting with over-the-shoulder coercion-resistance. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 47–61. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27576-0_4
6. Kula, T.: Challenge to Lambton Shores election dismissed, The Observer (Sarnia) (2019). https://www.theobserver.ca/news/local-news/challenge-to-lambton-shores-election-dismissed
7. Cardillo, A., Akinyokun, N., Essex, A.: Online voting in ontario municipal elections: a conflict of legal principles and technology? In: Krimmer, R., Volkamer, M., Cortier, V., Beckert, B., Küsters, R., Serdült, U., Duenas-Cid, D. (eds.) E-Vote-ID 2019. LNCS, vol. 11759, pp. 67–82. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30625-0_5
8. Vallis, M.: So you gave personal info to a company caught in a data breach. Now what?, CBC News (2023). https://www.cbc.ca/news/canada/cybersecurity-consumer-protection-tips-1.6900450
9. Griffin, T.: Data breach exposed health info on pregnancies and births of 3 million Ontarians. National Post (2023). https://nationalpost.com/news/canada/perinatal-child-registry-data-breach-affects-3-million-ontarians
10. Abedi, M.: LifeLabs hack: what Canadians need to know about the health data breach. Global News (2019). https://globalnews.ca/news/6311853/lifelabs-data-hack-what-to-know/
11. Ohio secretary of state: Statewide voter files download page (2024). https://www6.ohiosos.gov/ords/f?p=VOTERFTP
12. Town of Atikokan: How to vote online in the 2022 Election (2022). URL: https://www.facebook.com/atikokantown/videos/1265289370957760/
13. Cardillo, A., Essex, A.: The threat of SSL/TLS stripping to online voting. In: Krimmer, R., Volkamer, M., Cortier, V., Goré, R., Hapsara, M., Serdült, U., Duenas-Cid, D. (eds.) E-Vote-ID 2018. LNCS, vol. 11143, pp. 35–50. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00419-4_3
14. Brunet, J., Pananos, A.D., Essex, A.: Review your choices: when confirmation pages break ballot secrecy in online elections. In: Electronic Voting: 7th International Joint Conference (E-Vote-ID), vol. 13553, LNCS, pp. 36–52 (2022)
15. Online Voting – Part 1: Implementation of Online Voting in Canadian Municipal Election (CAN/DGSI 111-1). Digital Governance Standards Institute (2024)

# Voting Under Pressure: Perceptions of Counter-Strategies in Internet Voting

Christina Nissen[1(✉)], Tobias Hilt[2], Jurlind Budurushi[3], Melanie Volkamer[2], and Oksana Kulyk[1]

[1] IT University of Copenhagen, Copenhagen, Denmark
chfn@itu.dk
[2] Karlsruhe Institute of Technology, Karlsruhe, Germany
[3] Baden-Wuerttemberg Cooperative State University, Stuttgart, Germany

**Abstract.** While internet voting can enhance democratic participation, concerns about voter coercion have emerged due to the uncontrolled voting environment. To mitigate this, researchers have proposed different types of *counter-strategies*, allowing voters to cast their intended vote despite being coerced. We conduct semi-structured interviews ($N = 26$) to investigate voters' perceptions of six types of counter-strategies concerning their effectiveness. Our findings show that the voter's perception of the effectiveness of counter-strategies depends on both the technical and personal skills of voters, concrete risks, and ease of use. Overall, our findings pave the way for future research aimed at developing user-friendly solutions that are effective against voter coercion.

**Keywords:** Coercion resistance · Counter-strategies · User study · Internet voting

## 1 Introduction

Internet voting systems are an opportunity to make elections more accessible for disabled voters and for those abroad who have limited access to physical polling places. However, implementing internet voting introduces security risks due to the uncontrolled voting environment. One particular risk is voter coercion, where an adversary could force the voter to cast their vote in a particular manner, undermining the integrity of election results [20]. In order for the coercion to be successful, the adversary needs to have a possibility to confirm that the voter is complying; otherwise, the voter could lie to the coercer while still voting according to their own wishes.

To mitigate the problem of voter coercion, various counter-strategies have been proposed [6,20,25]. The main idea behind these counter-strategies is to provide a way for the voter to deceive the coercer. If successful, the voter is thus able to either vote according to their actual intention or, at least, to cancel the vote cast during coercion, while the coercer has no possibility of detecting it.

However, the effectiveness of these counter-strategies depends on whether the voters can execute them. Previous research, however, has identified a


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 158–174, 2026.
https://doi.org/10.1007/978-3-032-05036-6_10

number of assumptions about the voters' behaviour and capabilities that the currently proposed counter-strategies rely on [22]. The empirical validation of these assumptions has been very limited so far. As such, while a few experiments investigate the usability of specific voting systems implementing a variant of proposed counter-strategies [8,27,28], no in-depth investigation has been conducted of how voters would perceive the feasibility of different types of counter-strategies and the risks connected to them in a real-world election. In this work, we address this gap by investigating the following research question:

**RQ.** How do voters perceive the feasibility of counter-strategies to ensure coercion-resistance in internet voting systems?

We conduct semi-structured interviews with a total of $N = 26$ participants from Denmark and Germany. While neither country uses Internet voting for political elections, Germany has used it in a nationwide election for the first time with the social elections 2023 [19], and the topic is regularly discussed in Denmark [21]. Thus, public interest exists in both nations. As Denmark and Germany are stable, high-trust democracies that have not implemented Internet voting broadly, they are ideal settings for studying how people perceive counter-strategies without direct exposure to Internet voting systems. This study helps identify potential barriers to adoption in countries that could implement online voting in the future but have not yet done so.

In this study, we introduce participants to six different types of counter-strategies. For each type of strategy, we ask our participants about their perceptions of these strategies' effectiveness in protecting against voter coercion, along with their assessments of the strategies' usability. We conduct a thematic analysis using an open coding approach to answer our research questions.

Our findings reveal key factors that determine the voters' perceptions of the feasibility of the counter-strategies, such as *memorability* of the counter-strategy (e.g., requiring the voter to remember secrets), personal characteristics of the voter (e.g., being able to lie convincingly to a coercer), their environment (e.g., having a trusted person), and technical skills. While some of the evaluated counter-strategies were perceived as usable, our participants were also aware of their limitations in being effective in protecting against voter coercion. Therefore, our findings highlight the need to consider the trade-offs between usability and security in voters' perceptions of the counter-strategies.

## 2 Related Work

A substantial number of counter-strategies have been offered to address the issue of voter coercion [3,4,6,7,9,11,15,17,18,23–25,34,38]. These solutions enable voters to deceive coercers by appearing to comply while ensuring that coerced votes are not counted in the final results. A classification of such counter-strategies divides them into three types – namely, *fake credentials*, *masking*, and *deniable vote updating* [22]. Furthermore, the analysis of these types has identified a number of assumptions about the voters' capabilities required to

apply these counter-strategies successfully, such as being able to remember secret credentials without writing them down and to enter these credentials correctly without making a mistake. The study concludes that usability issues might prevent these counter-strategies from being effective; however, without conducting an empirical investigation to validate these conclusions. Further works have focused specifically on usability of coercion-resistant voting in proposing their own solutions [14,29,30], however, these solutions have not been empirically evaluated.

To the best of our knowledge, only a few user studies have evaluated the usability of counter-strategies in coercion-resistant voting, namely, investigating the counter-strategies based on *fake credentials* [8,27,28]. Cristiano et al. [8] report a mean system usability scale (SUS) score of 84.9, a measurement for assessing perceived usability, with all participants managing to complete their tasks within ten minutes. However, the study finds that the participants write down their credentials, reducing the offered protections against coercion resistance. Merino et al. [27] find a mean SUS score of 70.4, with 95% of the participants exposed to fake credentials understanding their use. However, they report that 10% of these participants mistakenly voted using a fake credential, which in a real-world election would result in these votes not being counted.

## 3 Counter-Strategies

To identify the full scope of available counter-strategies, we extend a systematic literature review conducted in 2020 that identified three types of counter-strategies: fake credentials, deniable vote updating, and masking [22] . We apply the same methodology to identify counter-strategies that have been proposed after the initial literature review has been conducted, i.e., covering papers published from 2020 to 2024. We identify three new counter-strategies: *decoy tokens*, *flexible vote updating*, and *signal-based nullification*. An overview of all six types of counter-strategies is provided below.

*Fake Credentials* [6,10,20]. During voter registration in a controlled environment, the voter receives a credential, which they use to authenticate when voting. Under coercion, the voter uses a fake credential, which is accepted by the system but results in a vote being excluded from the tally. This allows the voter to cast a true vote when unobserved.

*Masking* [37,38]. During voter registration in a controlled environment, the voter receives a masking value, which they use to mask their ballot when voting, allowing the system to later extract the vote. Under coercion, the voter provides a fake masking value, making the vote appear to follow the coercer's instructions.

*Deniable Vote Updating* [17,25]. Under coercion, the voter casts the vote according to the instructions. When unobserved, the voter returns to cast their true vote.

*Decoy Tokens* [24,34]. During voter registration in a controlled environment, the voter receives a set of tokens, one valid and the rest fake. Under coercion, the voter assigns a fake token according to the instructions, while the valid token is assigned according to their true preference, which is tallied.

*Flexible Vote Updating* [15]. In one scenario, the voter updates a coerced vote by adding the correct references to previous cast ballots together with the new ballot. In another, if the voter has cast a valid vote before coercion, the voter provides incorrect references together with the ballot, preventing the coerced vote from being tallied.

*Signal-Based Nullification* [7]. During voter registration in a controlled environment, the voter registers YES/NO public keys and keeps the corresponding private keys. They can share these with helpers and agree on a signal in case of coercion. Under coercion, the vote is cancelled via private key proof, either by the voter or a helper, following the voter's signal. This strategy only enables cancellation of a coerced vote.

## 4 Methodology

We describe the interview guide, recruitment, ethical considerations, and the data analysis in our study.

### 4.1 Interview Guide

The interview guide was prepared in three languages: English, German, and Danish. The English version was prepared first and iterated in three pilot studies. The translations into the other two languages were done by the paper authors, who were fluent in these languages. To further validate the consistency among the translated versions, the translations were translated back to English using a large language model, and the output was compared with the original interview guide in English to detect any flaws with the translation.

The guide was structured as follows (see Appendix A)[1]. First, the participants were told about the purpose of the study and asked to read and sign a consent form (see Appendix A). The participants were furthermore provided with a one-sentence description of voter coercion, namely, 'Coercion in voting happens when someone pressures or forces you to vote in a way they want you to vote,' to guarantee a common understanding of the research topic we are investigating.

To address our research question, the participants were shown a description for each one of the six counter-strategies (see Sect. 3), prepared in lay terms (see Fig. 1 as an example for such a description).

The descriptions were presented to the participants one at a time in random order, and the participants were asked the following questions for each counter-strategy:

[1] The interview guide includes additional questions beyond the scope of this paper; here, we focus exclusively on questions related to counter-strategies.

**4.Deniable vote updating:**

1. Casting Your Vote: When you vote, you authenticate your identity before being able to vote. You then cast your vote for your preferred candidate.

2. Dealing with Coercion: If someone pressures you to vote a certain way, then you cast your vote as they instruct you to. Once the person pressuring you is gone, you can go back and cast another vote for your preferred candidate. You can vote as many times as you want. The voting system will update your vote if you follow the procedure as described.

**Fig. 1.** Description of deniable vote updating

– Do you believe this counter-strategy could work against voter coercion? Why/Why not?
– Do you think this is doable for a regular voter? Why/Why not?

Furthermore, we asked the participants to rank the six counter-strategies and justify their reasoning[2].

Finally, we included questions on socio-demographic information, such as gender, age, and level of education. Additionally, we included a scale-based question to assess participants' IT skills.

### 4.2 Recruitment and Ethics

For recruitment, we used a combination of purposive and convenience sampling [5]. The study was conducted between July and early September 2024, shortly after the EU election, which took place between June 6 and 9, 2024. Participants were purposively selected based on three criteria: (1) being between 18 and 30 years old, (2) either holding a university-level degree or being in the process of completing one, (3) being eligible to vote in EU elections, and (4) participants without specialised technical knowledge. The first two criteria were chosen since younger people with higher education are more likely to be first adopters of digital technologies; hence, in case Internet voting is introduced, this demographic is likely to be among the first ones using it [12,35]. This focus supports our study's motivation to investigate perceptions in countries like Denmark and Germany, where Internet voting has not yet been broadly implemented, helping to identify potential barriers to adoption in countries that may consider implementing online voting in the future. The third criterion was chosen given the time proximity to the EU election, reasoning that people who either voted or at least were eligible to vote in it would be more likely to pay attention to news and other media reports related to election processes; hence, they would be more likely to be aware of potential election integrity issues that might occur. Furthermore, we intentionally chose participants without any specialised technical knowledge, as Internet voting systems are intended for the general population, the majority of whom have no specialised technical knowledge.

[2] We do not include the rankings in the results section, as our primary interest is to find issues and perceptions concerning the counter-strategies rather than the rankings themselves.

Therefore, the participants' lack of technical expertise reflects the real-world user base of such systems, making their opinions relevant for evaluating the feasibility and usability of counter-strategies in practice. The recruitment for the study was conducted via flyers distributed at universities, social media, and snowball sampling [5]. We stopped recruiting new participants when data saturation was reached [16].

We coordinated the process alongside the ethical guidelines and received approval from the ethical committee of the German institution. At the Danish institution, ethical approval is not mandatory. However, we completed a privacy impact assessment for the project, which was approved by the institution's legal department. Participants from Germany received ten euros as reimbursement, aligning with the minimum wage. Denmark has no minimum wage, so the participants from Denmark received a gift card worth 13.41 euros (100 DKK) for an ice cream shop or a chocolate shop. For the Danish participants, the higher compensation is due to generally higher living costs, and the choice of providing gift cards instead of direct payment is due to institutional regulations. Prior to the interview, participants were asked to sign a consent form outlining the study's purpose, withdrawal options, and data handling procedures. Contact details for the researchers were also provided for any enquiries.

### 4.3 Data Analysis

All interviews were auto-recorded and transcribed using Amberscript and WhisperAI. Afterwards, the two interviewers reviewed the transcripts to ensure alignment with the original recordings. We used MAXQDA to support a coded thematic analysis of the interviews, employing an open coding approach to address the research question. We coded the interviews at the question level. The two researchers who conducted the interviews also carried out the coding in their respective languages. To develop the initial codebook, the two coders independently coded a randomly selected interview translated into English and discussed their findings. Subsequently, they each coded two interviews in their respective languages to identify additional codes, recognising that a single interview would not capture all potential codes. Based on these interviews, they discussed and revised the codebook. To ensure agreement between the coders, four interviews were translated into English and coded independently. An iterative process of discussion and refinement of the codebook followed, ultimately reaching a Cohen's Kappa of 0.76. Cohen's Kappa measures inter-coder reliability by assessing how much coders agree beyond what would be expected by chance, and a value of 0.76 indicates substantial agreement between the coders [31]. After the acceptable level of agreement was reached, the rest of the interviews were coded by each coder separately in their native language. To identify the themes, we grouped the codes using Post-its to visualise patterns and connections. We refer to Appendix A for the codebook.

## 5 Results

In this section, we present the results of our study. We recruited 26 participants in total, with 14 participants from Denmark and 12 from Germany. Our participants' ages ranged from 22 to 30, 14 of our participants were men, and 12 were women. We refer to Appendix A for more detailed demographics. In the following, we refer to the participants using the notation P# (e.g., P1 says: "quote").

We identified four key themes: system attributes, voter capabilities, risks, and security and privacy considerations.

### 5.1 System Attributes

The attributes of the counter-strategy (and, by extension, the voting system as a whole) that determine its feasibility are *convenience*, *closeness to established practices*, *flexibility*, *non-coercive cases*, and *general feasibility in practice*. Table 1 provides an overview of the identified attributes for each counter-strategy.

**Table 1.** System attributes for counter-strategies, as reported by participants. Green (+): attribute fulfilled; Red (–): not fulfilled; Yellow (+/–): mixed views. Empty cells: no mentions.

| | Convenience | Flexibility | Practical limitations and pitfalls | Closeness to established practices | Non-coercive cases |
|---|---|---|---|---|---|
| Fake credentials | -/+ | - | | + | |
| Decoy tokens | -/+ | | - | - | - |
| Flexible vote updating | -/+ | + | - | + | + |
| Deniable vote updating | + | | | + | + |
| Masking | - | -/+ | - | - | - |
| Signal-based nullification | -/+ | - | - | | |

**Convenience** has been mentioned by all of our participants, with a total of 230 mentions. The participants perceived strategies requiring physical visits (e.g., fake credentials) or multiple complex steps to complete them (e.g., masking) as inconvenient. P18 says: *"Well, because there are too many steps involved. It requires too much for people to bother voting, I would say."*

**Closeness to established practices** is an attribute that covers how close the strategy is to the established voting practices and digital platforms. Eight participants mentioned this attribute 17 times. P16 says: *"Because it seems like something you would normally do in connection with public authorities, that you log in with your ID. These are procedures that you have gone through before for something similar."*

**Flexibility** covers whether a counter-strategy is flexible to different scenarios of coercion. Seven participants mentioned this 14 times. The participants perceived flexible vote updating as flexible. P26 says: *"It gives you the flexibility to vote both before and after someone has blackmailed you into voting."* In contrast, fake credentials and signal-based nullification are perceived as less flexible.

The feasibility of a counter-strategy when the voter casts their ballot without being under any coercion is captured with **non-coercive cases**. Nine participants mentioned this 13 times and perceived some strategies negatively and others positively in non-coercive cases. P16 says: *"If you are not subjected to pressure, then you only need to vote once."*

**Practical limitations and pitfalls** is an attribute that covers how participants perceived the real-world viability of the counter-strategies. It encompasses factors such as the presence of significant gaps or pitfalls, as well as strategies that may appear effective in theory but fail to deliver in practice. Ten participants mentioned this 17 times, and they especially perceived signal-based nullification as incomplete, with 12 mentions. P24 says: *"So, it sounds like it was developed somewhat in an academic setting. It sounds great, but it doesn't work in practice."*

### 5.2 Voter's Capabilities

The feasibility of the counter-strategies is perceived to depend on voters' own capabilities, namely, their *memorisation skills*, *understanding skills*, and their *assertiveness* and *technical skills*.

**Table 2.** Voter capabilities required for counter-strategies, as reported by participants. Green (-): skill not required; Red (+): skill required; Yellow (+/-): mixed views. Empty cells: no mentions.

| | Memory | Understanding | Assertiveness | Technical |
|---|---|---|---|---|
| Fake credentials | + | -/+ | + | - |
| Decoy tokens | + | -/+ | + | -/+ |
| Flexible vote updating | + | -/+ | + | - |
| Deniable vote updating | - | - | - | - |
| Masking | + | + | + | |
| Signal-based nullification | + | + | | |

The success of almost every counter-strategy relies on the voter's ability to remember specific elements such as credentials, tokens, or other components (Table 2). **Memory skills** were mentioned a total of 119 times by 25 participants. Moreover, they considered alphanumerical passwords harder to recall than visual symbols (e.g., coloured shapes). P13 says: *"Remembering symbols is easier for people; I think it is better than the password. It is easier for people to remember whether it was a triangle, an orange triangle, or a blue circle."*

Some strategies also require high skills in understanding complex procedures (e.g., masking). 14 participants mentioned **understanding skills** 35 times, 14 of which are about masking. P19 says: *"Well, I have a hard time fully understanding the addition of numbers; maybe others will too."*

The success of most counter-strategies relies on the voter's **assertiveness**, especially the ability of the voter to lie convincingly enough to deceive the

coercer, which was mentioned by 20 participants 64 times. P14 says: *"But I also think it requires a bit of the person who votes. They also have to be good at lying."*

Finally, **technical skills** of the voters were mentioned as an important factor in counter-strategy feasibility (mentioned by 12 participants 23 times). Some counter-strategies (e.g., deniable vote updating) were seen as requiring fewer technical skills, while others raised mixed perceptions (e.g., decoy tokens). P23 says: *"Most people can manage drag-and-drop."*

### 5.3 Risks

The discussion of presented counter-strategies raised the risks of various mistakes disrupting one's vote, understood here as direct actions with clear negative consequences, namely, *typo mistakes*, *casting a wrong vote*, and *cancelling the vote*, as well as the risk of *last-minute coercion*.

The risk of a **typo mistake** is associated with fake credentials, which would lead to casting an invalid vote (mentioned by nine participants 13 times). P20 says: *"You think you voted, but you entered the information incorrectly, so you did not vote anyway."*

The risk of mistakenly **casting a wrong vote** (mentioned by 14 participants 34 times) is associated with decoy tokens (15 mentions), masking (15 mentions), and flexible vote updating (four mentions). P23 says: *"But it could be risky to have the possibility of voting incorrectly - even just by mistake."*

The risk of **cancelling a vote** covers several risks associated with the signal-based nullification strategy, which 15 participants mentioned 29 times. This includes the potential to cancel another person's vote by mistake because voters might fail to create a unique code or because a signal is misinterpreted. P14 says: *"There will definitely be situations where you consider something to be a signal without it being a signal."* Moreover, this code encompasses the participant dissatisfaction with the inability to recast a vote once revoked, leaving their choice unrepresented.

The risk of **last-minute coercion** (mentioned by 16 participants 26 times) is associated with deniable (19 mentions) and flexible (seven mentions) vote updating. P10 says: *"If the coercer stays with you until the end of the election and you can't change your vote, it is problematic."*

### 5.4 Security and Privacy Considerations

The security and privacy aspects of the discussed counter-strategies, reflecting perceptions of potential vulnerabilities or safeguards rather than direct consequences, include *coercer's capabilities and knowledge*, *security of knowledge-based secrets*, *writing down knowledge-based secrets*, *system feedback*, *depending on others*, and *voting several times*.

The **coercer's capabilities and knowledge** are security considerations associated with all counter-strategies, and 25 participants mentioned it a total

of 122 times. The participants considered it to be public knowledge how the counter-strategies work, which they believed the coercer will use to their advantage. This could be by pressuring the voter to reveal their knowledge-based secret or waiting until the last minute to coerce. Additionally, they believed that the coercer is likely to suspect the voter of using a counter-strategy. P16 says: *"I think if this is known to everyone, it might not work as well because they would know that you are getting that code, and they would start trying to pressure you to give this code to them."*

The **security of knowledge-based secret** is an aspect associated with fake credentials, decoy tokens, and masking, and 18 participants mentioned this 37 times. The participants considered a strategy to be effective because the voter is the only one knowing their secret. P23 says: *"Well, again, here you have something that only you know. I think that has proven to be a reasonably secure solution."*

**Writing down knowledge-based secrets** is associated with several strategies, and the participants mentioned it 35 times (19 mentions for fake credentials, five for masking, four for decoy tokens, and two for flexible vote updating and signal-based nullification). The participants perceived a strategy to be less secure if the voter is writing down their knowledge-based secret. P8 says: *"You would write it down somewhere. When you are not prepared and someone comes in while you are voting and the password is there, the coercer could just vote for you directly."*

The importance of providing only limited **system feedback** was mentioned by 16 participants 29 times, in association with fake credentials, masking, and deniable and flexible vote updating. In particular, lack of feedback was deemed essential for successfully deceiving the attacker, as explained by P1: *"The attacker has to trust you because the system does not give any feedback."*

The security issues of being allowed to **vote several times** are associated with deniable and flexible vote updating, mentioned by 12 participants 105 times. The participants expressed concerns about overloading the server and the possibility of coercion happening over a longer time period. P24 says: *"If we say all five million Danes think it is super fun to vote all the time, it could overload the system, and then it collapses."* However, they noted positively that voting multiple times allows voters to cast their intended vote in a more secure situation.

**Depending on others** is considered to be a privacy and security concern related to signal-based nullification (mentioned by 23 participants 57 times), especially in the context of family coercion. The participants expressed concerns that not everyone trusts others, and the coercers could be family members. P26 says: *"I believe coercion often comes from people you know rather than strangers. And I think that would almost give people a free pass to have your vote cancelled, more than it would actually protect you from coercion."*

## 6 Discussion

**Usability vs. Security.** Our participants' opinions on the counter-strategies revealed a conflict between usability and security. As such, while they men-

tion memorability as a potential issue for the success of almost every counter-strategy, they nonetheless emphasise the importance of keeping a **knowledge-based secret** (e.g., masking value) only known to the voter themselves for security reasons. This trade-off is a known issue in usable security and is exacerbated for coercion-resistant voting due to potential issues with storing the secret somewhere where the coercer can demand access. This issue has furthermore been noted by our participants: as such, while our participants suggested writing down or taking a picture of their knowledge-based secret to remember it, they also consider this a security issue if the attacker accesses the stored secret by finding or obtaining it through coercion. Previous research on the usability of **fake credentials** avoided this problem by letting the participants write down their credentials [8]. Hence, future work needs to focus on designing and evaluating other techniques either for the voters to remember their secrets without writing them down or for developers of the system to provide storage options not accessible to the attacker even in case of coercion. Such studies can furthermore be beneficial for all counter-strategies relying on memorability, which, according to our participants, applies to all except deniable vote updating.

A further example of the conflict between usability and security relates to **deniable vote updating**. Most of our participants find this strategy to be the easiest to apply, but note the risk of last-minute coercion. Since this risk would not be an issue for the other counter-strategies, the effectiveness of the deniable vote updating strategy in protecting against coercion can be seen as weaker. For real-world elections, decisions, therefore, need to be made on a case-by-case basis to determine whether the last-minute coercion risk outweighs the benefits of an overall simplicity of the counter-strategy. An open question that furthermore remains is how to communicate the relevant risks, in particular, the capacities of a potential coercer, to both voters and decision-makers, particularly in relation to the scalability of an attack. The risk of last-minute coercion is furthermore mentioned as an issue for **flexible vote updating**, despite this counter-strategy actually having mechanisms available to protect against this attack. Such a misconception highlights an issue with the strategy's understandability, and it remains an open question of how to explain the strategy without misrepresenting the risks. Previous research has shown that understandability directly influences how transparent an internet voting system is perceived to be [1]. Transparency, which is widely recognised by both researchers and policymakers as a key factor in building trust and fostering acceptance of internet voting systems [2,13,26,36]. Therefore, the understandability is not only important for their effectiveness but also for the broader acceptance of the voting system as a whole.

In addition to usability issues, our participants pointed to the **personal circumstances of the voter** regarding their environment and social circle. Participants are concerned about having to rely on others in **signal-based nullification** to be effective against coercion – either because the voters might be socially isolated and lack close trusted contacts or because the helpers themselves could potentially be the coercers. Consequently, they perceive signal-based nul-

lification as ineffective against voter coercion in such situations. Deciding to use a particular counter-strategy should, therefore, take into account such personal circumstances, and studies need to be conducted to get a thorough understanding of the population that is of particularly high risk of coercion.

Participants further emphasise that voters need certain **resources and skills**, in particular the ability to lie to the coercer, which is required in most cases according to our participants. As such, when discussing using a **knowledge-based secret** in front of the coercer (e.g., by authenticating with fake credentials), the participants highlight that individuals who are not comfortable with deception might struggle in convincing the coercer that they are indeed using the right secret. This raises an important point about the psychological and social challenges of implementing such counter-strategies, suggesting that personal traits could influence the effectiveness of knowledge-based secrets in preventing coercion. Still, it remains an open question of how the voter's assertiveness impacts the effectiveness of counter-strategies in practice.

**Role of Public Information.** Applicable to all counter-strategies, the participants consider publicly available information about the procedures voters have to perform to apply these counter-strategies to be a problem for their effectiveness. As such, our participants note that for counter-strategies implementing a knowledge-based secret, the coercer might try to be present during the disclosure of the knowledge-based secret or use extreme threats to make the voter reveal it. For counter-strategies vulnerable to last-minute coercion, the coercer might attack right before the deadline, making it impossible to update the vote. In general, the coercer might suspect that the voter is trying to deceive them if they know that an Internet voting system has implemented a counter-strategy, making it harder for the voter to convincingly deceive a coercer. On the other hand, information about available counter-strategies might discourage the coercer from attempting coercion. Namely, as the coercer will have no way of knowing for sure whether the voter complied with the coercer's instructions, they might decide that the risks they face attempting coercion are not worth the uncertainty of the result. Furthermore, making information about the voting system publicly available aligns with recommendations from experts [2,13,26,36] and is perceived as improving election transparency and trust by the voters [1]. The specific trade-offs introduced by public information about counter-strategies therefore remain an open question, and an open challenge is how to design Internet voting systems that maintain transparency without making counter-strategies ineffective.

*Design implications.* Based on the identified challenges, we propose the following design implications, which we, as authors, have derived from our findings to help mitigate the concerns raised by our participants about the counter-strategies:

**Visual Secrets (Fake Credentials, Decoy Tokens, Flexible Vote Updating, Signal-Based Nullification).** To support memorability, voting systems should consider using visual secrets, such as images or icons, instead of complex passwords or long codes. Research shows that pictures are generally easier for

users to recall than alphanumeric information, which can improve both usability and the effectiveness of the counter-strategies [32].

**Multimodal Explanations (All Counter-Strategies).** To enhance understandability, voting systems should support various media formats for explaining the counter-strategies, including text, audio, and video (e.g., demo videos). Providing explanations in multiple formats can help accommodate different learning styles and improve comprehension among non-expert voters [33].

**Fallback to Physical Voting (All Counter-Strategies).** To accommodate voters who may struggle with assertiveness, such as lying convincingly to a coercer, systems should allow users to override their online vote by casting a final vote at a physical polling station. This provides a secure, controlled environment for voters who feel unsafe or uncertain using coercion-resistant strategies online.

**Private Browsing or Incognito Mode by Default (Fake Credentials, Deniable and Flexible Vote Updating).** To prevent system feedback, such as browser history or cached data, from being used by a coercer to verify whether the voter has revisited the voting platform (and potentially changed their vote), the system should either prompt users to re-access the platform via private browsing mode after an initial vote, or automatically enforce this mode. This helps obscure traces of revisiting the system, thereby supporting plausible deniability and increasing coercion-resistance.

**Restricting Copying of Secrets (Fake Credentials, Decoy Tokens, Masking, Flexible Vote Updating).** To reduce the risk of voters easily copying and storing their knowledge-based secrets, the voting system should prevent simple extraction. For example, secrets (e.g., credentials) can be rendered as non-selectable images rather than text, which can be easily marked and copied using shortcuts. This subtle design choice can help by making it difficult for the voters to write down and store their secret, thus making it harder for the coercer to find or demand access to them.

**Waiting Period to Prevent System Overload (Deniable and Flexible Vote Updating).** To avoid overloading systems or potential abuse, the system should implement a waiting period after a certain number of rapid votes (e.g., five votes in quick succession).

**Non-disclosing Prompts (All Counter-Strategies).** To avoid user confusion and reduce the likelihood of errors (e.g., typos), the interface should include non-disclosing prompts such as: *"For your privacy, the system will not indicate whether your login was successful or not. Please double-check your entry before submitting."*

*Limitations.* Our findings may be subjected to bias due to the participants' lack of experience with Internet voting. To address this bias, future studies should explore how voters from countries such as Estonia, with an Internet voting system utilising deniable vote updating, perceive the counter-strategies. Another

limitation is that all the participants are between 22 and 30 years old with a university-level degree or are in the process of completing one. This limits the generalisability of our findings, but since younger people with high education are more likely to be first adopters of digital technologies, hence, in case Internet voting is introduced, this demographic is likely to be among the first ones using it [12,35]. Still, future work should investigate perceptions among older voters and people with disabilities, as Internet voting often is proposed to enhance accessibility and participation for these groups. While the study provides valuable insights into voters' perceptions of the counter-strategies, its ecological validity is limited because the participants did not use them in an actual voting system. This may have affected their assessments of the strategies' feasibility and usability, so their responses might not fully reflect real-world reactions. Nonetheless, explanations of counter-strategies would likely be provided in any implemented internet voting system, suggesting the insights gathered here remain valuable.

## 7 Conclusion

We conducted 26 semi-structured interviews to explore how voters perceive the counter-strategies designed to resist coercion in Internet voting. From our findings, we conclude that voters prefer a counter-strategy that is easy to use, namely, the **deniable vote updating**, which further has the advantage of leaving the voting process unchanged in the absence of coercion. However, the counter-strategy also introduces the risk of last-minute coercion. Thus, for elections with a high risk for voter coercion, a more complex counter-strategy should be used. Furthermore, we conclude that regardless of how **signal-based nullification** is implemented, the voters are unlikely to accept the fundamental concept of having helpers cancel their coerced vote. Voters want to be able to cast an uncoerced vote rather than only cancelling a coerced vote, and they are generally reluctant to involve others in the inherently private act of voting. Future work should focus on improving and evaluating the usability of counter-strategies.

**Acknowledgments.** The work reported in this paper was supported by Independent Research Fund Denmark, grant number 10.46540/2101-00006B.

## A Appendix

Due to the extensive length the interview guide, consent form, codebook, and demographics are provided in a single external link, accessible here: https://osf.io/3cd2r/?view_only=cfa43eae0e00457fb144c56e7ae6ca5c.

## References


1. Agbesi, S., Budurushi, J., Dalela, A., Nissen, C., Kulyk, O.: How to increase transparency and trust in internet voting systems: an experimental study. In: Proceedings of the 13th Nordic Conference on Human-Computer Interaction. NordiCHI '24, Association for Computing Machinery, New York, NY, USA (2024)
2. Agbesi, S., Dalela, A., Budurushi, J., Kulyk, O.: "What will make me trust or not trust will depend upon how secure the technology is": factors influencing trust perceptions of the use of election technologies. E-Vote-ID **2022**, 1 (2022)
3. Aranha, D.F., Battagliola, M., Roy, L.: Faster coercion-resistant e-voting by encrypted sorting. Cryptology ePrint Archive, Paper 2023/837 (2023)
4. Araújo, R., Ben Rajeb, N., Robbana, R., Traoré, J., Youssfi, S.: Towards practical and secure coercion-resistant electronic elections. In: Heng, S.H., Wright, R.N., Goi, B.M. (eds.) Cryptology and Network Security, pp. 278–297. Springer, Berlin Heidelberg, Berlin, Heidelberg (2010)
5. Bjørner, T.: Why 'Qualitative Methods for Consumer Research'?, pp. 11–15. Hans Reitzels Forlag, Denmark (2015)
6. Chaieb, M., Yousfi, S.: LOKI vote: a blockchain-based coercion resistant E-voting protocol. In: Themistocleous, M., Papadaki, M., Kamal, M.M. (eds.) EMCIS 2020. LNBIP, vol. 402, pp. 151–168. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63396-7_11
7. Chaum, D., et al.: VoteXX: a solution to improper influence in voter-verifiable elections. Cryptology ePrint Archive, Paper 2022/1212 (2022)
8. Christiano, L., Longo, R., Spadafora, C.: Click and cast: assessing the usability of vote app. In: Electronic Voting: 9th International Joint Conference, E-Vote-ID 2024, Tarragona, Spain, October 2–4, 2024, Proceedings (2024)
9. Clark, J., Hengartner, U.: Selections: internet voting with over-the-shoulder coercion-resistance. In: Danezis, G. (ed.) Financial Cryptography and Data Security, pp. 47–61. Springer, Berlin Heidelberg, Berlin, Heidelberg (2012)
10. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: toward a secure voting system. In: 2008 IEEE Symposium on Security and Privacy (SP 2008), pp. 354–368 (2008)
11. Cortier, V., Gaudry, P., Yang, Q.: Is the JCJ voting system really coercion-resistant? Cryptology ePrint Archive, Paper 2022/430 (2022)
12. Ehin, P., Solvak, M., Willemson, J., Vinkel, P.: Internet voting in Estonia 2005–2019: evidence from eleven elections. Gov. Inf. Q. **39**(4), 101718 (2022)
13. of Europe, C.: Recommendation cm/rec(2017)5[1] of the committee of ministers to member states on standards for e-voting
14. Feier, C., Neumann, S., Volkamer, M.: Coercion-resistant internet voting in practice. In: Informatik 2014, pp. 1401–1414. Gesellschaft für Informatik e.V., Bonn (2014)
15. Giustolisi, R., Garjan, M.S., Schuermann, C.: Thwarting last-minute voter coercion. Cryptology ePrint Archive, Paper 2023/1876 (2023)
16. Guest, G., Bunce, A., Johnson, L.: How many interviews are enough? An experiment with data saturation and variability. Field Meth. **18**, 59–82 (02 2006)
17. Haines, T., Mueller, J.: How not to VoteAgain: pitfalls of scalable coercion-resistant e-voting. Cryptology ePrint Archive, Paper 2020/1406 (2020)
18. Haines, T., Müller, J., Querejeta-Azurmendi, I.n.: Scalable coercion-resistant e-voting under weaker trust assumptions. In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, pp. 1576–1584. SAC '23, Association for Computing Machinery, New York, NY, USA (2023)

19. Hilt, T., Kulyk, O., Volkamer, M.: German social elections 2023: An overview and first analysis. In: E-Vote-ID 2023. Lecture Notes in Informatics - Proceedings, Gesellschaft für Informatik (GI) (2023), 46.23.01; LK 01
20. Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections, pp. 37–63. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
21. Kristensen, P.K., Pedersen, L.E.M.: Under corona har vi klaret alt online – derfor kan du ikke stemme digitalt (2021). https://www.dr.dk/nyheder/politik/kommunalvalg/under-corona-har-vi-klaret-alt-online-derfor-kan-du-ikke-stemme. Accessed 06 May 2025
22. Kulyk, O., Neumann, S.: Human factors in coercion resistant internet voting – a review of existing solutions and open challenges (2020)
23. Locher, P., Haenni, R., Koenig, R.E.: Coercion-resistant internet voting with everlasting privacy. In: Clark, J., Meiklejohn, S., Ryan, P.Y., Wallach, D., Brenner, M., Rohloff, K. (eds.) Financial Cryptography and Data Security, pp. 161–175. Springer, Berlin Heidelberg, Berlin, Heidelberg (2016)
24. Longo, R., Spadafora, C.: Amun: Securing e-voting against over-the-shoulder coercion. Cryptology ePrint Archive, Paper 2021/851 (2021)
25. Lueks, W., Querejeta-Azurmendi, I., Troncoso, C.: VoteAgain: a scalable coercion-resistant voting system. In: 29th USENIX Security Symposium (USENIX Security 20), pp. 1553–1570. USENIX Association (2020)
26. Marky, K., Gerber, P., Günther, S., Khamis, M., Fries, M., Mühlhäuser, M.: Investigating State-of-the-Art practices for fostering subjective trust in online voting through interviews. In: 31st USENIX Security Symposium (USENIX Security 22), pp. 4059–4076. USENIX Association, Boston, MA (2022)
27. Merino, L.H., et al.: E-vote your conscience: perceptions of coercion and vote buying, and the usability of fake credentials in online voting. In: 2024 IEEE Symposium on Security and Privacy (SP), pp. 3478–3496 (2024)
28. Neto, A.S., Leite, M., Araújo, R., Mota, M.P., Neto, N.C.S., Traoré, J.: Usability considerations for coercion-resistant election systems. In: Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems. IHC '18, Association for Computing Machinery, New York, NY, USA (2018). 10.1145/3274192.3274232, https://doi-org.ep.ituproxy.kb.dk/10.1145/3274192.3274232
29. Neumann, S., Feier, C., Volkamer, M., Koenig, R.: Towards a practical JCJ / civitas implementation. Cryptology ePrint Archive, Paper 2013/464 (2013)
30. Neumann, S., Volkamer, M.: Civitas and the real world: Problems and solutions from a practical point of view. In: 2012 Seventh International Conference on Availability, Reliability and Security, pp. 180–185 (2012)
31. O'Connor, C., Joffe, H.: Intercoder reliability in qualitative research: debates and practical guidelines. Int. J. Qual. Meth. **19**, 1609406919899220 (2020)
32. Paivio, A., Rogers, T.B., Smythe, P.C.: Why are pictures easier to recall than words? Psych. Sci. **11**(4), 137–138 (1968). https://doi.org/10.3758/BF03331011
33. Song, H., Healey, J., Siu, A.F., Wigington, C., Stasko, J.: Experts prefer text but videos help novices: an analysis of the utility of multi-media content. In: Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems, pp. 1–9. CHI '23, ACM (2023). https://doi.org/10.1145/3544549.3585900
34. Spadafora, C., Longo, R., Sala, M.: Coercion-resistant blockchain-based e-voting protocol. Cryptology ePrint Archive, Paper 2020/674 (2020)
35. Vassil, K., Solvak, M., Vinkel, P., Trechsel, A.H., Alvarez, R.M.: The diffusion of internet voting. usage patterns of internet voting in Estonia between 2005 and 2015. Govern. Info. Quart. **33**(3), 453–459 (2016)

36. Volkamer, M., Spycher, O., Dubuis, E.: Measures to establish trust in internet voting. In: Proceedings of the 5th International Conference on Theory and Practice of Electronic Governance, pp. 1–10 (2011)
37. Wen, R., Buckland, R.: Masked ballot voting for receipt-free online elections. In: Ryan, P.Y.A., Schoenmakers, B. (eds.) E-Voting and Identity, pp. 18–36. Springer, Berlin Heidelberg, Berlin, Heidelberg (2009)
38. Zhaoju, Z., Hanbo, L., Hong, D.: Verifiable receipt-free electronic voting system based on mask ballot. In: 2021 IEEE 9th International Conference on Smart City and Informatization (iSCI), pp. 47–52 (2021)



# Dice, but Don't Slice: Optimizing the Efficiency of ONEAudit

Jacob V Spertus[1(✉)], Amanda Glazer[2], and Philip B. Stark[1]

[1] University of California, Berkeley, USA
{jakespertus,pbstark}@berkeley.edu
[2] University of Texas, Austin, USA
amanda.glazer@austin.utexas.edu
https://www.berkeley.edu, https://www.utexas.edu

**Abstract.** ONEAudit provides more efficient risk-limiting audits than other extant methods when the voting system cannot report a cast-vote record linked to each cast card. It obviates the need for re-scanning; it is simpler and more efficient than 'hybrid' audits; and it is far more efficient than batch-level comparison audits. There may be room to improve the efficiency of ONEAudit further by tuning the statistical tests it uses and by using stratified sampling. We show that tuning the tests by optimizing for the reported batch-level tallies or integrating over a distribution reduces expected workloads by 70–85% compared to the current ONEAudit implementation across a range of simulated elections. The improved tests reduce the expected workload to audit the 2024 Mayoral race in San Francisco, California, by half—from about 200 cards to about 100 cards. In contrast, stratified sampling does not help: it increases workloads by about 25% on average.

## 1 Introduction

*Risk-limiting audits* (RLAs) [24] are a key ingredient in *evidence-based elections* [2,25], which produce affirmative evidence that the reported winners really won. By manually inspecting a random sample from a trustworthy record of validly cast votes [1], an RLA guarantees that wrong reported outcomes[1] will, with high probability, be caught and corrected before they become final. If routinely employed, RLAs could support trustworthy elections and earn voters' trust in the electoral process.

This paper aims to reduce barriers to routine RLAs by lowering the workload of ONEAudit [27] (described later in this section), currently the most flexible, transparent, and efficient RLA method.[2] We show that stratifying the sample

[1] Contest winners, not exact tallies.

[2] ONEAudit (overstatement-net-equivalent audit) [27] creates reference values for ballot cards from batch or contest totals when cast-vote records—digital records of the machine interpretations of individual ballot cards—are not available from the voting system. That makes it possible to conduct card-level comparison audits even


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 175–190, 2026.
https://doi.org/10.1007/978-3-032-05036-6_11

using batch information does not improve the efficiency of ONEAudit, even when the sharpest known methods for stratified inference are used [19]. On the other hand, new methods to optimize the bets for testing-by-betting [31,32] can substantially reduce ONEAudit sample sizes.

The overall cost of an RLA involves a fixed cost of setting up the audit and a variable cost that depends on the number of ballot cards sampled.[3] How the election jurisdiction organizes ballot cards (the sizes of batches, whether cards are imprinted with identifiers, and similar considerations) affects the marginal cost per card. All else equal, smaller samples are less expensive and make it easier to complete an RLA during the canvass period, so reducing sample sizes may induce more jurisdictions to conduct RLAs.

When the reported outcome is wrong, an RLA is intended to lead to a full hand count. When the reported outcome is correct, the sample size required to conduct an RLA depends on many factors: the number and sizes of contests under audit; the reported tallies in those contests and the rate of errors in those tallies; how physical cards are translated into numeric populations of *assorters* and reported outcomes are translated into formal *assertions* about contest winners and losers; the availability of cast-vote records (CVRs) 'linked' to individual identifiable cards; the sampling design (including the size of the sampling unit, whether sampling probabilities are uniform across units, whether units are sampled with or without replacement, and whether the sample is stratified); and the risk-measuring functions used to check assertions.

Some details are dictated by the voting technology, legal requirements, and the particular logistics of handling and organizing ballot cards. Some are under the control of the auditor, and can be chosen to maximize the *efficiency* of the audit—often defined in terms of the expected number of cards that must be checked by hand (e.g., in Spertus [18] or Spertus et al. [19])—or to enhance the simplicity and transparency of the audit [22,27]. For instance, batch-level comparison audits may naturally fit existing hardware, auditing practices, or recount laws; card-level comparison audits (CLCAs) are more efficient, but require the voting system to produce CVRs linked to identifiable ballot cards; and stratification may be useful to accommodate laws mandating independent sampling across jurisdictions, enforce workload fairness constraints, or increase efficiency, but it introduces additional complexity into the analysis [16,19,20].

The most efficient RLA strategy is a CLCA using style-based sampling [11,12] and comparison-optimal betting test supermartingales (TSMs) [18]. But that strategy needs the voting system to produce a linked CVR for each card, which is rare in the US. For instance, in California votes are counted by a mixture of precinct count optical scan (PCOS) and central count optical scan (CCOS). Modern voting systems can produce linked CVRs for cards tabulated with CCOS

when the voting system cannot export cast-vote records uniquely linked to individual ballot cards.

[3] A *ballot* is a set of ballot *cards* cast by a single voter. Ballots are typically not kept together as an identifiable unit after being cast. The sampling unit for an RLA is a single card or a batch of cards, typically grouped by voting machine or precinct.

but not for cards tabulated with PCOS—a deliberate limitation intended to preserve voter privacy.

Historically, heterogeneous voting equipment has been accommodated by ignoring CVRs and conducting a *card-polling audit* (CPA) [15]; by using stratified sampling and conducting a *hybrid audit* [16,20,23]; or by using size-weighted sampling to conduct a batch-level comparison audit, treating CVR-linked cards as batches of size one [21].[4]

ONEAudit [27] is a more recent method of conducting RLAs of elections when linked CVRs are not available for every cast card. It compares manual interpretations of cards to reference values constructed in a way that is "overstatement-net-equivalent" to the reported results: the reference values yield the correct winner if and only if the reported results identified the correct winner. This allows ONEAudit to take advantage of linked CVRs where they are available and batch subtotals elsewhere, without the need for stratification. Stratification may still be useful to accommodate legal[5] or logistical[6] requirements, and potentially to increase efficiency, a possibility explored below.

The efficiency of ONEAudit depends in part on the heterogeneity of the votes in the batches from which the reference values are derived. If the reported votes in a batch are all for Alice, then the ONEAudit reference value for every card in the batch exactly matches how it was tabulated; and if the reported tabulation is correct, every reference value will match the human reading of the votes from the corresponding card. In that case, ONEAudit is just as efficient as a CLCA that uses linked CVRs from the voting system. In contrast, if the reported votes were 50% for Alice and 50% for Bob, no single reference value can exactly match the human reading of the votes from the corresponding card, even if the batch-level tally is perfectly accurate. Then, ONEAudit would be less efficient than a CLCA that used linked CVRs from the voting system (if those CVRs existed!), but it is still more efficient than batch-level comparison audits.

The next section formalizes the notation for RLAs, with a particular focus on ONEAudit. In Sect. 3 we review testing-by-betting and propose new betting strategies for ONEAudit. We review stratified RLAs in Sect. 4. Section 5 uses simulations to evaluate the workload of ONEAudit with different betting and stratification in a range of hypothetical elections. Section 6 estimates the sample

[4] Fuller et al. [10] proposed a method that re-scans PCOS cards as needed for the audit, and Stark [26] proposed a method that imprints PCOS cards with nonces as they are scanned. To the best of our knowledge, neither method has been used in an actual audit.

[5] Some states (e.g., California) require jurisdictions to draw their own audit samples independently, which yields a stratified sample for contests that cross jurisdictional boundaries. Other states (e.g., Colorado) draw audit samples centrally from the state as a whole, without stratification.

[6] For instance, stratification may be helpful to allow auditing to start before all cards have been tabulated by partitioning the population of cards into those tabulated as of a particular date and those tabulated later.

sizes required for ONEAudit for the 2024 San Francisco mayoral race. Section 7 concludes with a discussion of our findings and recommendations.

## 2 RLAs via Assorters and Assertions

Under the SHANGRLA framework [23], RLAs are reduced to testing claims about the means of lists of non-negative numbers. The numbers come from the actual votes on ballot cards. In particular, an *assorter* $A(\cdot)$ maps a vote to a number in $[0, u]$, where $u$ is a known upper bound that depends on the assorter in question. (The mapping may depend on auxiliary information available before the audit starts.) Let $m_i$ denote the *manual vote record* for the $i$th card: the actual votes on card $i$ as a human would read them, following voter intent rules for the jurisdiction. The true assorter mean is $\bar{A}^m := \frac{1}{N}\sum_{i=1}^{N} A(m_i)$; it is unknown unless there is a full hand count. The number of assorters involved in auditing a contest depends on the social choice function for that contest, the number of candidates, and the audit strategy.

For instance, the winners of a multi-winner plurality contest with $K$ candidates and $W$ winners can be characterized by $W(K-W)$ assorters, one for each (reported winner, reported loser) pair [23]. Each assorter takes the value 1 if the card has a vote for the reported winner in the pair, the value 0 if it has a vote for the reported loser in the pair, and the value 1/2 otherwise, so $u = 1$. If the means of all $W(K-W)$ resulting lists are all greater than 1/2, every reported winner really got more votes than every reported loser.

An *assertion* is a claim that an assorter mean is greater than 1/2. An assertion can be checked by testing its *complementary null hypothesis* that the mean is less than or equal to 1/2: to reject that hypothesis is to conclude that the assertion is true. For most social choice functions (including all scoring rules), a reported outcome is correct if and only if every assertion in a given set of assertions is true [3,23]; but for IRV contests, a reported outcome is correct if and only if every assertion in one of some number of sets of assertions is true [8,9].

Consider a single assertion $\bar{A}^m > 1/2$. Card-polling audits (CPAs, [15,24]) check the assertion directly by sampling values of $A(m_i)$. Card-level comparison audits (CLCAs, [22,24]) and ONEAudit [27] involve the difference between $A(m_i)$ and a 'reference value' $r_i$, specified before the audit starts. For instance, if a CVR $c_i$ is available for card $i$, we could take $r_i = A(c_i)$. We assume that the reference values satisfy $\bar{r} := \frac{1}{N}\sum_{i=1}^{N} r_i > 1/2$, or equivalently that the *reported assorter margin* $v := 2\bar{r} - 1 > 0$. (According to the reference values, the reported outcome is not wrong.) The *overstatement* of $r_i$, i.e., the amount by which it exceeds $A(m_i)$, is $\omega_i := r_i - A(m_i)$. The *overstatement assorter* is an affine transformation of the overstatement. Let $\bar{x} := \frac{1}{N}\sum_{i=1}^{N} x_i$. Then $\{\bar{x} \le 1/2\} \iff \{\bar{A}^m \le 1/2\}$: the two conditions are equivalent. The advantage of basing an RLA on $\{x_i\}_{i=1}^{N}$ is that it tends to have lower variance when each reference value $r_i$ is close to its corresponding assorter value $A(m_i)$, which reduces the sample size needed to confirm the reported outcome when it is correct. If every reference value is equal to its corresponding assorter value,

$x_i = (2 - v/u)^{-1}$ for every card $i$. This tends to make CLCAs far more efficient than CPAs when the statistical test is tuned appropriately [18].

$$x_i := O(m_i) = \frac{1 - \omega_i/u}{2 - v/u}, \tag{1}$$

For cards for which the voting system does not provide a linked CVR, $r_i$ can be set arbitrarily (as long as $\bar{r} > 1/2$). ONEAudit derives $r_i$ for those cards using other information the voting system can export, such as batch subtotals. In the U.S., current voting systems generally can report a CVR 'linked' to each cast card for vote-by-mail ballots, but can report only groups of CVRs for batches of ballots cast in person (e.g., cast in a particular precinct or scanned by a particular scanner), with no way to tell which CVR in the group corresponds to which card in the batch. In that situation, a reference value $r_i$ can be derived from the average of $A(c_i)$ (the average of the assorter applied to the CVRs) for cards in the batch. That is how $\{r_i\}_{i=1}^N$ were derived for in-person ballots in the pilot ONEAudit RLA in San Francisco, CA, discussed below.

When the reported assorter total for a batch is correct (i.e., when $\sum_{i \in \text{batch}} r_i = \sum_{i \in \text{batch}} A(b_i)$), the mean of the overstatement assorter in the batch is $(2 - v/u)^{-1}$. This follows from Eq. 1: the overstatement assorter is an affine transformation of the overstatement error $\omega_i$, and the average overstatement in a batch is 0 when the reported assorter total in that batch is correct. Thus, when the voting system behaves correctly, the overstatement error has the same mean in every batch, but the heterogeneity may differ across batches. In plurality contests, a batch is more homogeneous when the original reported assorter mean is near 0 (favoring the loser) or 1 (favoring the winner), or when there are many non-votes. Constructing batches to reduce heterogeneity (e.g., by reducing batch size) could improve efficiency if the statistical test can take advantage of the lower variance.

## 3 Dicing: Efficient Testing by Betting for ONEAudit

*Betting test supermartingales* (TSMs)—real-valued stochastic processes that are nonnegative supermartingales starting at 1 if the null hypothesis is true—are the basis of the most efficient, sequentially valid RLAs known [18,24,33].

Let $X_i$ denote a random draw from a population on $[0, 1]$ with true mean $\mu$. Define where $\eta_j$ is (an upper bound on) the mean of the population just before the $i$th draw if $H_0 : \mu \leq \eta$. Then $(M_t)_{t \in \mathbb{N}}$ is a TSM for $H_0$. For sampling uniformly and independently with replacement $\eta_j = \eta$; for sampling uniformly without replacement $\eta_j = (\eta - \sum_{\ell=1}^{j-1} X_\ell)/(N - j - 1)$. The TSM is the accumulated wealth of a gambler after $t$ rounds of betting that the null is true, risking $\lambda_j \eta_j \in [0, 1]$ of their wealth on the $j$th round. The bets $\lambda_j$ must be nonnegative to prevent the gambler from profiting when the draw is less than $\eta_j$ (in which case the game would be favorable rather than fair or sub-fair when the null is true), and cannot exceed $1/\eta_j$ or the gambler might go into debt, which is forbidden. The $j$th bet can depend on history $(X_1, \ldots, X_{j-1})$ but not on $X_k$ for any

$k \geq j$: it must be *predictable* or *non-anticipating*. Ville's inequality [28] implies that $P_t := 1 \wedge (1/M_t) \in [0,1]$ is a sequentially-valid $P$-value for $H_0$; that is, if the null hypothesis is true, the chance that $\inf_t(1/M_t) \leq p$ is at most $p$, for any $p \in [0,1]$.

$$M_t := \prod_{j=1}^{t} [1 + \lambda_j (X_j - \eta_j)],$$

An RLA is efficient if it stops after examining relatively few ballots when the reported outcome is correct. We measure efficiency by the expected value and 90th percentile of the number of cards the audit examines before stopping. For a given set of votes and a given set of assertions, efficiency depends on how the bets $\lambda_i$ are selected. Stark [24] and Waudby-Smith et al. [33] discuss how to select $\lambda_i$ for CPAs. The resulting tests are nearly equivalent to the Bernoulli sequential probability ratio test derived by Wald [29], but use a predictable or *a priori* "plug-in" estimate of the unknown alternative mean. Spertus [18] describes optimal betting for CLCAs, and presents predictable approximations. ONEAudit generates far more complicated assorter populations, even for simple assertions (e.g., two-candidate plurality), and efficient betting strategies for ONEAudit have yet to be explored. That is the topic we address next.

### 3.1 Kelly Optimality and *a Priori* Betting

Past implementations of ONEAudit [27] set the bets implicitly using a truncated shrinkage estimator of the assorter mean ('shrink-trunc' in ALPHA [23]) or explicitly using COBRA [18]. Both seek to approximate *Kelly optimality*, which maximizes the expected log growth of the TSM at time $j$ for a particular population:

$$\lambda_j^* := \underset{\lambda \in [0, 1/\eta_j]}{\arg\max}\, \mathbb{E}\left\{\log[1 + \lambda(X_j - \eta_j)]\right\}.$$

The expected value is with respect to the probability distribution induced by the design, for the assumed population. The *oracle* Kelly bet maximizes the expected log growth for the true population of assorter values, which is unknown in practice. An *a priori* Kelly bet maximizes the log growth for an assumed population derived from the reported tallies [33]; this betting scheme is efficient when those tallies are (approximately) correct. The truncated shrinkage estimate bet and COBRA are closed-form, approximately Kelly-optimal bets for CPA and CLCA populations, respectively, but bets optimized for those populations are not efficient for ONEAudit overstatement populations, as demonstrated below.

For a generic population and sampling design, the Kelly bet can be found numerically. In particular, if we *postulate* a finite-population $\{\tilde{x}_i\}_{i=1}^N$ and cards are drawn IID (rather than without replacement), then the *a priori* Kelly bet solves

$$0 = \frac{d}{d\lambda} \mathbb{E} \log[1 + \lambda(\tilde{X}_i - \eta)] = \sum_{i=1}^{N} \frac{\tilde{x}_i - \eta}{1 + \lambda(\tilde{x}_i - \eta)}, \tag{2}$$

and $\lambda^*$ can be found by bisection search, for instance. If the distributions of $\{x_i\}_{i=1}^N$ and $\{\tilde{x}_i\}_{i=1}^N$ are the same, the *a priori* Kelly bet is the oracle Kelly bet. However, if the distribution of $\{\tilde{x}_i\}_{i=1}^N$ differs substantially from the distribution of the true values $\{x_i\}_{i=1}^N$, the resulting bets may be far from optimal.

The same reported tallies and CVRs used to construct the reference values for the assorters may provide a good initial guess of the population of overstatements. One could use that guess to find the *a priori* Kelly bet, and commit to this bet throughout the audit. If the reported tallies have large errors (but the reported outcomes are still correct) this could be worse than strategies that do not rely on the reported results, because it anchors the bets to an inefficient choice.

### 3.2 Approximately Optimal Predictable Betting: AGRAPA

Waudby-Smith and Ramdas [31] note that the strategy in Eq. (2) could be made predictable by updating $\lambda_t$ at each time step, replacing $\tilde{x}_i$ with the sample $X_i$ and $N$ with $(t-1)$. This requires $t$ numerical optimizations, one for each time step. A more tractable bet comes from a Taylor series approximation to the derivative. The resulting strategy, AGRAPA (approximate growth rate adapted to the particular alternative), uses the bets

$$\lambda_t^{\text{ag}} := 0 \vee \frac{\bar{X}_{t-1} - \eta}{\hat{\sigma}_{t-1}^2 + (\bar{X}_{t-1} - \eta)^2} \wedge \frac{c}{\eta}, \tag{3}$$

where $\bar{X}_{t-1}$ is the lagged sample mean, $\hat{\sigma}_{t-1}^2$ is the lagged sample variance, and $c \in [0,1]$ is a user-chosen constant. The bet $\lambda_t^{\text{ag}}$ is larger when the sample mean is further above $\eta$ (the payoff is expected to be higher) and when the sample variance is smaller (the bet is closer to a sure thing). Overstatement assorter values for cards with linked CVRs have low variance if the CVRs are accurate. If most of the cards have linked CVRs, it may be efficient to set $c := 1 - \epsilon$ for some small $\epsilon$ (e.g., $\epsilon = 0.05$); this lets the bet become aggressive without betting the whole farm [18].

### 3.3 Strength in Diversity: Universal Portfolios

The seminal work of Cover [7] suggests averaging across a set of bets rather than maximizing over allowed bets. The ($f$-weighted) "universal portfolio" bet is defined as:

$$\lambda_t^{\text{up}} := \eta^{-1} \left[ \frac{\int_0^1 \gamma \prod_{i=1}^{t-1} [1 + \gamma(X_i/\eta - 1)] f_\beta(\gamma) d\gamma}{\int_0^1 \prod_{i=1}^{t-1} [1 + \gamma(X_i/\eta - 1)] f_\beta(\gamma) d\gamma} \right].$$

For a mixing density $f$ on $[0,1]$. Cover and Ordentlich [6] study taking $f$ to be a Beta$(1/2, 1/2)$ density or the uniform density. We take $f$ to be uniform. Waudby-Smith et al. [32] show that for sampling with replacement, this choice asymptotically does almost as well as the oracle Kelly bet. Calculating $\lambda_t^{\text{up}}$ is

computationally expensive and numerically unstable, involving the ratio of integrals of high-order polynomials at every time step $t$. But there is an inexpensive approximation to the resulting TSM under $\lambda_t^{\text{up}}$:

$$\prod_{i=1}^{t}[1+\lambda_i^{\text{up}}(X_i-\eta)] = \int_0^{1/\eta} \prod_{i=1}^{t}[1+\lambda(X_i-\eta)]d\lambda \approx \frac{1}{D}\sum_{j=1}^{D}\prod_{i=1}^{t}[1+\lambda_j(X_i-\eta)],$$

where $\{\lambda_j\}_{j=1}^{D}$ is a length-$D$ equispaced grid on $[0, 1/\eta]$. In the form appearing in the middle, this strategy was suggested for nonparametric testing by Kaplan [14] a few years before Cover's seminal paper; it is also analogous to Wald's mixture SPRT [30] and Robbins' tests-of-power-one [17]. In words, the TSM with the universal portfolio bet is equivalent to a mixture of TSMs over a uniform distribution on bets. This is analogous to the gambler splitting up their initial wealth, placing an equal amount on each possible bet, and summing up their total winnings across the bets. That this technique is nearly as efficient as the Kelly-optimal strategy follows intuitively: the sum of terms growing exponentially is dominated by the terms with the largest growth rates, and the Kelly-optimal strategy, which maximizes that rate, is one of those terms.

We do not compute the integral exactly but approximate it by a Riemann sum with $D$ terms; $D$ controls the accuracy of the approximation and the computational burden. This is similar to the various discrete mixture strategies proposed in Waudby-Smith et al. [33], Spertus [18], and Waudby-Smith and Ramdas [31], and simpler than refining the grid or adaptively re-balancing the portfolio as $t$ grows [9,31]. A fixed, relatively small value of $D$ (say, $D = 100$) is adequate for RLAs, where the sample sizes are typically modest. In other problems, a rough discretization might have a large impact on performance, for instance, if we cared about maximizing wealth as $t \to \infty$, in which case the wealth could become quite peaked around some $\lambda$ that is not included in the grid. There is extensive research on approximating the universal portfolio more accurately [13], but for our purposes, the juice is not worth the squeeze.

## 4 Slicing: Stratification

ONEAudit obviates the need for stratified hybrid audits, which partition the population into two strata based on the voting technology, draw cards independently across strata, and measure the risk using a stratified test [16,19,20]. Union-of-intersections test sequences (UI-TSs) provide the sharpest known method for sequential stratified testing [19]. They partition the complementary null into a union of intersections, each of which is tested using a product across strata of betting TSMs constructed separately within individual strata. In theory, an optimal UI-TS is the product of Kelly-optimal TSMs within each stratum. In practice, finding that UI-TS is computationally prohibitive and the oracle Kelly bet is unknown. For any simple alternative, the Kelly-optimal UI-TS can be approximated by partitioning the null into convex regions. For each region, the Kelly bet is found for a "representative" intersection null (e.g., the

centroid of the region); that bet is used for all TSMs within the region. The TSMs are concave over the intersection nulls within each region, so only the vertices of the regions need to be checked in order to compute an approximately Kelly-optimal UI-TS.

Spertus et al. [19] show that a stratified Kelly-optimal UI-TS is more efficient than an unstratified Kelly-optimal betting TSM when relatively low variance strata have relatively different means. This mirrors the familiar fact that stratification is helpful in reducing mean-squared error when the between-stratum variance is high and the within-stratum variance is low [5]. We might suspect that stratifying on batch information—such as the reported assorter mean of the batch—could increase the efficiency of ONEAudit, but Sect. 2 shows that if the CVRs are error-free, every batch has the same mean; only the variance will differ across batches. We expect the reported tallies to be quite accurate unless something in the election has gone seriously wrong. Stratification is most likely to be advantageous for ONEAudit if it can increase the efficiency of a UI-TS over an unstratified TSM when the *variability* differs in different strata even when the stratum means are equal.

## 5 Simulations

We simulated RLAs with a 5% risk limit to evaluate whether stratification or new betting strategies improve the efficiency of ONEAudit. The simulations rescale the original population of overstatements $\{x_i\}_{i=1}^N$ and null mean by $2u/(2u-v)$ to map the population and null means to $[0, 1]$. Simulations were conducted in Python 3.13.2; code is available at https://github.com/spertus/UI-TS.

### 5.1 Stratification

Overstatement populations were constructed with plurality overstatement assorters derived from 10,000 cards with CVRs and 10,000 cards spread across 10 reporting batches, each containing 1,000 cards. Every card had a valid vote. (Sample sizes generally increase with more invalid votes (e.g., Stark [27]), but we do not expect the relative performance of methods to change.).

The global reported assorter mean was $\bar{A}^c \in \{0.505, 0.51, 0.55, 0.60\}$. The stratum-level means were parameterized by the "across gap" $\Delta_{\mathrm{a}} \in \{0.0, 0.5\}$, where $\Delta_{\mathrm{a}} = 0.5$ implies the mean in the batch stratum was $\bar{A}^c - 0.25$ and the mean in the CVR stratum was $\bar{A}^c + 0.25$. The batch-level means in the batch stratum were parameterized by the "within gap" $\Delta_{\mathrm{w}} \in \{0.0, 0.5\}$, where $\Delta_{\mathrm{w}} = 0.5$ implies the batch-level means ranged uniformly between $(\bar{A}^c - \Delta_{\mathrm{a}}/2 - \Delta_{\mathrm{w}}/2)$ and $(\bar{A}^c - \Delta_{\mathrm{a}}/2 + \Delta_{\mathrm{w}}/2)$. The reported tallies were exactly accurate so that $\bar{A}_b^m = \bar{A}_b^c$ for every batch $b$.

We compared RLA sample sizes of an unstratified betting TSM and a stratified betting UI-TS. The unstratified sample was drawn uniformly with replacement; the stratified sample was drawn by proportional sampling with replacement and interleaved round-robin, alternating between the two strata. The

betting TSM used the oracle Kelly-optimal bets, computed by solving Equation (2) by bisection. The UI-TS used the banding strategy described in Spertus et al. [19], partitioning the null into $G = 100$ bands, deriving the oracle Kelly-optimal bets for each stratum at the centroid of each band, evaluating the intersection nulls at the vertices of each band, and minimizing over all 200 vertices. Since the bets were fixed within each band, the resulting bets were only approximately Kelly-optimal for each intersection null in the union, but we expect the slack was negligible since the bands were fairly narrow.

Table 1 displays the expected sample sizes of the two approaches. The Kelly-optimal TSM with unstratified sampling dominates the Kelly-optimal UI-TS with stratified sampling. The geometric mean of expected sample sizes is about 20% lower without stratification.

**Table 1.** Estimated expected sample sizes to confirm that the assorter mean is greater than 1/2 at risk limit 5% for unstratified and stratified ONEAudit using Kelly-optimal betting strategies. There are 20,000 cards in all, 10,000 with CVRs and 10,000 in 100 batches of 1,000 cards, for which ONEAudit assorter values were used to generate populations of overstatements. $\bar{A}^c$ is the global reported assorter mean; $\Delta_a$ is the spread between the assorter mean for the cards with CVRs and the assorter mean of the batches of 1,000 cards; $\Delta_w$ is the maximum spread of assorter means across batches. For example, in the last row the overall assorter mean is 0.6, the mean for the CVRs is 0.85, the mean for the batches is 0.35, and the 10 batch means are spread evenly between 0.1 and 0.6. Sampling is with replacement and the sample sizes were capped at 20,000 cards, so these estimates are biased slightly down. TSM: test supermartingale; UI-TS: union-of-intersections test sequence. The last column is the ratio of the TSM sample size to the stratified UI-TS sample size, as a percentage. Stratification increases the expected sample size in all these numerical experiments

| $\bar{A}^c$ | $\Delta_a$ | $\Delta_w$ | Unstratified TSM | Stratified UI-TS | Ratio (%) |
|---|---|---|---|---|---|
| 0.505 | 0.0 | 0.0 | 18,290 | 20,000 | 91 |
| 0.505 | 0.0 | 0.5 | 17,428 | 20,000 | 87 |
| 0.505 | 0.5 | 0.0 | 15,917 | 20,000 | 80 |
| 0.505 | 0.5 | 0.5 | 14,268 | 20,000 | 71 |
| 0.525 | 0.0 | 0.0 | 1,145 | 1,515 | 76 |
| 0.525 | 0.0 | 0.5 | 1,064 | 1,386 | 77 |
| 0.525 | 0.5 | 0.0 | 837 | 1,131 | 74 |
| 0.525 | 0.5 | 0.5 | 757 | 1,130 | 67 |
| 0.550 | 0.0 | 0.0 | 327 | 407 | 80 |
| 0.550 | 0.0 | 0.5 | 293 | 407 | 72 |
| 0.550 | 0.5 | 0.0 | 228 | 286 | 80 |
| 0.550 | 0.5 | 0.5 | 216 | 273 | 79 |
| 0.600 | 0.0 | 0.0 | 85 | 88 | 97 |
| 0.600 | 0.0 | 0.5 | 71 | 92 | 77 |
| 0.600 | 0.5 | 0.0 | 59 | 70 | 84 |
| 0.600 | 0.5 | 0.5 | 57 | 68 | 84 |

### 5.2 Betting

We compared expected sample sizes for ONEAudit RLAs using different betting strategies. The populations were again parameterized by the contest-level reported assorter mean $\bar{A}^c$, and heterogeneity parameters $\Delta_{\mathrm{a}}$ and $\Delta_{\mathrm{W}}$ as described above. Their values were also the same. We either let $\bar{A}^m = \bar{A}^c$ or introduced errors into the reported batch tallies (which we expect will generally be more erroneous), setting $\bar{A}^m$ to be half-way between $\bar{A}^c$ and $\eta$, thus halving the true margin. The simulations involve 100,000 cards in batches, each of which contains 1,000 cards, and 100,000 cards with CVRs. Cards were drawn by simple random sampling (without replacement) and the TSMs were computed accordingly.

Following Waudby-Smith et al. [33], bets were computed as if the sample were drawn IID. The Kelly-optimal bets were found by solving Equation (2) over $[0, 1/\eta]$ using bisection. The truncated shrinkage bet involves shrinking the running sample mean toward $\bar{A}^c$, with the default value of the tuning parameter $c$ and $d = 20$. The COBRA bet was computed assuming no 1-vote overstatements and 0.1% rate of 2-vote overstatements; it was thus fixed just below the maximum bet of $\lambda = 1/\eta$ in most cases. AGRAPA was computed as in Equation (3), with $c = 0.99$. The universal portfolio wealth was computed using the approximation in Sect. 3.3, with the TSM mixed over $D = 100$ equispaced points on $[0, 1/\eta]$.

Results appear in Table 2. AP Kelly had the smallest sample sizes, nearly the same as the oracle Kelly bets, even when the reported tallies were wrong. Universal portfolio was the next best, followed closely by AGRAPA. COBRA was usually the worst bet, except when the mean was far enough above the null that the oracle Kelly bet was near $1/\eta$. The bet based on the truncated shrinkage estimate also had poor performance: it is typically too conservative since it assumes, incorrectly, that the variance is maximal, as it would be for a Bernoulli population distribution. The geometric mean of the workload ratios comparing AP Kelly to other bets suggests AP Kelly could on average produce workloads that are 91% smaller than the COBRA bet, 74% smaller than the bet based on the truncated shrinkage estimate, 31% smaller than AGRAPA, and 25% smaller than the universal portfolio bet. Similarly, the universal portfolio workload was about 88% smaller than the COBRA bet, 66% smaller than the truncated shrinkage estimate bet, and 9% smaller than AGRAPA, on average.

## 6 Case Study: San Francisco 2024 Mayoral Race

We consider the 2024 mayoral race in San Francisco as a case study. This instant-runoff voting (IRV) contest included thirteen candidates. Daniel Lurie, who received 26% of the first-choice selections and 55% after all but two candidates were eliminated, defeated incumbent London Breed, who received 24% of the first-choice elections and 45% of the final round votes. All computations for this IRV election were carried out using the SHANGRLA codebase, to which we contributed new functions for testing by betting. The Jupyter Notebook we used to demonstrate the IRV ONEAudit and estimate workloads under different bets is

**Table 2.** Estimated expected (90th percentile) sample sizes to confirm that the assorter mean is greater than 1/2 at risk limit 5% for TSMs with various betting strategies (right columns) for sampling without replacement from ONEAudit overstatement populations with various reported assorter means ($\bar{A}^c$), true assorter means ($\bar{A}^m$) spread between the CVR and batch strata ($\Delta_{\mathrm{a}}$), and spread between batches within the batch stratum ($\Delta_{\mathrm{w}}$). Estimates were made from the sample mean and 0.9 empirical quantile of 1,000 sample sizes simulated by drawing simple random samples without replacement. Each population consists of 200,000 total cards; 100,000 cards have CVRs and 100,000 cards are in batches, each of size 1,000, modeling precinct-level totals. Oracle Kelly: optimal bet for sampling with replacement from the actual population of overstatements. AP Kelly: optimal bet based on sampling with replacement from an assumed (*a priori*) population of overstatements. Universal Portfolio: discrete approximation to the universal portfolio of Cover [7] with D = 100 points; see Waudby-Smith and Ramdas [31] and Waudby-Smith et al. [32]. AGRAPA: approximate growth-rate adapted to the particular alternative bet [31]. Truncated shrinkage: bet set implicitly by a truncated shrinkage estimate of the population mean [24] using the default parameters in SHANGRLA https://github.com/pbstark/SHANGRLA. COBRA: comparison-optimal bet [18] for sampling with replacement, computed on the assumption that there are no 1-vote overstatements and 2-vote overstatements occur for 0.1% of cards. The columns are sorted in rough order of increasing average sample size

| $\bar{A}^c$ | $\bar{A}^m$ | $\Delta_{\mathrm{a}}$ | $\Delta_{\mathrm{w}}$ | Oracle Kelly | AP Kelly | Universal Portfolio | AGRAPA | truncated shrinkage | COBRA |
|---|---|---|---|---|---|---|---|---|---|
| 0.505 | 0.502 | 0.0 | 0.0 | 73,202 (115,922) | 71,493 (113,043) | 103,786 (149,880) | 106,304 (154,946) | 158,008 (189,423) | 191,042 (199,476) |
| 0.505 | 0.502 | 0.0 | 0.5 | 67,277 (107,638) | 67,760 (109,147) | 93,319 (143,173) | 98,171 (147,857) | 160,152 (189,269) | 189,330 (199,511) |
| 0.505 | 0.502 | 0.5 | 0.0 | 58,016 (96,992) | 59,136 (96,467) | 85,558 (131,639) | 82,390 (131,582) | 157,225 (186,797) | 187,403 (199,044) |
| 0.505 | 0.502 | 0.5 | 0.5 | 55,133 (94,712) | 52,792 (87,743) | 75,683 (123,296) | 78,644 (125,641) | 155,773 (185,975) | 189,746 (199,002) |
| 0.505 | 0.505 | 0.0 | 0.0 | 25,941 (49,347) | 25,211 (48,603) | 38,689 (68,893) | 40,921 (71,757) | 76,194 (115,980) | 185,968 (196,804) |
| 0.505 | 0.505 | 0.0 | 0.5 | 24,247 (45,459) | 23,496 (43,965) | 37,553 (67,685) | 38,654 (70,340) | 76,887 (114,366) | 186,269 (199,014) |
| 0.505 | 0.505 | 0.5 | 0.0 | 18,69 (34,957) | 20,118 (37,337) | 30,793 (57,850) | 28,742 (53,071) | 74,181 (107,468) | 182,898 (194,580) |
| 0.505 | 0.505 | 0.5 | 0.5 | 17,292 (32,493) | 18,461 (34,509) | 28,033 (52,630) | 26,052 (47,518) | 73,300 (104,434) | 181,746 (194,394) |
| 0.510 | 0.505 | 0.0 | 0.0 | 24,118 (46,033) | 25,225 (46,615) | 39,721 (71,015) | 42,119 (74,312) | 78,111 (117,215) | 186,123 (196,758) |
| 0.510 | 0.505 | 0.0 | 0.5 | 23,088 (43,063) | 22,713 (43,211) | 37,572 (67,753) | 37,964 (69,282) | 76,050 (114,675) | 188,060 (199,015) |
| 0.510 | 0.505 | 0.5 | 0.0 | 18,743 (34,358) | 18,943 (36,126) | 29,915 (57,553) | 28,649 (54,976) | 76,199 (109,565) | 182,339 (194,401) |
| 0.510 | 0.505 | 0.5 | 0.5 | 17,542 (34,520) | 16,851 (31,960) | 27,151 (51,861) | 26,973 (50,630) | 74,457 (106,746) | 183,134 (194,178) |
| 0.510 | 0.510 | 0.0 | 0.0 | 7,080 (13,844) | 7,156 (14,092) | 11,157 (22,204) | 12,161 (22,733) | 24,448 (38,449) | 177,439 (188,711) |
| 0.510 | 0.510 | 0.0 | 0.5 | 6,479 (12,989) | 6,521 (12,677) | 9,969 (19,450) | 11,302 (21,740) | 23,474 (37,470) | 182,562 (198,021) |
| 0.510 | 0.510 | 0.5 | 0.5 | 4,889 (9,818) | 4,547 (9,074) | 6,873 (13,881) | 7,232 (14,328) | 23,127 (35,595) | 160,778 (180,462) |
| 0.510 | 0.510 | 0.5 | 0.0 | 5,659 (11,478) | 5,638 (10,993) | 8,147 (16,340) | 7,985 (15,510) | 23,655 (36,571) | 162,751 (181,016) |
| 0.550 | 0.525 | 0.0 | 0.0 | 1,178 (2,288) | 1,245 (2,512) | 1,533 (3,293) | 1,845 (3,642) | 4,480 (7,347) | 130,302 (158,069) |
| 0.550 | 0.525 | 0.0 | 0.5 | 1,116 (2,252) | 1,103 (2,164) | 1,349 (2,970) | 1,637 (3,370) | 4,277 (6,826) | 164,400 (195,030) |
| 0.550 | 0.525 | 0.5 | 0.0 | 843 (1,739) | 866 (1,777) | 1,019 (2,183) | 1,120 (2,249) | 4,228 (6,530) | 73,347 (116,466) |
| 0.550 | 0.525 | 0.5 | 0.5 | 769 (1,502) | 763 (1,526) | 867 (1,829) | 1,083 (2,140) | 4,109 (6,117) | 66,991 (112,060) |
| 0.550 | 0.550 | 0.0 | 0.0 | 309 (616) | 313 (643) | 373 (781) | 436 (880) | 1,139 (1,774) | 110,469 (190,030) |
| 0.550 | 0.550 | 0.0 | 0.5 | 288 (570) | 294 (585) | 321 (678) | 414 (847) | 1,080 (1,692) | 67,345 (189,989) |
| 0.550 | 0.550 | 0.5 | 0.0 | 233 (467) | 235 (483) | 259 (541) | 304 (608) | 1,099 (1,640) | 436 (1,163) |
| 0.550 | 0.550 | 0.5 | 0.5 | 206 (417) | 211 (413) | 248 (511) | 279 (563) | 1,078 (1,580) | 415 (978) |
| 0.600 | 0.550 | 0.0 | 0.0 | 310 (618) | 315 (622) | 354 (755) | 431 (903) | 1,223 (1,902) | 52,712 (98,107) |
| 0.600 | 0.550 | 0.0 | 0.5 | 289 (576) | 269 (530) | 310 (629) | 400 (804) | 1,186 (1,790) | 43,011 (89,454) |
| 0.600 | 0.550 | 0.5 | 0.0 | 189 (367) | 190 (368) | 210 (408) | 270 (528) | 1,165 (1,698) | 234 (545) |
| 0.600 | 0.550 | 0.5 | 0.5 | 179 (355) | 181 (344) | 203 (394) | 232 (471) | 1,136 (1,627) | 233 (510) |
| 0.600 | 0.600 | 0.0 | 0.0 | 80 (155) | 82 (158) | 90 (171) | 104 (214) | 300 (447) | 130 (298) |
| 0.600 | 0.600 | 0.0 | 0.5 | 78 (154) | 79 (158) | 85 (153) | 98 (213) | 300 (443) | 124 (294) |
| 0.600 | 0.600 | 0.5 | 0.0 | 59 (114) | 61 (121) | 77 (129) | 82 (156) | 298 (440) | 61 (123) |
| 0.600 | 0.600 | 0.5 | 0.5 | 57 (110) | 58 (113) | 73 (129) | 75 (142) | 295 (420) | 60 (122) |

available at https://github.com/spertus/UI-TS/blob/main/Code/SF_oneaudit_example.ipynb.

The election produced 1,603,908 CVRs, of which 216,286 were for cards cast in 4,223 precinct batches and 1,387,622 CVRs were for vote-by-mail (VBM) cards. VBM CVRs are linked to the corresponding card, facilitating card-level comparison auditing, but the in-person CVRs are not linked to individual cards, only to tabulation batches. The CVRs were incorporated into the audit using ONEAudit. RAIRE [4] was used to generate the assertions for the audit to test. The 80th percentile of the distribution of sample sizes for each of the five bets introduced above was estimated from the 0.8 empirical quantile of sample sizes required to confirm the winner in 100 simulated audits with no error in the reported tallies. Results appear in Table 3. Kelly optimal and AGRAPA reduced the sample size by about half compared to the truncated shrinkage estimate bet or the COBRA bet.

**Table 3.** Estimated 80th percentile sample sizes of ONEAudit RLAs at a 5% risk limit using betting TSMs with various betting strategies computed under sampling without replacement from the ONEAudit overstatement population corresponding to the 2024 San Francisco Mayoral race, which used instant-runoff voting (IRV). Estimates are the 0.8 empirical quantile of 100 sample sizes simulated by drawing simple random samples of cards without replacement and treating the CVR for each card as if it were an accurate manual-vote record (MVR). For cards cast in precincts, the assorter applied to the MVR is compared to the ONEAudit average assorter value for the batch. Kelly-optimal: optimal for sampling with replacement from the overstatements implied by the batch-level results, on the assumption that the results are accurate. AGRAPA: approximate growth-rate adapted to the particular alternative [31]. Universal Portfolio: discrete approximation to Cover [7] with D=100 points; see Waudby-Smith and Ramdas [31] and Waudby-Smith et al. [32]. Truncated shrinkage: bet set implicitly by a truncated shrinkage estimate of the population mean [24] using the default parameters in SHANGRLA https://github.com/pbstark/SHANGRLA. COBRA: comparison-optimal bet [18] on the assumption that there are no 1-vote overstatements and a 0.1% rate of 2-vote overstatements

| Kelly-optimal | AGRAPA | Universal Portfolio | truncated shrinkage | COBRA |
|---|---|---|---|---|
| 111 | 111 | 165 | 192 | 211 |

## 7 Discussion

ONEAudit is simpler and more efficient without stratification. ONEAudit can be made substantially more efficient by employing approximately Kelly-optimal or universal portfolio strategies for betting. The universal portfolio can be quickly and sufficiently approximated by a discrete mixture of betting TSMs over a grid of bets, which has already proven useful for card-polling and card-level comparison audits. Simulations and data from a 2024 IRV contest in San Francisco, CA, strongly suggest that unstratified ONEAudit with *a priori* Kelly

optimal betting or discrete portfolio betting are simple and efficient methods to conduct RLAs of elections that use diverse voting technology.

There remain gaps to be explored. We did not determine how much tally error would be necessary to make *a priori* Kelly betting worse than the universal portfolio or some other strategy that does not rely on the reported results. We also only simulated elections with tally errors in batches, not in CVRs. We expect errors in the batches to have less effect on efficiency, but also to be more common in real elections. In practice, it could be desirable to ensure both efficiency when the reported tallies are accurate and robustness to possible tally errors by mixing an *a priori* Kelly TSM and a universal portfolio TSM. The mixture could start with high weight (e.g., 0.9) on the *a priori* Kelly TSM and, as the sample size increases, shift weight to the universal portfolio, which asymptotically attains nearly the same growth rate as the oracle Kelly TSM. Alternatively, the universal portfolio could use a weight function in the quadrature that puts more weight on the value of $\lambda$ corresponding to the *a priori* Kelly bet instead of uniform weights. These ideas could easily be implemented in the SHANGRLA codebase.

We only examined bets optimized for sampling with replacement, and therefore did not update as the conditional null mean changed. This makes it easier to compute the Kelly TSM—it only needs to be computed once—but the bets could be adjusted to account for sampling without replacement. We expect the effect of this change to be minor: the marginal improvement only becomes noticeable once the sample includes a substantial portion of the population, at which point a full hand count is generally more efficient than random sampling.

It is possible that stratification would increase efficiency if the strata partitioned batches by error rate, a case we did not examine. We speculate that it would take quite high error rates and that the strata would need to discriminate accurately between low and high error rates for stratification to make much difference. If auditors believed they had such information, a different kind of inspection (e.g., a forensic audit) might be more appropriate, since high error rates should occur only if there were gross procedural errors, misconfiguration, or malfunction. As we see it, with ONEAudit and universal-portfolio betting (possibly augmented by shrinking towards the *a priori* Kelly bet), the only reason for stratification is to accommodate legal requirements—not to reduce the overall workload or to accommodate diverse election equipment.

## References

1. Appel, A., DeMillo, R., Stark, P.: Ballot-marking devices cannot assure the will of the voters. Politics, and Policy, Election Law Journal, Rules (2020)
2. Appel, A., Stark, P.: Evidence-based elections: create a meaningful paper trail, then audit. Georgetown Law Technol. Rev. **4**(2), 523–541 (2020)
3. Blom, M., Stark, P.B., Stuckey, P.J., Teague, V., Vukcevic, D.: Auditing Hamiltonian elections. In: Bernhard, M., et al. (eds.) FC 2021. LNCS, vol. 12676, pp. 235–250. Springer, Heidelberg (2021)
4. Blom, M., Stuckey, P., Teague, V.: Ballot-polling risk limiting audits for IRV elections. In: Krimmer, R., et al. (eds.) Electronic Voting, pp. 17–34. Springer, Cham (2018)

5. Cochran, W.: Sampling Techniques, 3rd edn. John Wiley & Sons Inc, New York (1977)
6. Cover, T.M., Ordentlich, E.: Universal portfolios with side information. IEEE Trans. Inf. Theory **42**(2), 348–363 (1996)
7. Cover, T.: Universal portfolios. Math. Financ. **1**, 1–29 (1991)
8. Ek, A., Stark, P., Stuckey, P., Vukcevic, D.: Adaptively weighted audits of instant-runoff voting elections: AWAIRE. In: Volkamer, M., et al. (eds.) Electronic Voting, pp. 35–51. Springer Nature, Cham (2023)
9. Ek, A., Stark, P., Stuckey, P., Vukcevic, D.: Efficient Weighting Schemes for Auditing Instant-Runoff Voting Elections, pp. 18–32. Springer Nature (2024)
10. Fuller, B., Harrison, A., Russell, A.: Adaptive risk-limiting comparison audits. In: 2023 IEEE Symposium on Security and Privacy (SP), pp. 3314–3331 (2023)
11. Glazer, A., Spertus, J., Stark, P.: More style, less work: card-style data decrease risk-limiting audit sample sizes. Digital Thre. Res. Pract. (DTRAP) **2**, 1–15 (2021)
12. Glazer, A., Spertus, J., Stark, P.: Stylish risk-limiting audits in practice. In: Proceedings of E-Vote-ID (2023). Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn (To appear)
13. Kalai, A., Vempala, S.: Efficient algorithms for universal portfolios. J. Mach. Learn. Res. **3**(Nov), 423–440 (2002)
14. Kaplan, H.: A method of one-sided nonparametric inference for the mean of a nonnegative population. Am. Stat. **41**, 157–158 (1987)
15. Lindeman, M., Stark, P., Yates, V.: BRAVO: ballot-polling risk-limiting audits to verify outcomes. In: Proceedings of the 2011 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections (EVT/WOTE '11). USENIX (2012)
16. Ottoboni, K., Stark, P., Lindeman, M., McBurnett, N.: Risk-limiting audits by stratified union-intersection tests of elections (SUITE). In: Electronic Voting. E-Vote-ID 2018. Lecture Notes in Computer Science. Springer (2018)
17. Robbins, H., Siegmund, D.: The expected sample size of some tests of power one. Ann. Stat. **2**(3), 415–436 (1974)
18. Spertus, J.: COBRA: comparison-optimal betting for risk-limiting audits. In: International Conference on Financial Cryptography and Data Security, pp. 95–109. Springer, Cham (2023)
19. Spertus, J., Sridhar, M., Stark, P.: Sequential stratified inference for the mean. arXiv preprint arXiv:2409.06680 (2024)
20. Spertus, J., Stark, P.: Sweeter than SUITE: supermartingale stratified union-intersection tests of elections. In: Krimmer, R., et al. (eds.) Electronic Voting, pp. 106–121. Springer, Cham (2022)
21. Stark, P.: Risk-limiting post-election audits: *P*-values from common probability inequalities. IEEE Trans. Inf. Foren. Secur. **4**, 1005–1014 (2009)
22. Stark, P.: Super-simple simultaneous single-ballot risk-limiting audits. In: Proceedings of the 2010 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections (EVT/WOTE '10). USENIX (2010)
23. Stark, P.: Sets of half-average nulls generate risk-limiting audits: SHANGRLA. Finan. Crypt. Data Secur. Lect. Notes Comput. Sci. **12063** (2020)
24. Stark, P.: ALPHA: audit that learns from previously hand-audited ballots. Ann. Appl. Stat. **17**(1), 641–679 (2023)
25. Stark, P., Wagner, D.: Evidence-based elections. IEEE Secur. Priv. **10**, 33–41 (2012)
26. Stark, P.: Non(c)esuch ballot-level comparison risk-limiting audits. In: K., Sokratis et al. (ed.) Computer Security. ESORICS 2022 International Workshops, pp. 541–554. Springer International Publishing, Cham (2023)

27. Stark, P.: Overstatement-net-equivalent risk-limiting audit: ONEAudit. In: International Conference on Financial Cryptography and Data Security, pp. 63–78. Springer (2023)
28. Ville, J.: Étude critique de la notion de collectif. Gauthier-Villars, Paris (1939)
29. Wald, A.: Sequential tests of statistical hypotheses. Ann. Math. Stat. **16**, 117–186 (1945)
30. Wald, A.: Sequential Analysis. Dover Publications, New York (1947)
31. Waudby-Smith, I., Ramdas, A.: Estimating means of bounded random variables by betting. J. Royal Statist. Soc. Ser. B Statist. Methodol. **86** (2024)
32. Waudby-Smith, I., Sandoval, R., Jordan, M.: Universal log-optimality for general classes of e-processes and sequential hypothesis tests. arXiv:2504.02818 (2025)
33. Waudby-Smith, I., Stark, P.B., Ramdas, A.: RiLACS: risk limiting audits via confidence sequences. In: Krimmer, R., Volkamer, M., Duenas-Cid, D., Kulyk, O., Rønne, P., Solvak, M., Germann, M. (eds.) E-Vote-ID 2021. LNCS, vol. 12900, pp. 124–139. Springer, Cham (2021)



# Re-voting Under Surveillance: National eID Transaction Logs as a Threat to Coercion Resistance in Estonian Internet Voting

Tarvo Treier(✉)

Department of Software Science, Tallinn University of Technology, 12618 Tallinn, Estonia
tarvo.treier@taltech.ee

**Abstract.** Estonia's nationwide Internet-voting scheme relies on the state-mandated electronic identity (eID) infrastructure for strong voter authentication and qualified electronic signatures. A statutory safeguard against coercion is re-voting: a voter may cast multiple electronic ballots during the advance-voting period, with only the last one counted, and the number of ballots must remain secret. This expectation is violated by current eID audit practice: every signing transaction-including each vote-is irrevocably logged by the eID service provider and displayed to the credential holder. These logs reveal the exact count and timing of a voter's interactions with the voting system, compromising the intended secrecy of re-voting and enabling coercers to detect whether the voter has changed their choice. This paper presents a threat model and examines concrete design alternatives-such as persistent log filtering, dedicated voting credentials, and offline signing-analysing their respective trade-offs in security, usability, regulatory compliance, and system complexity. The findings demonstrate how well-intentioned components can interact to break coercion resistance through a metadata-based side-channel. The identified vulnerability has been responsibly disclosed to the relevant Estonian authorities. The case underlines the need for composition-aware risk assessment whenever election systems depend on external digital infrastructures.

**Keywords:** Internet voting · coercion resistance · side-channel · electronic identity · Estonia

## 1 Introduction

Internet voting (i-voting) allows citizens to cast their vote outside a traditional polling station, from any location with internet access. This increases convenience and may raise turnout, but it also introduces new challenges for election security, integrity, secrecy, and voter freedom. A particularly critical issue is the


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 191–207, 2026.
https://doi.org/10.1007/978-3-032-05036-6_12

integration of national digital infrastructures-especially electronic identity (eID) systems-into the electoral process. While eID systems improve authentication and usability, their use in voting may introduce unforeseen security risks and goal conflicts.

### 1.1 Background and Motivation

Estonia is widely recognised as a pioneer of digital democracy, having introduced nationwide i-voting already in 2005 [3]. Its success is underpinned by a mandatory eID infrastructure [14]. The same infrastructure used by citizens for daily digital signatures and secure transactions has also enabled the move to online elections. However, this infrastructural trust may become a vulnerability if side effects compromise other critical goals.

One of the core features of Estonia's i-voting system is re-voting, which allows voters to cast multiple electronic ballots during the advance-voting period, with only the last one being counted. This mechanism is intended to enhance coercion resistance [10], allowing a voter to privately override a coerced vote [24]. Estonian law ($§48^6(9)$ of the Riigikogu Election Act) requires that the system must not allow a voter to prove to anyone which of their electronic votes was counted [20]. This implies that it must be impossible to detect how many votes were cast or when. The Estonian Supreme Court has affirmed the central role of re-voting in safeguarding electoral freedom, calling it the only sufficiently effective measure for ensuring voter autonomy under electronic conditions [25].

Re-voting, therefore, must be more than a theoretical safeguard-it must work in practice. Its success depends in part on the secrecy of voting frequency and timing. A 2024 risk analysis by the Cybersecurity Committee of the Estonian Academy of Sciences identified 31 distinct risks in i-voting [4], none of which covered re-vote detection via eID logs.[1]

A public demonstration in March 2023 showed that combining an e-vote cryptogram, the eID transaction log and an official confirmation from the Electoral Service of whether the electronic vote was overridden can prove how an individual voted [19]. The Electoral Service responded that this proof is no stronger than photographing a paper ballot [12]. A lightning talk at E-Vote-ID 2023 reiterated the same risk and pointed to the eID transaction log as evidence [18]. None of these sources, however, analysed the log itself as an independent side-channel, mapped its legal implications, or compared mitigation options. The present paper fills that gap by presenting a threat model, demonstrating in practice how the eID log leaks re-voting, and comparing six mitigation families side-by-side.

[1] The author notified a member of the Cybersecurity Committee of the Estonian Academy of Sciences about the missing log-trace risk on 2 Oct 2024. At a meeting with the National Electoral Service on 4 Jun 2025, several committee members confirmed that mitigation work was under way. The accepted manuscript was forwarded to the Electoral Service on 24 Jun 2025.

### 1.2 Problem Statement

Classic side-channel attacks exploit physical characteristics of hardware-such as power use, timing patterns, or cache behaviour [11,26]. However, modern information systems may leak sensitive data through metadata produced for legitimate purposes. This paper refers to such leakage as a *metadata-based side-channel*.

In the Estonian case, the risk arises not from cryptographic flaws, but from the interplay of otherwise secure systems. Every qualified digital signature is logged by the eID service, including each e-vote. A typical log entry may read `Signing, 2025-03-01 14:02:17, Voting, OK`. These entries are visible to the credential holder and persist under long-term audit rules. Consequently, both voters and coercers may observe the number and timing of votes cast. Although no vote content is revealed, this metadata alone compromises re-voting secrecy and undermines receipt-freeness. Receipt-freeness means that an attacker cannot determine a voter's exact behaviour, which prevents them from coercing the voter by dictating a specific choice [10].

This represents a composition attack [8], where the combination of two secure components-the national eID transaction log and the re-voting protocol-produces an unintended privacy vulnerability. The metadata in eID transaction logs functions as a side-channel, revealing whether a voter has changed their vote.

While logs from the voting system itself are anonymised or destroyed according to electoral law, eID transaction logs are retained for at least ten years in line with trust service regulation. Thus, re-voting remains traceable long after the election concludes.

However, the presence of verifiable metadata alone presents a serious and independent threat to the effectiveness of re-voting. While a coercer can sometimes persuade a voter to photograph a paper ballot, the eID transaction log constitutes a markedly different form of evidence. It is produced automatically by a qualified trust-service provider, written once and retained for at least ten years, creating a persistent and highly credible record of the voter's actions. That official provenance gives coercers a systemic, low-cost channel for checking whether a re-vote occurred-a threat that scales far more easily than methods requiring physical presence or elaborate coordination.

Moreover, because the entries remain available for years, their mere perceived auditability may deter voters from casting a secret re-vote even if the coercer never consults them. As soon as the coercer has supervised the voter once-during an i-voting session-demanding a follow-up look at the transaction log after the election adds virtually no burden; the voter is already accustomed to the coercer's scrutiny.

### 1.3 Contribution

This study is based on the premise that re-voting is Estonia's primary coercion-resistance mechanism, and that its effectiveness hinges on concealing the number

and timing of voting actions. It is shown that transaction logs from the national eID system create a metadata-based side-channel that compromises this secrecy.

The contributions of this work are as follows:

- **Threat Model:** A model characterises log leakage and states the exact condition under which re-voting secrecy is violated.
- **Impact Assessment:** The effect of this side-channel on coercion resistance is evaluated from both technical and legal perspectives.
- **Mitigation Strategies:** Several design alternatives are compared-such as log filtering, separate voting credentials, and offline signing-and their trade-offs analysed.
- **Risk Taxonomy Extension:** An extension to the 2024 national risk register is proposed, adding a category for metadata-based side-channels and outlining recommendations for mitigation.

The Estonian case illustrates a broader lesson: coercion resistance can fail when component interactions are not subject to composition-aware risk assessment. This paper provides a foundation for such analysis in i-voting systems.

## 2 Background: Re-voting and eID Transaction Logs

Estonia's i-voting scheme depends on two components built for different purposes: the re-voting protocol and the nationwide eID ecosystem operated by qualified trust-service providers. Each component is effective for its original goal, yet their interaction creates the metadata-based side-channel analysed in Sect. 3. This section explains the two pillars and relates the Estonian setting to experiences in Norway and Switzerland.

### 2.1 Re-voting as a Safeguard Against Coercion

Estonian i-voting allows a voter to submit multiple electronic ballots during the advance-voting period; only the last ballot is counted [24]. The mechanism serves two purposes:

1. **Correction:** a coerced or erroneous vote can be privately overridden later.
2. **Deterrence:** a coercer can never be sure the supervised vote will remain final.

Section $48^6(9)$ of the *Riigikogu Election Act* bars a voter from proving which electronic ballot was counted [20]. The Estonian Supreme Court calls re-voting practically the only sufficiently effective safeguard of voter freedom in an uncontrolled environment (3-4-1-13-05, para 30) [25]. Foundational work on election security defines a hierarchy of goals: *receipt-freeness*-the voter cannot prove how they voted-and the stronger *coercion resistance*-an adversary cannot tell whether the voter obeyed a demand [10]. In the Estonian system, re-voting is presented as the primary means of approximating these goals. The safeguard works only while the number and timing of votes remain secret.

Prior research already questions whether re-voting is compatible with other goals. Pereira shows that a malicious client can silently replace the voter's final encrypted ballot, leaving the voter unsure which ballot is stored [16]. This paper exposes a complementary weakness-a coercer can detect re-voting via metadata in the eID transaction log-and compares concrete defences.

When nationwide i-voting started in 2005, voters had no convenient public view of signature logs, so re-voting was deemed effective even outside polling places. Over time, user-facing portals such as *MyID*[2] were launched, letting citizens browse every qualified signature they produced. This transparency created a metadata side-channel: a coercer who sees the log can confirm whether and when the voter re-voted. Section 3 analyses this channel.

### 2.2 The National eID Ecosystem and Its Transaction Logs

The Estonian eID ecosystem-ID Card, Mobile-ID and Smart-ID-is widely used in banking and public e-services [14]. Each qualified signature or authentication produces a transaction-log record that voters can view in *MyID*. Crucially, these logs are maintained by the trust-service provider and cannot be altered or deleted by the credential holder-the provider may only append new records-thereby preserving their integrity for audit purposes. The logs serve multiple goals:

- **User Transparency:** Citizens can detect misuse of their identity.
- **Non-repudiation Evidence:** European Telecommunications Standards Institute (ETSI) EN 319 401 §7.10 mandates log retention [6].
- **Incident Response:** ETSI EN 319 401 §7.9.1 requires anomaly detection [6].
- **Regulatory Compliance:** Article 24 of Electronic Identification and Trust Services (eIDAS) Regulation obliges trust-service providers to implement security measures and report incidents [5].
- **Ten-Year Retention:** The service provider's privacy policy states that trust-service logs are kept for at least ten years to prove service correctness [22].

A simplified entry for an i-voting action may read:

```
Signing   2025-03-01 14:02:17   Voting   OK
```

The log reveals no ballot content, but does disclose:

- the exact number of vote-related signatures, and
- timestamps with one-second precision for each signature.

Because the audit trail is visible to the voter-and to anyone who can compel its disclosure-the metadata acts as a de facto receipt showing whether and when a re-vote occurred. Election-system logs, by contrast, must be anonymised or destroyed, creating a systemic inconsistency that enables the side-channel analysed next.

[2] SK ID Solutions AS, https://myid.skidsolutions.eu/en/.

### 2.3 International Parallels

Other countries have found that transparency artefacts can be repurposed as receipts.

- **Norway (Return Codes):** During the 2011 and 2013 i-voting pilots, each electronically cast ballot triggered an ordinary SMS whose return code matched the personal code sheet mailed to the voter in advance. As in Estonia, a voter could cast a new electronic ballot to override a coerced one, and every re-vote generated a fresh SMS. Although the message thread could in principle be deleted, coercion resistance still depended on the voter remembering to remove the codes before any potential coercer inspected the phone. Researchers warned that such codes "simplify the task of coercers and vote buyers" [1]. The government discontinued the pilot in 2014, citing security concerns and waning political support [15].
- **Switzerland (sVote Receipt):** The sVote platform returned a digitally signed package containing the encrypted ballot and a return code that voters could compare with a pre-printed paper sheet. A subsequent technical audit concluded that the same package allows the client to prove that a particular vote was cast, because the server signature binds the code to the encrypted ballot and is publicly verifiable [17]. Further cryptographic flaws were shown to let manipulated votes pass verification, and-because of those findings-no version of sVote was used in either the May 2019 Swiss referendum or the federal elections of October 2019 [9].

Sewell and Vitek note that assembling a system from partially trustworthy components is safe only if their interactions are encapsulated by a wrapper mechanism [21]. The Norwegian and Swiss cases illustrate that un-encapsulated interactions can create unexpected side-channels. In Estonia, no such wrapper exists between eID transaction logs and re-voting, so neutral metadata can function as a receipt and undermine coercion resistance. The next section analyses this side-channel in depth.

## 3 Metadata-Based Side-Channel Analysis

This section shows how the re-voting protocol $R$ and the eID transaction log $L$-each designed for a different purpose-interact to create a metadata side-channel that undermines coercion resistance. First, a concise threat model is presented. Second, the impact of composition on security properties is examined. Finally, residual risk is assessed even in the presence of the paper-ballot fallback.

### 3.1 Threat Model and Attack Scenario

The leakage threat is presented precisely: the model specifies exactly what metadata an adversary can observe and how that observation leads to a coercion decision, without introducing additional cryptographic assumptions.

*Participants.* A single voter $V$ uses the re-voting protocol $R$. A coercer $C$ can supervise one voting session and later compel $V$ to display the voter-accessible *MyID* view of the log $L_V$. The qualified trust-service provider (TSP) appends log records and offers voters no means to alter or delete existing ones; any modification would require collusion with, or compromise of, the provider itself.

*Log Example.* Each e-vote appends an immutable entry such as

$\langle type = \mathsf{Signing}, time = \text{2025-03-01 14:02:17}, service = \mathsf{Voting}, result = \mathsf{OK} \rangle$.

**Definition 1 (Log-leakage).** *Let $L_V^{\mathsf{vote}} \subseteq L_V$ be the subsequence of vote-related entries. The leakage function*

$$\mathcal{F}(L_V^{\mathsf{vote}}) = \langle n, t_1, \ldots, t_n \rangle, \qquad t_1 < \cdots < t_n$$

*returns the number $n$ of ballots cast by $V$ and the corresponding one-second-precision timestamps $t_1, \ldots, t_n$.*

*Attack Steps.*

- **S1** At the start of the supervised session $C$ instructs $V$ to open *MyID*. $V$ displays the current log view, revealing $k = |\mathcal{F}(L_V^{\mathsf{vote}})|$, the number of vote entries recorded so far.[3]
- **S2** A ballot is then cast under coercer supervision at time $t_{k+1}$. $C$ orders $V$ not to vote again.
- **S3** After the voting period closes $C$ compels $V$ to reopen *MyID*. $C$ observes $\mathcal{F}(L_V^{\mathsf{vote}}) = \langle n, t'_1, \ldots, t'_n \rangle$.
- **S4** A re-vote is deemed to have occurred *iff* $n > k{+}1$. Otherwise the supervised ballot is accepted as potentially counted.

Counting the entries alone suffices; timestamps merely indicate when any unsupervised vote was cast. Because the log is immutable and voter-accessible, the two cases can be distinguished with certainty-no cryptanalysis or privileged system access is required. Merely knowing that such verification is possible may deter some voters from exercising their legal right to re-vote.

### 3.2 Breakdown of Security Properties Under Composition

**Voter log** $L_V$: Append-only, immutable, and reveals $\mathcal{F}(L_V)$ but no ballot content.

**Re-vote** $R$: Counts only the last ballot and assumes that $\mathcal{F}(L_V)$ remains hidden.

**Composition**: Once combined, the metadata becomes a verifiable receipt; both coercion resistance and receipt-freeness collapse.

Thus an adversary needs only the voter-specific log view together with the pre-coercion count $k$ to decide whether a re-vote occurred; timestamps are optional corroborating evidence.

[3] A *minimal-assumption variant* needs only the supervised timestamp $t_{sup} = t_{k+1}$; re-voting is detected as soon as any later entry appears.

### 3.3 Paper-Ballot Override: Partial Remedy

Estonian law allows a voter to annul all prior electronic votes by casting a paper ballot in person on election day. This serves as a last-resort defence, yet its effectiveness against a determined coercer is mixed.

- **Extended Exposure:** Pressure can persist until the final day, because the coercer needs to influence only one physical act-the trip to a polling station.
- **Visibility and social pressure:** Travelling to a polling place is a public act. Informal social monitoring (e.g. by family or neighbours) can make a secret visit risky, without requiring continuous digital surveillance.
- **Post-election Record:** After polling day any voter may request an official certificate from the National Electoral Service stating whether the final counted ballot was electronic or paper.[4] A coercer could compel the voter to obtain and reveal this confirmation, providing a definitive test for a paper-ballot override. Although mass requests would be conspicuous, the mere possibility further reinforces the deterrent.

Paper ballots therefore mitigate, but do not eliminate, the incentive to coerce.

### 3.4 Residual Risk Assessment

Although no ballot content is revealed, the metadata alone serves as a credible receipt. This is especially damaging because coercion itself leaves no corresponding trace in the election system, undermining the deterrent effect of re-voting. From an adversary's viewpoint the cost is low: the pre-coercion count $k$ and a single post-election inspection of the voter-accessible log are sufficient. Coercion resistance in the Estonian scheme thus hinges on hiding voting frequency-an assumption invalidated by current e-ID audit practice, which exposes $\mathcal{F}(L_V^{\mathsf{vote}})$ to the credential holder. Breaking the data flow from the voter-accessible log to the coercer is therefore essential; Sect. 4 explores concrete counter-measures.

## 4 Mitigation Strategies for the Metadata-Based Side-Channel

This section surveys technical and procedural measures that could neutralise, dampen, or compensate for the side-channel identified in Sect. 3. Five classical counter-measure classes-Hiding, Isolation, Leakage Dampening, Masking, and Encryption-are adopted from the taxonomy of Autio et al. [23]. For each class, one concrete design is sketched, its strengths and weaknesses are outlined, and a qualitative rating (High/Medium/Low) is given along three dimensions: effectiveness, deployability, and regulatory fit.

[4] A publicly available video demonstration illustrates how such a request and encrypted reply are obtained [19].

### 4.1 Hiding: Persistent Log Filter

A persistent filter would hide i-voting entries from the default *MyID* web view while retaining them on the server for audit purposes.

**Advantages:** Vote-related metadata is no longer visible online but remains accessible in a physical service centre upon request; audit staff retain full data.
**Drawbacks:** Transparency decreases, as some voters may mistrust a portal that no longer shows a "complete" history. Inspecting hidden entries requires visiting a service desk in person, adding mild inconvenience.
**Effectiveness - High:** Online leakage is blocked for routine coercion scenarios.
**Deployability - High:** Only front-end and API-level access-control changes are required.
**Regulatory fit - High:** ETSI and eIDAS retention rules are satisfied because no entries are deleted.

The filter therefore acts as a wrapper that interrupts the information flow between the trust-service database and routine users while preserving official access paths.

### 4.2 Isolation (A): Separate Voting Credential

A dedicated certificate issued solely for elections would have its own PKI hierarchy and separate logs.

**Advantages:** Election signatures never appear in the general e-ID log, so coercers learn nothing from that source.
**Drawbacks:** Requires a large roll-out effort, new middleware, and extra credential management, and may require new hardware or separate certificates on some devices.
**Effectiveness - High:** Leakage is eliminated at the source.
**Deployability - Low:** New PKI, software, and registration flows are needed.
**Regulatory fit - High:** Election logs fall outside commercial trust-service regulation, easing ten-year retention constraints.

The dedicated credential itself acts as a wrapper that cleanly separates election traffic from everyday e-ID usage. A similar approach is used in the *Selections* voting system [2].

### 4.3 Isolation (B): Offline Signing with the ID Card

A voting client could generate the signature locally inside the ID card chip and transfer only the sealed ballot, bypassing the remote signing service.

**Advantages:** No vote-related entry is written to the trust-service log, eliminating the side-channel without extra credentials.
**Drawbacks:** Works only for ID cards because Mobile-ID and Smart-ID rely on remote server-side signing, not local private keys. Requires new middleware and a wide software rollout. Immediate signature validation would be replaced by delayed verification, increasing the risk of undetected invalid signatures during voting.
**Effectiveness - Medium:** Central log leakage is completely avoided only for ID cards.
**Deployability - Medium:** New software is needed, but no PKI overhaul is required.
**Regulatory fit - High:** ETSI logging rules apply only to qualified remote signatures, not to local card operations.

The local signing client operates as a wrapper between the ID card chip and the voting server, eliminating remote log entries.

### 4.4 Leakage Dampening: User-Controlled Log Hiding

The portal could allow voters to hide specific entries from their online view after login.

**Advantages:** Quick to implement; empowers privacy-conscious voters by giving them direct control.
**Drawbacks:** Protection is entirely timing-dependent. The voter must hide the entry immediately after re-voting; any delay allows a coercer to compel the voter to show the log and detect the additional record. *The hide operation is one-way: once an entry is hidden, it cannot be unhidden online.*
**Effectiveness - Medium:** A one-shot defence that succeeds only if the voter hides entries before the coercer's next inspection.
**Deployability - High:** Requires front-end changes (e.g., a "hide" button) and corresponding API-level filtering.
**Regulatory fit - High:** Server logs remain intact, satisfying retention rules.

Here the voter interface itself acts as a wrapper, giving discretionary control over what metadata may leak. A comparable situation arose in Norway's 20112013 i-voting pilots [1], where return codes remained on voters' phones unless actively deleted.

### 4.5 Masking: Dummy Log Entries

The trust-service provider could inject realistic, cryptographically signed dummy entries mimicking voting signatures, obfuscating the real pattern.

**Advantages:** Even a visible log becomes ambiguous.
**Drawbacks:** Dummy management is complex and may pose legal risks if fake entries are challenged.
**Effectiveness - Medium:** Provides plausible deniability but may be overcome by sophisticated adversaries.
**Deployability - Low:** Requires dummy generation, separate signing keys, and retention policies.
**Regulatory fit - Low:** Artificial entries undermine evidential reliability.

The dummy-entry generator serves as a wrapper, blurring the leakage channel rather than sealing it. A similar obfuscation strategy is employed in the *VoteAgain* system [13], which introduces dummy ballots to obscure voting patterns.

### 4.6 Encryption: Controlled Log Access

Vote-related entries could be encrypted with a dual-key scheme; decryption would require the joint presence of the voter and a service operator at a physical office.

**Advantages:** Remote coercers cannot read the log; audit access remains possible.
**Drawbacks:** Requires new hardware security modules, procedures, and joint in-person sessions with operators, creating inconvenience. Effective only if the persistent log filter is already in place.
**Effectiveness - High:** Online leakage is blocked completely.
**Deployability - Low:** Significant infrastructure changes and new workflows are required.
**Regulatory fit - High:** Logs remain intact and auditable; only routine visibility is curtailed.

The dual-key encryption scheme and physical service point form a wrapper that blocks automated access while preserving auditable evidence. This mirrors "break-glass" protocols in healthcare, where dual control is required to access sensitive records [7].

### 4.7 Comparative Summary of Mitigation Options

The triplet after each option reports *Effectiveness* | *Deployability* | *Regulatory fit* (H = High, M = Medium, L = Low).

**Persistent log filter - H|H|H:** Quick fix with limited transparency loss.

**Separate voting credential** **H|L|H:** Strongest long-term solution; significant deployment burden.

**Offline signing** **M|M|H:** Strong for ID card users; excludes mobile credentials.

**User-controlled hiding** **M|H|H:** Inexpensive to implement, but shifts responsibility to voters.

**Dummy entries** **M|L|L:** Adds noise but weakens audit evidence.

**Controlled log access** **H|L|H:** Robust protection; costly and complex.

A phased response appears pragmatic. Introducing a persistent front-end filter offers quick and inexpensive mitigation against routine coercion. In parallel, exploring a dedicated voting credential provides a structurally stronger long-term solution, cleanly separating electoral activity from general-purpose e-ID usage.

## 5 Discussion

This section interprets the technical findings in a broader socio-legal context and reflects on the limitations of the present study. Five themes are addressed: the central finding, legal inconsistency, the interplay of verifiability and secrecy, threats to validity, and implications for future work.

### 5.1 Central Finding: Composition Matters

The analysis shows that coercion resistance in Estonian i-voting hinges on the secrecy of voting frequency. That secrecy collapses when the national eID transaction log is combined with the re-voting protocol, even though both components are sound in isolation. The result is a metadata-based receipt that a coercer can exploit without breaking cryptography or gaining privileged access. Similar composition pitfalls have appeared in Norway and Switzerland, where otherwise benign transparency artefacts became coercion tools. The lesson is clear: security claims must be re-evaluated whenever an election system is embedded in a wider digital infrastructure.

### 5.2 Legal Inconsistency: Two Retention Regimes

Section $48^6(67)$ of the *Riigikogu Election Act* requires the election service to erase or anonymise personal log data once the count is complete and the result certified. By contrast, Article 24 (2)(h) of the eIDAS Regulation obliges a qualified trust-service provider (TSP) to "*record and keep accessible, for as long as necessary after the activities of the provider have ceased, all relevant information concerning data issued and received ... for the purpose of providing evidence in legal proceedings and for the purpose of ensuring continuity of the service*" [5].

ETSI EN 319 401 repeats the same principle in Clause 7.10 *Collection of evidence*: "*The TSP shall record and keep accessible for an appropriate period of time, including after the activities of the TSP have ceased, all relevant information concerning data issued and received ...*" [6].

The qualified TSP implements these texts through its privacy policy, which retains audit data related to qualified electronic signatures for at least ten years [22].

As a result, the same electronic ballot falls under two different retention rules: the election subsystem must purge identifying data soon after polling day, whereas the remote-signing subsystem keeps a linkable audit trail for a decade or more. This clash jeopardises the Election Act's intent that the system must not enable a voter to demonstrate which electronic ballot was ultimately counted.

Reconciling the regimes will require either (i) a statutory carve-out that explicitly exempts election signatures from the election-log deletion rule, or (ii) a technical wrapper that keeps the long-term eID audit trail hidden from routine disclosure while still fulfilling the provider's retention duty.

### 5.3 The Interplay of Verifiability and Secrecy

Fixing the eID log leak is a necessary first step, but it does not resolve the broader tension between verifiability and secrecy. This is illustrated by proposals for a public bulletin board (PBB)-a transparency mechanism currently absent from Estonian i-voting.

A PBB's primary goal is universal verifiability: publishing all received ballots (or their cryptographic checksums) allows voters and observers to verify that no ballots were dropped or altered. However, a PBB that explicitly marks each voter's final ballot [16] achieves verifiability at the cost of re-voting secrecy, creating an explicit receipt.

A checksum-only PBB is a privacy-preserving alternative avoiding the explicit receipt problem. However, this PBB alone cannot prevent a malicious client from tricking the voter into signing a substituted final ballot, as Pereira shows [16].

Thus, a holistic solution requires architectural layering: the system must first guarantee what-you-see-is-what-you-sign to defend against client-side attacks. Only then can a checksum-only PBB meaningfully defend against a malicious server. Sealing the existing eID log leak is a prerequisite for this entire architecture to become coercion-resistant.

### 5.4 Threats to Validity

Four limitations should be noted.

**Scope:** The study focuses on national eID transaction logs and does not analyse other potential side-channels such as network traffic or client logs.

**Behavioural Data:** No large-scale empirical survey of voter responses to log visibility was conducted; deterrence is inferred from plausibility and precedent.

**Dynamic Environment:** Regulatory frameworks and platform features evolve; mitigation proposals must therefore be revisited before deployment.

**Legal interpretations:** The author is not a legal expert. All regulatory interpretations in this paper reflect a technical perspective and should be independently verified from a legal viewpoint before policy implementation.

These caveats do not negate the main argument but indicate areas for further research.

### 5.5 Implications for Future Work

Future research could quantify the behavioural impact of log visibility through controlled user studies. Legal scholars may examine how trust-service regulation could accommodate election-specific secrecy requirements without weakening auditability in other domains. Finally, system designers should incorporate composition-aware risk assessment into standard election-technology life cycles.

## 6 Conclusion and Recommendations

This paper has shown that the national eID transaction log exposes the number and timing of electronic ballots, thereby defeating the secrecy on which Estonia's re-voting safeguard depends. The vulnerability is not rooted in cryptographic failure but arises from the composition of two otherwise secure systems: the re-voting protocol and the long-term audit trail mandated for qualified electronic signatures. Because the leakage involves only metadata, it persists even if ballot content remains perfectly encrypted. The result is a credible receipt exploitable for coercion at negligible technical cost.

### 6.1 Actionable Recommendations

To mitigate the issue, four concrete actions are proposed, each varying in complexity and timeframe:

1. **Immediate Mitigation Persistent Log Filter:** Hide vote-related entries in the web portal by default, retaining physical access for voters who need it. This change is inexpensive, complies with eIDAS retention rules, and blocks the routine side-channel for *all* credential types.
2. **Long-Term Option Separate Voting Credential:** Issue an election-specific certificate or token with its own audit trail, isolating voting activity from everyday eID use without restricting voter access to their data.

3. **Regulatory Alignment:** Amend either trust-service regulation or Estonia's election law to resolve the conflict between the decade-long retention of eID transaction data and short-lived election logs.
4. **Composition-Aware Risk Analysis:** Make cross-system interaction analysis mandatory whenever the Estonian i-voting scheme or its supporting infrastructures undergo modification, explicitly addressing the risks identified by composition analysis.

Successful implementation will require coordination among election officials, trust-service providers, regulators, legislators, and voters.

### 6.2 Final Remark

Democratic legitimacy depends not only on cryptographic soundness, but also on the social realities of voter autonomy. As digital infrastructures expand, election systems must be assessed in the context of these broader ecosystems. Only ongoing, multidisciplinary analysis-spanning technical, legal, and social domains-can prevent well-intentioned features from interacting in harmful ways.

**Acknowledgments.** The author used OpenAI's ChatGPT(OpenAI. ChatGPT (model o3), 2025. https://www.openai.com/) to improve language and structure. All research ideas, analysis and conclusions are solely the author's.

## References

1. Barrat, J., Chevalier, M., Goldsmith, B., Jandura, D., Turner, J., Sharma, R.: Internet voting and individual verifiability: the Norwegian return codes. In: 5th International Conference on Electronic Voting 2012 (EVOTE2012), pp. 35–45. Gesellschaft für Informatik e.V., Bonn (2012). https://dl.gi.de/items/e9eeff35-f08c-4e0f-ab6d-47bbc8a5ea6e
2. Clark, J., Hengartner, U.: Selections: internet voting with over-the-shoulder coercion-resistance. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 47–61. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27576-0_4
3. Drechsler, W., Madise, Ü.: Electronic voting in Estonia. In: Kersting, N., Baldersheim, H. (eds.) Electronic Voting and Democracy: A Comparative Analysis, pp. 97–108. Palgrave Macmillan UK, London (2004). https://doi.org/10.1057/9780230523531_6
4. Estonian Academy of Sciences, Cybersecurity Committee: TA küberturvalisuse komisjoni valimiste tehnoloogia riskianalüüs 2024. https://koodivaramu.eesti.ee/riigi-valimisteenistus/valimised-turve/-/blob/main/2024/TA-k%C3%BCberturvalisuse-komisjoni-valimiste-tehnoloogia-riskianal%C3%BC%C3%BCs-2024.pdf (2024)
5. European Parliament and Council: Regulation (EU) no 910/2014 of the European parliament and of the council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/ec. https://eur-lex.europa.eu/eli/reg/2014/910 (2014)

6. European Telecommunications Standards Institute: ETSI EN 319 401 v3.1.1: Electronic Signatures and Infrastructures (ESI); General Policy Requirements for Trust Service Providers. https://www.etsi.org/deliver/etsi_en/319400_319499/319401/03.01.01_60/en_319401v030101p.pdf (2024)
7. Ferreira, A., et al.: How to break access control in a controlled manner. In: 19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06), pp. 847–854 (2006). https://doi.org/10.1109/CBMS.2006.95
8. Ganta, S.R., Kasiviswanathan, S.P., Smith, A.: Composition attacks and auxiliary information in data privacy. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 265–273. KDD '08, Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1401890.1401926
9. Haines, T., Lewis, S.J., Pereira, O., Teague, V.: How not to prove your election outcome. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 644–660 (2020). https://doi.org/10.1109/SP40000.2020.00048
10. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, pp. 61–70. WPES '05, Association for Computing Machinery, New York, NY, USA (2005). https://doi.org/10.1145/1102199.1102213
11. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9
12. Liive, R.: Valimisteenistus seletab lahti, miks digiaktivisti väiteid e-hääle salajasuse murdmise osas ei vasta tõele. Geenius Meedia (2023). https://digi.geenius.ee/rubriik/uudis/valimisteenistus-seletab-lahti-miks-digiaktivisti-vaiteid-e-haale-salajasuse.murdmise-osas-ei-vasta-toele/
13. Lueks, W., Querejeta-Azurmendi, I., Troncoso, C.: VoteAgain: a scalable coercion-resistant voting system. In: 29th USENIX Security Symposium (USENIX Security 20), pp. 1553–1570. USENIX Association (2020). https://www.usenix.org/conference/usenixsecurity20/presentation/lueks
14. Martens, T.: Electronic identity management in Estonia between market and state governance. Ident. Info. Soc. **3**(1), 213–233 (2010). https://doi.org/10.1007/s12394-010-0044-0
15. Norwegian Ministry of Local Government and Modernisation: Internet voting pilot to be discontinued. https://www.regjeringen.no/en/historical-archive/solbergs-government/Ministries/kmd/press-releases/2014/Internet-voting-pilot-to-be-discontinued/id764300/ (2014)
16. Pereira, O.: Individual verifiability and revoting in the Estonian internet voting system. In: Matsuo, S., Gudgeon, L., Klages-Mundt, A., Perez Hernandez, D., Werner, S., Haines, T., Essex, A., Bracciali, A., Sala, M. (eds.) Financial Cryptography and Data Security. FC 2022 International Workshops, pp. 315–324. Springer International Publishing, Cham (2023). https://doi.org/10.1007/978-3-031-32415-4_21
17. Pereira, O., Teague, V.: Report on the SwissPost-Scytl e-voting system, trusted-server version. https://www.bk.admin.ch/dam/bk/de/dokumente/pore/E-Voting_Report_Pereira_Teague_Juli%202019.pdf.download.pdf/E-Voting_Report_Pereira_Teague_Juli%202019.pdf (2019)
18. Põder, M.: Hard evidence of an e-vote in the IVXV protocol. Lightning talk, E-Vote-ID 2023 (2023). https://infoaed.ee/proof2023/
19. Põder, M.: Kuidas tõendada oma e-häält? https://youtu.be/vWMaoqPY2-M (2023). YouTube

20. Riigikogu: Riigikogu valimise seadus. https://www.riigiteataja.ee/akt/124052024010 (2024). English Translation available
21. Sewell, P., Vitek, J.: Secure composition of insecure components. In: Proceedings of the 12th IEEE Computer Security Foundations Workshop. pp. 136–150 (1999). https://doi.org/10.1109/CSFW.1999.779769
22. SK ID Solutions AS: Principles of processing personal data (privacy policy). https://www.skidsolutions.eu/upload/files/SK-POPPD-EN-CURRENT.pdf (2022)
23. Spence, A., Bangay, S.: Security beyond cybersecurity: side-channel attacks against non-cyber systems and their countermeasures. Int. J. Inf. Secur. , 1–17 (2021). https://doi.org/10.1007/s10207-021-00563-6
24. State Electoral Office of Estonia: Elektroonilise hääletamise üldraamistik ja selle kasutamine Eesti riiklikel valimistel (2024). https://www.valimised.ee/sites/default/files/2024-05/RVT%20korraldus%20nr%2011%20lisa%20%28IVXV%20EHS%20%C3%BCldraamistik%29.pdf
25. Supreme Court of Estonia: Riigikohtu põhiseaduslikkuse järelevalve kolleegiumi otsus 3-4-1-13-05. https://www.riigikohus.ee/et/lahendid?asjaNr=3-4-1-13-05 (2005). English translation available
26. Yarom, Y., Falkner, K.: Flush+reload: a high resolution, low noise, L3 cache side-channel attack. In: 23rd USENIX Security Symposium (USENIX Security 14), pp. 719–732. USENIX Association, San Diego, CA (2014). https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/yarom



# Recommendations to OSCE/ODIHR (on How to Give Better Recommendations for Internet Voting)

Jan Willemson(✉)

Cybernetica, Narva Mnt 20, 51009 Tartu, Estonia
jan.willemson@cyber.ee

**Abstract.** This paper takes a critical look at the recommendations OSCE/ODIHR has given for the Estonian Internet voting over the 20 years it has been running. We present examples of recommendations that can not be fulfilled at all, but also examples where fulfilling a recommendation requires a non-trivial trade-off, potentially weakening the system in some other respect. In such cases OSCE/ODIHR should take an explicit position which trade-off it recommends. We also look at the development of the recommendation to introduce end-to-end verifiability. In this case we expect OSCE/ODIHR to define what it exactly means by this property, as well as to give explicit criteria to determine whether and to which extent end-to-end verifiability has been achieved.

## 1 Introduction

Organization for Security and Co-operation in Europe (OSCE) through its Office for Democratic Institutions and Human Rights (ODIHR)[1][2] is one of the major international organizations engaged in election observation. Targeting primarily the electoral processes of the 57 OSCE member states[3], ODIHR's contribution to setting the standards for international election observation reaches far beyond OSCE. This is achieved thanks to publishing very detailed observation reports and an excellent series of election handbooks.[4]

ODIHR reports cover all the major aspects of elections from candidate registration, minority issues, campaign financing and media to election administration and assessment of technical details of voting. The latter also covers electronic methods including voting machines and vote casting over Internet.

Several OSCE member states (including Armenia, Canada, Estonia, France, Norway, Russia, Switzerland and United States of America) have allowed Internet voting at various times and circumstances. Out of these countries, Estonia

[1] https://www.osce.org/odihr

[2] Even though the full formal acronym of the organization is OSCE/ODIHR, we will just use ODIHR in this paper for brevity.

[3] https://www.osce.org/participating-states; despite the organization's name referring to Europe, there are also several member states from Asia and North America.

[4] https://www.osce.org/odihr/elections/handbooks


D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, pp. 208–223, 2026.
https://doi.org/10.1007/978-3-032-05036-6_13

has enabled casting votes via Internet in legally binding elections continuously the longest, since 2005. (For a historical overview and some technical details of Estonian Internet voting we refer the reader to Ehin *et al.* [13].) By 2025, Estonia has had 5 municipal, 5 parliamentary and 4 European Parliament elections allowing to cast the votes over Internet.[5]

Since ODIHR observes parliamentary and presidential elections, its missions have produced five final reports covering Internet voting in Estonia.[6] These reports provide a valuable insight into the development of Internet voting in general, extending beyond the mere Estonian case study.

This paper looks back at the 20 years of Internet voting in Estonia through the lens of ODIHR reports. We review some of the (more controversial) recommendations given in the reports, their context and evolution over the years.

## 2 Background

ODIHR has produced five final reports on Estonian Internet-enabled parliamentary elections over the years[7], respectively from 2007 [1], 2011 [2], 2015 [3], 2019 [4] and 2023 [5]. These reports typically start from describing the general operational principles of the system, followed by the observations made by the rapporteurs, and recommendations.

Even through ODIHR has no legal authority over the country's electoral system, the recommendations are still considered as guidelines for improvement. Systematic failure to meet some of the recommendations may shape the attitude of international community regarding the state of democracy in the country. This is why authorities generally try to act upon the ODIHR guidelines. In the countries where democratic traditions are not yet well established, these recommendations may be used e.g. by the civil society organizations to pressure the government to implement some changes.

On the other hand, potential reputation issues and pressure towards the authorities also mean that the recommendations should not be given lightly. It is easy to write a few sentences into a report, but if implementing them turns out to be unreasonable or even impossible, the recommendations may cause more harm than good.

This has regrettably been the case in Estonia. Even if a recommendation is impossible to implement, failure to do so can be used by politically-motivated actors to spread distrust in the electoral system, including Internet voting.

The goal of this paper is to give some recommendations to ODIHR on how to ensure that their recommendations are actually implementable and achieve a reasonable balance between the conflicting requirements of voting.

[5] https://www.valimised.ee/en/archive/statistics-about-internet-voting-estonia

[6] In Estonia, presidential elections are not national, but are held in the parliament or by an extended electoral body comprised of members of the parliament and representatives of the municipal councils.

[7] ODIHR also produces other kinds of documents, for example Needs Assessment Mission reports and preliminary versions of the final reports. However, the official recommendations are given in the final versions, and thus we concentrate on them.

Table 1 presents a short summary of the five reports, showing how many pages were devoted to Internet voting, and how many formal recommendations were given.

**Table 1.** ODIHR reports about Estonian Internet voting

| Year | Pages | Recommendations |
|---|---|---|
| 2007 | 12.5 | 9 |
| 2011 | 7 | 13 |
| 2015 | 4 | 8 |
| 2019 | 2 | 5 |
| 2023 | 3.5 | 6 |

It is natural that in the first years more coverage was devoted to Internet voting as that was a very new thing, even at the international level. As we see, over the years 41 recommendations have been issued, with some of them being repeated or partially overlapping. The topics range from the proposals to update the legislation to very specific cryptographic ideas. Due to the page limit of the current paper we are unable to cover them all. Rather, we will be concentrating on the more interesting findings.

## 3 Vote Secrecy and Coercion Resistance

Vote secrecy is one of the core requirements of democratic elections. Its goal is to ensure voting freedom by making it hard for the potential coercer to learn the voter's true preference. Conventional polling station paper voting attempts to achieve vote secrecy by enforcing the voters to fill in the ballots in private voting booths. This worked well in the 19th century Australia [9], and in the rest of the democratic world for the next 100. . . 150 years.

Due to the apparent success of secret ballot, this requirement is often listed among desirata for other modes of voting (including remote ones) as well. This is also emphasized by ODIHR e.g. in their 2007 report [1] which states:

> "Because the [Internet] voter is not voting in a supervised and controlled environment such as a polling station, it cannot be ensured that the voter is casting his/her vote in secret."

However, by the end of the 20th century, technological advancements in the recording equipment significantly weakened privacy guarantees of the polling booths [7]. It is possible to intercept the vote via various side channels [10,20, 21,24] or simple video recording[8][9][10].

[8] https://www.bbc.com/news/world-asia-56377642, accessed January 30th, 2025.
[9] https://www.timesofisrael.com/hidden-cameras-in-arab-voting-booths-were-netanyahus-idea-tv-report/, accessed January 30th, 2025.
[10] https://www.bbc.com/news/world-asia-68707488, accessed January 30th, 2025.

To compensate for the loss of privacy and potential coercion issues, Estonia offers the possibility to re-cast one's electronic vote in case the voter has felt coerced to vote against his/her true preference.[11][12] As a result, we can argue that the system becomes more secure against coercion than polling station paper voting. There it is enough for the coercer to observe (e.g. via video transmission) one act of paper voting and be sure that the vote will be counted as cast as there is no way to change a coerced paper vote afterwards.

Of course, the information concerning which electronic vote is the last one and will be tallied becomes sensitive as knowing it would help the coercer to achieve his goal. This is also noted by ODIHR in 2007 [1]:

> "However, the OSCE/ODIHR EAM noted that one technical aspect of the system undermines the objective of the recast possibility. Namely, the vote storage server records the time that each voter casts his/her last electronic vote. This log, which is available to political parties and observers, could potentially be misused to know whether a voter did in fact recast his/her vote electronically.
>
> The OSCE/ODIHR EAM recommends that the NEC consider modifying the design of the internet voting system so that the time of voting is not recorded. In the interests of maintaining the transparency of the system, however, the log should continue to be available to observers."

This recommendation was repeated in 2011 [2].

Even though the concern of timed logs being sensitive is a valid one, just not recording the timestamps is not a viable solution. First of all, as noted above, these timestamps are needed to determine which one of the re-cast votes should be tallied. And second, the i-votes are digitally signed to provide both eligibility and integrity of the votes. The Digital Signature Act (later superseded by the EU eIDAS regulation) explicitly requires digital signatures to have timestamps to be able to determine whether the signature was given during the validity period of the signature key certificate.

> *We recommend that, before giving recommendations, ODIHR analyses whether the prospective recommendation can be implemented in the first place, both from the technical and legal perspective.*

Another potential issue with vote secrecy is presented in the 2019 report making use of the fact that the Estonian Internet voting protocol implements individual verifiability by making the encryption randomness available to the audit device via a QR-code [4]:

[11] Most of the protective effect of re-voting actually comes from deterrence – if the potential coercer knows that the voter can re-vote, he hopefully does not attempt to coerce the voter in the first place.

[12] Since 2021, it is also possible to change one's electronic vote by voting on paper in the polling station on election Sunday.

> "A review of operational and technical frameworks by the ODIHR EET indicates that an internal attacker with privileged access to digital ballots could break the vote secrecy of any voter who published an image of the QR code online, even after the expiry of the code's validity. This contradicts national legislation and international standards pertaining to vote secrecy"

It is noteworthy that an actual attack scenario is described here (which is something ODIHR does not do very often). Let us analyze this scenario in more detail.

The attack involves two parties:

1. an internal attacker with privileged access to the digital ballots, and
2. a voter wilfully making his/her verification QR-code public.

The attack has two steps:

1. the voter publishes his/her individual verification QR-code online, and
2. the insider uses the encryption randomness within the QR-code to open the digital ballot.

As a result, ODIHR recommends mitigating internal attacks, completely forgetting that the attack needs a voter willing to violate the secrecy of their own vote in the first place!

In the paper voting domain, the situation is more or less equivalent to a voter taking a selfie together with their filled in ballot (sometimes called *stemfie* from the Dutch word *stemmen*), and posting it online. No authority can essentially prevent it (even though some jurisdictions have attempted to outlaw stemfies, with varying degrees of success [18]). Internet voting is actually considerably more secure against this type of attack, as besides the voter wilfully opening his/her vote it would also require a corrupt insider.

> *We recommend that ODIHR gives a thorough consideration to the attacker models and required assumptions before publishing attack scenarios to justify the recommendations. Among other things, ODIHR should decide whether a voter interested in violating the secrecy of his/her own vote is considered an attacker or not. If yes, ODIHR should refer to this as a threat in all the reports about paper voting. Also, ODIHR should in this case present a clear strategy to fight against stemfies. If the voter wilfully publishing their vote is not an attacker, ODIHR should retract the above-cited paragraph from their 2019 report as not describing an actual attack.*

## 4 End-To-End Verifiability

Next to vote secrecy, integrity-related goals form another (and perhaps an even more important) set of requirements on voting systems. Integrity attacks can

occur against almost all the steps of voting (e.g. the votes could be altered or deleted, the ballot box could be stuffed with ineligible votes, the tally could be computed incorrectly, etc.). Hence, in order to make sure that the end result truly reflects societal preferences, each step should be verifiable.

The explicit properties to consider, and even a potential technical approach, are first referred to in the 2011 ODIHR report [2]:

> "In recent years, advances have been made in the field of cryptography to enable end-to-end verification of the votes cast, i.e. a possibility for an individual voter to verify that his/her vote was (i) cast as intended, (ii) recorded as cast, and (iii) counted as recorded. Such individual verifiability usually relies on giving the voter a code that allows him/her to check later whether their vote was correctly recorded or even counted."

This is a fascinating statement in several aspects. The first ideas about verifiable online voting can be tracked back to 1980 s to the pioneering works of Benaloh *et al.* [8,11]. However, by early 2010 s, the terminology still hadn't settled, which can be seen from the above quote essentially equating end-to-end (E2E) and individual verifiability. In recent research, it is more customary to follow the verifiability categories by Kremer *et al.* [19]:

- *Individual verifiability*: a voter should be able to check that his vote belongs to the ballot box.
- *Universal verifiability*: anyone should be able to check that the result corresponds to the content of the ballot box.
- *Eligibility verifiability*: only eligible voters may vote.

For a good overview of different proposed verifiability notions and comparison of their formal definitions we refer to the 2016 paper by Cortier *et al.* [12].

Thus, following the Kremer-style approach, only (i) cast-as-intended and (ii) recorded-as-cast checks would fall under individual verifiability, whereas (iii) counted-as-recorded would nowadays be called universal verifiability.

Whether individual, universal and eligibility verifiability combined give rise to E2E verifiability or not, is still a matter of ongoing academic discussion (and ultimately a matter of definition). For example, the recent report by German BSI [22] gives the following definition:

> "Let $\tau$ be a trust assumption, and let $\mathcal{A}$ be a set of algorithms/attacks that an attacker can execute under the trust assumption $\tau$. We say that a voting system is end-to-end verifiable under the trust assumption $\tau$ against $\mathcal{A}$ if under any attack $A \in \mathcal{A}$ (obeying the trust assumption $\tau$), the probability that the final result published by the voting system is accepted, even though this result does not match the voters' votes, is negligible."

An important feature of this definition is that here E2E verifiability is not a yes/no property, but is parametrized by the trust assumptions and a negligibility threshold. Hence, virtually any voting system is E2E verifiable, but some may need many trust assumptions to be satisfied. Intuitively we may say that such

systems are E2E verifiable only in a very weak sense, and we would instead like to have strong verifiability (i.e. as few trust assumptions as possible).

Unfortunately, the report [22] does not state which level of verifiability can be considered good enough.[13][14] This also holds true for the ODIHR reports.

After the 2011 ODIHR report and emergence of a proof-of-concept malicious voting application [16], Estonia implemented a solution for cast-as-intended and recorded-as-cast, or simply individual verification [17].[15] This was noted positively in the 2015 report, however, ODIHR rapporteur drew attention to lacking counted-as-recorded (or universal) verifiability [3].

The corresponding mechanisms were added with the 2017 update to the Estonian Internet voting system, code-named IVXV [13,15]. 2019 ODIHR report again took a positive note on the development, but this time the rapporteur gave a remark on insufficient safeguards against insider attackers [4]. As a motivating example, a scenario involving a voter leaking his/her individual verification QR-code was given. We studied this scenario in detail in Sect. 3 and concluded that Internet voting is actually more secure than paper voting against this type of attack exactly because it additionally needs a dishonest insider.

The 2023 report [5] describes another potential scenario.

> "However, it is known that the voter verification mechanism is vulnerable to compromise if the voting client application is altered. While it would be difficult to deploy this attack vector at scale and undetected, the possibility of this type of attack indicates a critical deficiency in the current design of the voter verification step. Namely, the application could be programmed to crash immediately after collecting the information about the properly submitted ballot, but before using this information to conduct the verification. If the voter restarts the application and attempts to vote again, possibly upon incorrectly assuming that their ballot has not submitted, the altered application would then first submit the ballot with an altered choice, but would perform the voter verification procedure on the ballot submitted before the crash, and the voter verification application would display the voter's original choice."

The rapporteur is referring to the attack by Pereira [23] making use of the fact that the current IVXV individual verifiability protocol does not reveal whether the audited ballot was the last one submitted by the voter. This is a conscious design decision taken to minimize the threat of coercion in the scenario where the attacker demands verification of the QR-code to see if it still verifies the vote after some time (attempting to make sure that the voter has not re-voted).

[13] Still, the report thoroughly assesses practicality of various possible approaches to verifiability, hence acknowledging that there is potentially a verifiability-practicality trade-off, where a good balance needs to be found.

[14] Also, the scope of the report [22] is limited to studying various verifiability techniques in isolation, and no attempt is made to evaluate more complex systems featuring a combination of these techniques.

[15] The Estonian approach to individual verification is now known as cast-and-audit [22].

This is one example of the conflict between verifiability and coercion resistance requirements. Pereira's attack is easy to circumvent by adding the freshness check, but that would increase coercibility to some degree. Is this an acceptable trade-off from the ODIHR point of view?

*We recommend that ODIHR explicitly considers the potential trade-offs implied by implementing its recommendations, and takes an explicit stance concerning which one of the trade-offs is the best.*

The 2023 report [5] also proposed an additional auditing step in the universal verification stage. This step was added for the 2024 European Parliament elections.

However, all these examples bring us to a bigger question mentioned earlier in the section – what are the criteria under which a voting system can be considered end-to-end verifiable at a sufficient level?

Consider the above-described attack by Pereira [23] as an example. Is the possibility of such an attack a reason to claim that IVXV does not have (a sufficient level of) E2E verifiability? Let us study the assumptions the attacker has to fulfill.

First of all, the attacker has to develop and distribute a malicious voting application. The ability to write an application is not a big problem for a mid-level programmer (so no remarkable assumptions here), but distributing and getting it run undetected is not so easy. All the following assumptions need to hold:

1. the voter is successfully directed to an unofficial distribution channel,
2. the voter does not verify authenticity of the voting application, and
3. the voter does not report suspicious crash of the application.

For assumption 1, it is easy to set up a lookalike website, but it will very probably be noticed and reported. For assumption 2, we need to take into account that under Windows and macOS the voting application is signed with a developer key and the OS verifies the signature before running it. It is possible for an attacker to register as a developer, but this will leave more traces. Linux users are supposed to verify the checksum of the application themselves, but on the other hand, Linux users are more likely to do it. For assumption 3, we note that in order to have a significant effect, the attacker needs to manipulate many votes. However, the probability that no crashes will be reported decreases exponentially fast in the number of crashes (see below for the computations).

The attack scalability issue is also noted by ODIHR [5] ("... it would be difficult to deploy this attack vector at scale and undetected ..."). But how does this translate to the level of (E2E) verifiability? Let us use the above-cited BSI definition [22] stating "... the probability that the final result published by the voting system is accepted, even though this result does not match the voters' votes, is negligible."

What does the "final result" mean here? In general, the voting system produces two results – the voting result (how many votes did every candidate get)

and the election result (who got the seats). We argue that the requirement that the voting result has to exactly match the sum of the individual preferences is too strong. Most notably, paper voting does not guarantee this requirement as the preferences are transferred to the ballots by hand. Human involvement is needed to interpret and count the ballots, and this process is known to be inherently imprecise [14].

Paper voting operates under the assumption that if there is no large-scale systematic violation (e.g. many polling station workers stuffing the ballot boxes[16] [17] [18] [19]), the stochastic mistakes cancel out and the election result is a fair reflection of the voters' preferences.

Of course, against large-scale paper voting violations there are safeguards in place. The ballot boxes are publicly observable, so in principle it is possible to at least detect, although not always physically prevent, malicious activities like ballot box stuffing. However, even detection of large scale fraud may be sufficient as it is possible to call the elections void (i.e. not to accept the result) in case the reports of fraudulent actions are frequent.

This is exactly what is happening in case of the Pereira attack in IVXV. Even if the attacker successfully distributes a malicious voting application and gets the voters to run it, they still must notice the crash. Some percentage of the voters will also report it, and this way the attack gets detected by the election organizer. The matter can be studied and, in case the reports are frequent, the election result may get rejected.

If there are $n$ voters who run the manipulated application, and a voter has probability $p > 0$ to report the crash, then the probability that the attach remains unreported is $(1-p)^n$ which converges to 0 exponentially fast in $n$. Thus, the probability that the manipulated result gets accepted undetected is negligible[20].

We conclude that Pereira's attack, even though possible in principle, does not necessarily violate E2E verifiability (at least the BSI flavour of it). Of course, there are more details to consider, e.g. how does the reporting frequency affect the above probability computation, and how frequent do the reports have to be to get the election result rejected. The latter question is also raised in the 2011 ODIHR report [2]:

> "Although the Election Act indicates that the NEC can invalidate the results of the Internet voting, it does not specify on which basis and under which circumstances the results of the Internet voting can be declared

[16] https://www.youtube.com/watch?v=5UFIQamKbjg
[17] https://www.youtube.com/watch?v=Ep1ef3_wLX0
[18] https://www.newsflare.com/video/634198/video-emerges-of-ballot-box-being-stuffed-during-russias-presidential-election
[19] https://globalnews.ca/video/4092550/russian-polling-station-workers-accused-of-stuffing-ballot-box-11-times-within-13-minutes/
[20] We say that a function $\mu : \mathbb{N} \to [0;1]$ is negligible if for every positive polynomial $p$ there exists $N \in \mathbb{N}$ such that for every $n > N$, the inequality $\mu(n) < \frac{1}{p(n)}$ holds. It can be shown that for any $q \in [0;1)$, the function $q^n$ is negligible.

> invalid. It further does not specify how and by which means voters can be informed that they have to recast their vote on paper on election day.
>
> The OSCE/ODIHR recommends that legal provisions with regards to all stages of the Internet voting, including conditions for invalidation of the Internet voting results, are further detailed and consolidated in the law."

The intent of this recommendation is good, but it is not clear what is the correct level of technical detail such conditions should be written into the law. Invalidation of (part of) the voting results is a very prominent action, and the state may want to retain the discretion to decide flexibly based on the concrete circumstances that are hard to foresee in the time of writing the legislation. Thus, in 2017 an amendment in the following wording entered force[21]:

> "... the National Electoral Committee has the right ... to annul the votes cast in the advance voting in part or in whole due to material violation of law and call on the voters to vote again during advance voting or on the election day;"

Thus, the flexibility to consider the circumstances is there. On the other hand this of course means that it is impossible to give a strict mathematical proof that depends on the exact conditions for annulling the votes.[22]

However, these details do not change our main message – just the existence of some attacks does not automatically mean that the system is not E2E verifiable on a sufficient level.

This brings us to the next important question – how does one exactly decide whether a given voting system is (sufficiently) E2E verifiable? We have been unable to identify an existing good methodology for that in the literature. The BSI report [22] explicitly leaves complete systems out of scope, and the recent ODIHR handbook on observing ICT in elections [6] does not present any methodology either.

However, we argue that such a methodology is urgently needed. If ODIHR never actually says that the system is good enough, and only keeps giving recommendations on issues that actually do not violate E2E verifiability, it becomes increasingly hard for the election organizers to do their job.

[21] https://www.riigiteataja.ee/en/eli/ee/510032014001/consolide/current

[22] Extending the argument we can see that this problem is inherent. The target of verification is detecting anomalies. In case of elections, the anomalies can hardly be fixed, so the only option is to cancel the voting event and try again. A precise mathematical definition of E2E verifiability hence has to rely on the (mathematical) criteria for annulling the result. In the real life, however, the annulling decision is taken by the lawyers who prefer not to be pre-committed to strict criteria, but rather have the power of discretion. It is ultimately the question of who gets to decide upon annulling the elections – mathematicians or lawyers. The current societal order seems to favour the lawyers in this role. Thus the mathematicians have to come up with E2E verifiability definitions that allow for the power of discretion for the lawyers, or to acknowledge that E2E verifiability can not be achieved in practice at all.

First, if there are no clear criteria that ODIHR uses to decide upon the level of E2E verifiability, it is unclear what should be the target of development. If the ODIHR rapporteur keeps pointing to marginal issues (and one can always find something to point to), the target keeps moving forever. Of course, the environment is changing, and the criteria should sometimes also change to follow the environment, but there should explicit and clearly communicated criteria at any given point in time. Moreover, these criteria should be communicated well ahead of the assessment in order to give sufficient time for their implementation.

And second, at least in the case of Estonia, any recommendations given by ODIHR get interpreted in the conservative media as evidence to support the narrative that electronic voting is insecure. Without the explicit claims in ODIHR reports referring to acceptable levels of verifiability, the election organizer is subjected to undue pressure from the critics. But of course, in order to give such claims, ODIHR first needs to create a methodology how such claims can be reached.

*We recommend that ODIHR develops and publishes clear criteria according to which it decides about (i) the level of verifiability of the studied voting system, and (ii) whether this level is sufficient or not.*

## 5 Recommendations that Do Not Help

With 41 recommendations, it becomes statistically likely that there are some among them that seem quite good at the first glance, but are impossible to implement or contradict some other requirements. In this Section we review some of the more interesting examples of this category.

The 2011 ODIHR report [2] notes and recommends the following.

> "Daily update of the voter register during the voting period as required by the Election Act was performed together with the daily backup of data. The project manager accessed the servers for daily data maintenance and backup breaking the security seals and using a data storage medium employed also for other purposes. This practice could potentially have admitted the undetected intrusion of viruses and malicious software.
>
> It is recommended that no maintenance of the Internet voting system servers is performed from the start to the end of the Internet voting process."

Using a single-purpose data transfer medium is indeed a good suggestion, but otherwise the recommendation to avoid maintenance violates the best practices of running IT-systems. Daily backups of the digital ballot box are essential to ensure continuity of elections even in the case of a major disruption. Also, daily updates of the voter register are required by the Election Act for a reason as the list of eligible voters changes every day (including in an unpredictable manner due to deaths). Thus, the running system needs to be accessed every day, and

physical access is actually easier to secure. Implementing a remote console access for these maintenance tasks would potentially be a considerably larger security vulnerability as it would be easier to misuse without detection.

> *We recommend that ODIHR aligns its recommendations with the best practices of maintaining IT systems, and considers alternatives before selecting its recommendations.*

The 2023 ODIHR report [5] reads:

> "During the election period, internet voting continued to enjoy a high level of public trust, owing to the transparency of the system and this voting method being an established practice. However, some voters distrust the results of internet voting, with notable divisions within the society between those who fully trust and those who fully distrust internet voting. This was also exemplified by the difference in the preferred political forces of those who voted in polling stations and those who voted online. This polarization was also represented in the political spectrum, with some parties resolutely supporting internet voting and other parties raising doubts before and after the elections, most notably EKRE. Following the announcement of results, some EKRE frontrunners, including its leader, made statements that the elections were stolen through internet voting and the party subsequently submitted several complaints requesting the annulment of internet voting, which were all dismissed. Notably, the allegations on systematic electoral fraud were made in the public domain without substantiation; such claims can harm public trust in democratic institutions. While the election authorities provided comprehensive voter information on its website and other platforms, they did not timely and proactively address some of the concerns raised by the contestants regarding the integrity of the internet vote.
>
> To further increase and maintain trust in internet voting, the election authorities should proactively address all concerns raised by election stakeholders who distrust the results of internet voting."

A similar recommendation was also given in 2019 [4]:

> "Other significant risks that may negatively affect public confidence in Internet voting include cyber-attack allegations from disinformation campaigns or human error.
>
> The SEO could review the potential effects of cyber-attack allegations against the Internet voting infrastructure, and develop a risk mitigation strategy."

While proactive communication is of course desirable and necessary, the above-cited recommendations are essentially impossible to implement in practice. First, the critics of Internet voting have shown remarkable creativity in coming up with a wide spectrum of claims, and addressing all of them proactively is

impossible. Second, it is important to realize that the claims and allegations against Internet voting are often not motivated by true concerns of security (in which case argumentation would be possible). Rather the goal of the opposing politicians seems to be discontinuing Internet voting altogether, hoping that younger and more liberal part of the electorate would then refrain from voting at all. One can not possibly address this goal via proactive communication.

*We recommend that ODIHR should not give recommendations that are impossible to fulfill.*

The 2023 report [5] recommends:

> "At the beginning of internet voting, on 27 February, the system was configured with an outdated voter register, which resulted in 63 voters casting votes for lists in their previous districts of registration before the misconfiguration was discovered and corrected. The voters were informed, and 59 of them voted again online, and one also voted at a polling station. The ballots of the four voters who did not vote again were revoked by the SEO, effectively invalidating their votes. The lack of quality assurance of configuration and other data can negatively affect the voters' trust in internet voting.
>
> To prevent errors or outdated information when configuring the components of the system, the election authorities should put in place a quality assurance process that includes the comprehensive testing of the internet voting system in its entirety before being deployed."

This was not an issue of the voting system misconfiguration, but rather a problem of the population registry not having updated information about the people who had recently moved to another electoral district.

The infrastructure required for Internet voting includes quite a few service providers, and population registry is just one of them. Having each provider continuously responsible for the quality of their service is a cornerstone of the Estonian e-state philosophy. Running specific tests for all the service providers before every state event is not reasonable. Rather it is worth noting how quickly it was possible to fix the problem, and the very limited impact it had as a result of the fast reaction.

*We recommend ODIHR to acknowledge that it is not reasonable to prevent all the potential problems. For some issues it is more resource-efficient to detect the problem, and have a team ready to act promptly on it.*

## 6 Conclusions

ODIHR has done an impressive amount of work to raise the standards of electoral integrity amongst its member states, but also on a larger international scale.

Both its observation reports and handbooks provide a valuable source for all the countries interested in improving their democratic processes.

Since introduction of Internet voting in Estonia in 2005, ODIHR has consistently reported on its state of development and given recommendations for improvement. Many of the recommendations have proven very useful and have been implemented in the legislation and/or in the system itself. Some, however, require further discussion and analysis.

ODIHR has relatively limited options for disputing or retracting its recommendations. There may be a good reason behind that – if raising a dispute would be too easy, ODIHR would need to put a lot of effort and valuable resources into dispute resolution.

On the other hand, if some recommendation is impossible to implement or significantly contradicts other requirements, it may end up causing more harm than good. Mere existence of unimplemented recommendations in ODIHR reports can be used to raise ungrounded allegations against the security of Internet voting.

Due to the lack of a better option, the author decided to share some of his thoughts in the form of an academic paper. The paper has also been deliberately formatted in the style of ODIHR reports.

The author hopes that the presented thoughts are useful for ODIHR (and other similar organizations) to improve their election observation processes, especially concerning Internet voting.

And, to remain true to the ODIHR style, let us conclude the paper with one final recommendation.

*We recommend that ODIHR makes discussing and disputing its recommendations more accessible. Also, clear process and criteria for retracting the recommendations should be developed.*

**Acknowledgments.** The paper has been supported by the Estonian Research Council under the grant number PRG2177.

## References

1. Estonia, Parliamentary Elections, 4 March 2007: Final Report. Tech. rep., OSCE/ODIHR (2007). https://www.osce.org/odihr/elections/estonia/25925
2. Estonia, Parliamentary Elections, 6 March 2011: Final Report. Tech. rep., OSCE/ODIHR (2011). https://www.osce.org/odihr/77557
3. Estonia, Parliamentary Elections, 1 March 2015: Final Report. Tech. rep., OSCE/ODIHR (2015). https://www.osce.org/odihr/elections/estonia/160131
4. Estonia, Parliamentary Elections, 3 March 2019: Final Report. Tech. rep., OSCE/ODIHR (2019). https://www.osce.org/odihr/elections/estonia/424229
5. Estonia, Parliamentary Elections, 5 March 2023: Final Report. Tech. rep., OSCE/ODIHR (2023). https://www.osce.org/odihr/elections/estonia/551179
6. Handbook for the Observation of Information and Communication Technologies (ICT) in Elections. OSCE/ODIHR (2024). https://www.osce.org/odihr/elections/558318

7. Benaloh, J.: Rethinking voter coercion: the realities imposed by technology. In: 2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013. USENIX Association (2013). https://www.usenix.org/conference/evtwote13/workshop-program/presentation/benaloh
8. Benaloh, J.C., Yung, M.: Distributing the power of a government to enhance the privacy of voters (extended abstract). In: Halpern, J.Y. (ed.) Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing, Calgary, Alberta, Canada, August 11-13, 1986pp. 52–62. ACM (1986). https://doi.org/10.1145/10590.10595
9. Brent, P.: The Australian ballot: not the secret ballot. Aust. J. Polit. Sci. **41**(1), 39–50 (2006). https://doi.org/10.1080/10361140500507278
10. Calandrino, J.A., Clarkson, W.: Some consequences of paper fingerprinting for elections. In: Jefferson, D., Hall, J.L., Moran, T. (eds.) 2009 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '09, Montreal, Canada, August 10-11, 2009. USENIX Association (2009). https://www.usenix.org/conference/evtwote-09/some-consequences-paper-fingerprinting-elections
11. Cohen, J.D., Fischer, M.J.: A robust and verifiable cryptographically secure election scheme (extended abstract). In: 26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985, pp. 372–382. IEEE Computer Society (1985). https://doi.org/10.1109/SFCS.1985.2
12. Cortier, V., Galindo, D., Küsters, R., Müller, J., Truderung, T.: SoK: verifiability notions for e-voting protocols. In: IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016, pp. 779–798. IEEE Computer Society (2016). https://doi.org/10.1109/SP.2016.52
13. Ehin, P., Solvak, M., Willemson, J., Vinkel, P.: Internet voting in Estonia 2005–2019: evidence from eleven elections. Gov. Inf. Q. **39**(4), 101718 (2022). https://doi.org/10.1016/j.giq.2022.101718
14. Goggin, S.N., Byrne, M.D., Gilbert, J.E.: Post-election auditing: effects of procedure and ballot type on manual counting accuracy, efficiency, and auditor satisfaction and confidence. Elect. Law J. Rules Polit. Policy **11**(1), 36–51 (2012)
15. Heiberg, S., Martens, T., Vinkel, P., Willemson, J.: Improving the verifiability of the Estonian internet voting scheme. In: Krimmer, R. (ed.) E-Vote-ID 2016. LNCS, vol. 10141, pp. 92–107. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52240-1_6
16. Heiberg, S., Willemson, J.: Modeling threats of a voting method. In: Design, Development, and Use of Secure Electronic Voting Systems, pp. 128–148. IGI Global (2014)
17. Heiberg, S., Willemson, J.: Verifiable internet voting in Estonia. In: Krimmer, R., Volkamer, M. (eds.) 6th International Conference on Electronic Voting: Verifying the Vote, EVOTE 2014, Lochau / Bregenz, Austria, October 29-31, 2014, pp. 1–8. IEEE (2014). https://doi.org/10.1109/EVOTE.2014.7001135
18. Koutsoulias, I.: Ballot selfies: balancing the right to speak out on political issues and the right to vote free from improper influence and coercion. J. Law Policy **26**, 349–394 (2018)
19. Kremer, S., Ryan, M., Smyth, B.: Election verifiability in electronic voting protocols. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 389–404. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15497-3_24

20. Krips, K., Willemson, J., Värv, S.: Implementing an audio side channel for paper voting. In: Krimmer, R., et al. (eds.) E-Vote-ID 2018. LNCS, vol. 11143, pp. 132–145. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00419-4_9
21. Krips, K., Willemson, J., Värv, S.: Is your vote overheard? A new scalable side-channel attack against paper voting. In: IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019, pp. 621–634. IEEE (2019). https://doi.org/10.1109/EuroSP.2019.00051
22. Moser, F., et al.: A study of mechanisms for end-to-end verifiable online voting. Tech. rep., BSI (2024). https://www.bsi.bund.de/EN/Service-Navi/Publikationen/Studien/Verifiable_Online-Voting/Verifiable_Online-Voting_node.html
23. Pereira, O.: Individual Verifiability and Revoting in the Estonian Internet Voting System. In: Matsuo, S., et al. (eds.) Financial Cryptography and Data Security. FC 2022 International Workshops - CoDecFin, DeFi, Voting, WTSC, Grenada, May 6, 2022, Revised Selected Papers. Lecture Notes in Computer Science, vol. 13412, pp. 315–324. Springer (2022). https://doi.org/10.1007/978-3-031-32415-4_21
24. Toreini, E., Shahandashti, S.F., Hao, F.: Texture to the rescue: practical paper fingerprinting based on texture patterns. ACM Trans. Priv. Secur. **20**(3), 9:1–9:29 (2017). https://doi.org/10.1145/3092816



# Author Index




D. Duenas-Cid et al. (Eds.): E-Vote-ID 2025, LNCS 16028, p. 225, 2026.
https://doi.org/10.1007/978-3-032-05036-6