David Duenas-Cid · Peter Roenne ·
Melanie Volkamer · Pierrick Gaudry ·
Thomas Haines · Isabelle Borucki ·
Leontine Loeber · Arne Koitmäe · Alo Einla (Eds.)

LNCS 17083

Electronic Voting

11th International Joint Conference, E-Vote-ID 2026
Tallinn, Estonia, October 6–9, 2026
Proceedings



Springer

OPEN ACCESS

Lecture Notes in Computer Science **17083**

Founding Editors

Gerhard Goos
Juris Hartmanis

Editorial Board Members

Elisa Bertino, *Purdue University, West Lafayette, IN, USA*
Wen Gao, *Peking University, Beijing, China*
Bernhard Steffen, *TU Dortmund University, Dortmund, Germany*
Moti Yung, *Columbia University, New York, NY, USA*

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 · Pierrick Gaudry ·
Thomas Haines · Isabelle Borucki ·
Leontine Loeber · Arne Koitmäe · Alo Einla
Editors

Electronic Voting

11th International Joint Conference, E-Vote-ID 2026
Tallinn, Estonia, October 6–9, 2026
Proceedings

Springer

Editors
David Duenas-Cid
Kozminski University
Warsaw, Poland

Melanie Volkamer
Karlsruhe Institute of Technology
Karlsruhe, Baden-Württemberg, Germany

Thomas Haines
Australian National University
Melba, ACT, Australia

Leontine Loeber
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands

Alo Einla
Information System Authority of the Republic of Estonia
Tallinn, Estonia

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

Pierrick Gaudry
Université de Lorraine
Nancy, France

Isabelle Borucki
Philipps-University Marburg
Marburg, Hessen, Germany

Arne Koitmäe
State Electoral Office of the Republic of Estonia
Tallinn, Estonia



ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-032-39511-5 ISBN 978-3-032-39512-2 (eBook)
https://doi.org/10.1007/978-3-032-39512-2

© The Editor(s) (if applicable) and The Author(s) 2027. This book is an open access publication.

Open Access This book is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this book or parts of it.
The images or other third party material in this book are included in the book's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the book's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.
This work is subject to copyright. All commercial rights are reserved by the author(s), whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Regarding these commercial rights a non-exclusive license has been granted to the publisher. Unless otherwise expressly agreed by the Publisher in writing, the Publisher does not authorize use or reproduction of all or any part of this work in any manner for the purpose of training, fine-tuning, evaluating, or developing artificial intelligence models, technologies, or systems. In accordance with Article 4(3) of the EU Directive 2019/790 on copyright and related rights in the Digital Single Market, the Publisher expressly reserves all rights with respect to this work under the text and data mining exception of the directive.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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

If disposing of this product, please recycle the paper.

Preface

This volume contains a selection of papers presented at E-Vote-ID 2026, the Eleventh International Joint Conference on Electronic Voting, held on October 6–9, 2026, in Tallinn (Estonia). This represented a major milestone for the conference, since Estonia is a frontrunner in the development and implementation of internet voting. The conference was hosted by the State Electoral Office and the Information System Authority of the Republic of Estonia.

The E-Vote-ID Conference resulted from merging EVOTE and Vote-ID, and dates back more than 20 years to the first E-Vote conference in Austria. Since that conference in 2004, over 2100 experts have attended the venue, including scholars, practitioners, representatives of various authorities, electoral managers, vendors, and PhD students. The conference collects 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, and political aspects, amongst others, and is 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;
- Poster and Demo Session.

E-Vote-ID 2026 received 35 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, 10 papers were accepted for this volume, representing 28% of the submitted proposals. The selected papers cover a wide range of topics connected to electronic voting, internet voting, and electoral technologies, including experiences and revisions of the actual use of E-voting systems and corresponding processes in elections.

We would like to thank the local organisers, Arne Koitmäe, Alo Einla, and their teams, for their excellent collaboration in preparing the conference. 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 2026

David Duenas-Cid
Peter Rønne
Melanie Volkamer
Arne Koitmäe
Alo Einla
Pierrick Gaudry
Thomas Haines
Leontine Loeber
Isabelle Borucki

Organization

General Chairs

David Duenas-Cid	Kozminski University, Poland
Peter Rønne	University of Luxembourg, Luxembourg
Melanie Volkamer	Karlsruhe Institute of Technology, Germany

Program Committee Chairs

Pierrick Gaudry	Centre National de la Recherche Scientifique, France
Thomas Haines	Australian National University, Australia
Leontine Loeber	Vrije Universiteit Amsterdam, The Netherlands
Isabelle Borucki	Philipps University of Marburg, Germany

Steering Committee

Jordi Barrat i Esteve	eVoting Legal Lab, Spain
Jurlind Budurushi	DHBW Karlsruhe, Germany
Micha Germann	University of Bath, UK
Nicole Goodman	Brock University, Canada
Thomas Hofer	Objectif Securité, Switzerland
Robert Krimmer	Dr. Krimmer Consulting OÜ, Estonia
Oksana Kulyk	IT University of Copenhagen, Denmark
Beata Martin-Rozumilowicz	Independent, UK
Adrià Rodriguez	Universitat Pompeu Fabra, Spain
Peter Y. A. Ryan	University of Luxembourg, Luxembourg
Iuliia Spycher	University of Bern, Switzerland

Program Committee

Marta Aranyossy	Corvinus University of Budapest, Hungary
Roberto Araujo	Universidade Federal do Pará, Brazil
Jordi Barrat i Esteve	eVoting Legal Lab, Spain
Michelle Blom	University of Melbourne, Australia

Jeremy Clark	Concordia University, Canada
Cesar A. Collazos	Universidad del Cauca, Colombia
Régis Dandoy	Universidad San Francisco de Quito, Ecuador
Staffan Darnolf	International Foundation for Electoral Systems, USA
Alexandre Debant	Université de Lorraine – CNRS, France
Constantin Catalin	University of Surrey, UK
Alexander Ek	KU Leuven, Belgium
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
Micha Germann	University of Bath, UK
Kristian Gjøsteen	Norwegian University of Science and Technology, Norway
Amanda Glazer	University of Texas at Austin, USA
Nicole Goodman	Brock University, Canada
Rüdiger Grimm	University of Koblenz, Germany
Rolf Haenni	Bern University of Applied Sciences, Switzerland
Thad Hall	Mercer County Elections, USA
Hugo Jonker	Open University of the Netherlands, The Netherlands
Mehmet Sabir Kiraz	De Montfort University, UK
Michael Kirsten	Ludwig-Maximilians-Universität, Germany
Michael McGregor	Toronto Metropolitan University, Canada
Florian Moser	Inria, France
Magdalena Musial-Karg	Adam Mickiewicz University, Poland
Michael Naehrig	IonQ, USA
Hannu Nurmi	University of Turku, Finland
Olivier Pereira	UCLouvain, Belgium
Ismael Peña-López	Universitat Oberta de Catalunya, Spain
Pascal Reisert	University of Stuttgart, Germany
Peter Sasvari	National University of Public Service, Hungary
Uwe Serdült	Ritsumeikan University, Japan
Rodney Smith	University of Sydney, Australia
Mihkel Solvak	University of Tartu, Estonia
Iuliia Spycher	University of Bern, Switzerland
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

Contents

Advances in STV Margin Computation 1
Michelle Blom, Alexander Ek, Peter J. Stuckey, Vanessa Teague, and Damjan Vukcevic

Deniable Vote Updating with Eligibility Verifiability via Trapdoor Chains 18
Janis Erdmanis

It Doesn't Take a Thief: Optical-Scan Voting Systems Fail Even Without Adversaries 35
Aleksander Essex and Philip B. Stark

Risk-Limiting Audits for Parliamentary Majorities 55
Jack Freestone, Dennis Leung, and Damjan Vukcevic

A Simple Protocol for Tracker Based Voting 70
Rosario Giustolisi, Peter Y. A. Ryan, and Maryam Sheikhi Garjan

SoK: Recommendations for Universally Verifiable Electronic Elections and Independent Audits 87
Audhild Høgåsen, Tjerand Silde, Chiara Spadafora, and Amund Fredrik Strømsnes

FiltrumVote: Scalable, Verifiable, and Coercion-Resistant Internet Voting 108
Leonardo Kimura, Dimitri Leskow, Diego F. Aranha, Roberto Araújo, and Marcos Simplicio

"In My Opinion, the Auditing Steps are Sufficient." Trust in Elections, Software, and RLAs 125
Christian Mack, Tobias Hilt, and Melanie Volkamer

École: Privacy in Diversity 141
Florian Moser and Léo Louistisserand

Does Internet Voting Benefit down-Ballot Candidates? A Look at Patterns from 1992 to 2023 in Estonia 158
Mihkel Solvak and Erik S. Herron

Author Index 165

Advances in STV Margin Computation

Michelle Blom[1(✉)], Alexander Ek[2], Peter J. Stuckey[1], Vanessa Teague[3], and Damjan Vukcevic[4]

[1] Department of Data Science and AI, Monash University, Melbourne, Australia
michelle.blom@monash.edu
[2] Department of Computer Science, KU Leuven, Leuven, Belgium
[3] Thinking Cybersecurity Pty Ltd., Melbourne, Australia
[4] Department of Econometrics and Business Statistics, Monash University, Melbourne, Australia

Abstract. *Single transferable vote* (STV) is a multi-winner preferential proportional electoral system. The *margin* is the smallest number of ballots that need to be manipulated to alter the set of winners. If we can compute the margin of an STV election, or a reasonable lower bound on the margin, we can use recent advances in auditing research to conduct a risk-limiting audit of the election's winners. Knowledge of the margin also provides insight into whether uncovered mistakes, or a known error rate in ballot interpretation, could have influenced the outcome. This paper presents substantial improvements on an existing algorithm for computing lower bounds on the margin of an STV election. These improvements allow us to compute higher lower bounds for real STV elections, making mismatch-based risk-limiting audits more practical.

1 Introduction

The *single transferable vote* (STV), also called proportional ranked-choice voting, is an electoral system where voters rank candidates in order of preference and multiple candidates are elected in a manner reflecting voter preferences proportionally. STV is used for national, state, and local elections in Australia, for EU, national, and local elections in Ireland and Malta, and for some elections in Aotearoa/New Zealand, Northern Ireland, Scotland, and the USA.

STV tabulation proceeds in rounds, with each round either electing or eliminating a candidate. In each case, every ballot sitting in the elected or eliminated candidate's tally pile is transferred to the next-highest-ranked eligible candidate on that ballot. When ballots are transferred from an elected candidate's tally, they are reduced in value according to a *transfer value*.

This work was supported by the Australian Research Council (Discovery Project DP220101012, OPTIMA ITTC IC200100009).
A. Ek—Funded by the European Union (ERC, CertiFOX, 101122653). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

© The Author(s) 2027
D. Duenas-Cid et al. (Eds.): E-Vote-ID 2026, LNCS 17083, pp. 1–17, 2027.
https://doi.org/10.1007/978-3-032-39512-2_1

A risk-limiting audit (RLA) is a process designed to efficiently provide affirmative statistical evidence that the reported winners really won, and correct the outcome (with a guaranteed high probability) if they did not win.

The *margin* is the minimum number of ballots that need to be altered—by changing the marked preferences on the ballots—to change who wins.[1] Determining the margin of an election informs us about how close it was, and allows us to conduct an RLA for the election using the mismatch-based method [3]. Mismatch-based audits require only a lower bound, not an exact value, for the margin. However, if this bound is not tight then such audits may require sampling far more ballots than truly necessary. In this paper we present significant improvements in STV margin lower bounding, which should substantially reduce required sample sizes for mismatch-based STV audits.

Blom et al. [2] presented a best-first branch-and-bound algorithm, which we will call BST-19, for computing a lower bound on the margin of an STV election by searching over a tree of possible tabulation *prefixes*. Each *prefix* defines a sequence of seatings and eliminations over a number of tabulation rounds, starting with the first round $r = 1$ and ending at some round $r = n$ which may or may not be the last round of tabulation. This tree captured outcomes that *could* occur if the cast ballots were manipulated. The algorithm used: a method for computing an upper bound on the margin; a linear relaxation of a mixed-integer non-linear program (MINLP) for computing a minimal manipulation to the ballots cast to realise an outcome that starts with a specific prefix; and heuristics for computing a lower bound on the number of ballots that would have to be altered to realise an outcome with a specific prefix. BST-19 is currently the only approach developed for computing lower bounds on the margin of an STV election, while several approaches exist for computing upper bounds [1,4].

In this this paper, we improve BST-19 in several ways:

- *Solving of MINLPs* without resorting to linear relaxation.
- *More sophisticated lower-bounding heuristics* by considering future rounds and more sophisticated reasoning about transfer values.
- *Use of better margin upper bounds* from recent work [1,4].
- *Dominance checks* that allow us to discard prefixes from the search tree if something equivalent has been or is being explored.

Our improved algorithm for STV margins, which we will call STV-26, is able to find exact or near-exact margins for many real-world STV elections. For large 2-seat Australian STV elections, we are able to find substantially higher lower bounds on the margin than previously possible. For STV elections of the size commonly conducted in the USA, and at the local council level in Scotland, we can find margins that would allow us to conduct mismatch-based RLAs [3].

An open source implementation is available in the following GitHub repository: https://github.com/michelleblom/pymarginstv

[1] A more general definition also allows alterations where ballots can be removed or added. Here we only allow alterations that keep the total number of ballots fixed.

2 Background

2.1 Single Transferable Vote

This paper considers the weighted inclusive Gregory method (WIGM) variant of STV. Voters rank candidates in order of preference, from first to last, in either a total order or leaving some candidates unranked, depending on the jurisdiction. Each ballot starts with a value of 1, which is subsequently reduced if used to elect a candidate to a seat. To be seated, a candidate's tally must reach or exceed the *quota* Q (*election threshold*). The *Droop quota* is typically used:

$$Q = \left\lfloor \frac{\#\text{ of validly cast ballots}}{\#\text{ of seats} + 1} \right\rfloor + 1. \tag{1}$$

Each candidate has a *pile* of ballots, and each ballot a *ballot value*. A candidate's *tally* V is the sum of the ballot values in their pile. If a candidate's tally exceeds the quota, they are said to have a *surplus*, and this is equal to the number of votes by which they exceed the quota, $V - Q$.

Initially, each candidate is given all ballots on which they are ranked first. Tabulation proceeds in rounds in which a single candidate is seated or eliminated, until all seats are filled. Ballots in a seated or eliminated candidate's pile are moved to other candidates' piles or discarded. If the number of unfilled seats equals the number of remaining candidates, we seat all remaining candidates.

If no candidate has a quota at the start of a round, we eliminate the candidate with the lowest tally (breaking ties as defined by the jurisdiction), moving all ballots in their pile (at their current value) to the next most preferred eligible candidate on the ballot. If no such candidate exists, the ballot *exhausts*. A *remaining* candidate is one not yet seated or eliminated. An *eligible* candidate is a remaining candidate that does not have a quota at the start of the round.

When seating a candidate, the ballots in their pile continue in the tabulation, but are reduced in value according to their *transfer value* τ, which is the proportion of the tally that is due to the surplus, $\tau = (V - Q)/V$. The new value of each ballot is equal to its current value multiplied by the transfer value. Each of these ballots is then given to the next most preferred eligible candidate on the ballot, or discarded if no such candidate exists. Ballots will skip any remaining candidate that had a quota at the start of the round, even if they have not yet been seated. Where multiple candidates achieve a quota simultaneously, they are seated in order of their tally, highest to smallest.

Example 1. Consider the 3-seat STV election between candidates `A` to `E` in Table 1, with 1230 validly cast ballots and a quota of 308 votes. The first preference tallies of `A` to `E` are 250, 120, 510, 0 and 350 votes. Candidates `C` and `E` have a quota. Candidate `C` has the largest surplus, at 202 votes, and is elected first. Their transfer value is $\tau_1 = 202/510 = 0.396$. The 400 [`C, D`] ballots are each given a weight of 0.396, and a total of 158.4 votes are added to `D`'s tally. The 110 [`C, E, D`] ballots are each given a weight of 0.396, and are also given to candidate `D`, skipping `E` as they already have a quota. Candidate `D` now has

Table 1. The 3-seat STV election from Example 1 between candidates A–E and a quota of $Q = \lfloor 1230/4 \rfloor + 1 = 308$ votes. (a) Number of ballots of each type. (b) Tallies after each counting round, noting when quotas were reached (in bold).

(a)

Ranking	Count
[A]	250
[B, A, C]	120
[C, D]	400
[E]	350
[C, E, D]	110

(b)

Candidate	Round 1	Round 2	Round 3	Round 4
	C elected	E elected	B eliminated	A elected
	$\tau_1 = 0.396$	$\tau_2 = 0.12$		
A	250	250	250	**370**
B	120	120	120	—
C	**510**	—	—	—
D	0	201.96	201.96	201.96
E	**350**	**350**	—	—

a tally of 201.96 votes. Candidate E is then elected. Their transfer value would be $\tau_2 = 42/350 = 0.12$, but all of the ballots in their tally exhaust. In the third round, no candidate has a quota's worth of votes, so the candidate with the smallest tally, B, is eliminated. The 120 [B, A, C] ballots go to A, each retaining their current value of 1. At the start of the fourth round, candidate A has reached a quota, at 370 votes, and is elected to the third and final seat.

2.2 Notation

Definition 1 (STV Election). *An* STV election *is defined as a tuple* $\mathcal{E} = (\mathcal{C}, \mathcal{B}, N, Q, \mathcal{W})$ *where* $\mathcal{C}$ *is the set of candidates up for election,* $\mathcal{B}$ *the multi-set of ballots cast in the election,* N *the number of seats to be filled,* Q *the election quota (Eq. 1), and* $\mathcal{W}$ *the subset of candidates elected to a seat (the winners). Each ballot* $b \in \mathcal{B}$ *is a partial or complete ranking over the candidates* $\mathcal{C}$*.*

Definition 2 (Margin). *The* margin*, or* margin of victory*, for an STV election* $\mathcal{E} = (\mathcal{C}, \mathcal{B}, N, Q, \mathcal{W})$ *is defined as the smallest number of ballot manipulations required to ensure that candidates* $\mathcal{W}' \neq \mathcal{W}$ *are elected to a seat (i.e., at least one candidate in* $\mathcal{W}'$ *must not appear in* $\mathcal{W}$*). A single manipulation changes the ranking on a single ballot* $b \in \mathcal{B}$ *to an alternate ranking.*

Definition 3 (Election order). *Given an STV election* $\mathcal{E} = (\mathcal{C}, \mathcal{B}, N, Q, \mathcal{W})$*, we represent the outcome of* $\mathcal{E}$ *as an* election order π*, where* π *is a sequence of tuples (c, a) with* $c \in \mathcal{C}$*,* $a \in \{0, 1\}$*, and each* $c \in \mathcal{C}$ *appears in at most one tuple. The tuple (c, 1) denotes that candidate c is elected to a seat, while (c, 0) that c has been eliminated. An order* π *is* complete *if it involves the election of* N *candidates, and* partial *if fewer than* N *candidates have been elected in* π*.*

For example, the order π = [(A, 0), (C, 1), (B, 0), (D, 1)] indicates that candidate A is eliminated in the first round of counting, C is next elected to a seat, B is then eliminated, and then D is elected to a seat.

An election $\mathcal{E}$ is tabulated in rounds. We use $\mathcal{S}_\pi$ to denote the set of candidates that have not yet been elected or eliminated by the end of prefix π (i.e., the subset of candidates still standing at the end of π), and $\mathcal{S}_{\pi,r}$ the subset of candidates still standing at the start of round r of π.

2.3 Margin Lower Bound Computation with BST-19

This section describes the overall structure of the STV margin lower bounding algorithm of [2]. We do not change this structure in this paper, but improve some of its components. The key components of the approach are: computation of an *upper bound* on the margin; exploration of a tree of alternate election outcome *prefixes*; heuristics for computing a lower bound on the manipulation required to realise outcomes that start with a specific prefix; and methods for pruning (ignoring) portions of this tree.

```
BST-19(E = (C, B, N, Q, W))
1   rul ← ComputeUpperBound(E) ▷ Sect. 3.1
2   F ← InitialiseFrontier(E)
3   while F ≠ ∅ do
4       n, F ← Remove first node from F
5       F, rul ← Expand(n, F, rul, E)
6   done
7   return rul

Expand(n = (l, π), F, rul, E = (C, B, N, Q, W))
8   for each remaining candidate c do
9       for each action a ∈ {0, 1} do
10          π' ← π || [(c, a)] ▷ Add (c, a) to the end of π
11          l' ← Apply lower bounding heuristics to π' ▷ Sect. 3.2
12          if π' is a complete outcome and l' < rul then
13              rul ← l'
14          else if π' is partial outcome and l' < rul then
15              F ← Insert (l', π') into F
16  F ← Remove all nodes n'' = (l'', π'') where l'' ≥ rul
17  return F, rul
```

Fig. 1. Key components of the BST-19 STV margin lower bounding algorithm. Note that *rul* is a 'running upper limit' on the margin *lower bound* returned by the algorithm, as per Sect. 2.3, and not an upper bound on the margin itself.

Figure 1 shows how these components fit together. BST-19 used several heuristics to compute an upper bound on the STV margin (step 1). We replace these with a heuristic that produces tighter upper bounds (see Sect. 3.1). This upper bound is used to initialise a 'running upper limit on the eventual lower bound', *rul*, that the algorithm will return. BST-19 maintains a frontier of nodes, F, each node $(l, \pi) \in F$ composed of a lower bound l and a prefix π. The prefix is

a partial or complete election order, and l is a lower bound on the manipulation that would be required to realise an outcome that starts with π. The frontier is initialised (step 2) by considering each candidate $c \in \mathcal{C}$ and creating two nodes: one where they are seated and one where they are eliminated in the first round. When a node is created with prefix π, we apply heuristics that compute a lower bound l on the manipulation required to realise an outcome starting with π (see Sect. 3.2). If this lower bound is equal to or higher than our 'running upper limit' (*rul*), we ignore the node, otherwise we add it to our frontier. Nodes on the frontier are sorted in order of their lower bound, smallest to largest.

While our frontier is non-empty, we remove the first node $n = (l, \pi)$ and expand it (steps 4–5). When expanding a node, we create two child nodes for each remaining candidate c (not yet seated or eliminated in π). One child seats c next and the other eliminates them (steps 9–10). If a child seats all of the reported winners of the election, we ignore it, as we are only interested in outcomes that *change* who wins. We apply our lower bounding heuristics (see Sect. 3.2) to compute an l for each child (steps 10–11). If the child is a complete outcome, filling all seats, and $l < rul$, we update *rul* to l (steps 12–13). If the child is a partial outcome and $l < rul$ we add it to our frontier (steps 14–15). Otherwise it is ignored. All nodes $n'' = (l'', \pi'')$ on the frontier where $l'' \geq rul$ are removed (step 16). Expansion updates the frontier and our 'running upper limit' *rul*.

When we have explored or ignored all nodes on the frontier, we return *rul* as the computed lower bound on the election $\mathcal{E}$'s margin (step 7).

In the next section we describe how we have improved the computation of margin upper bounds (Sect. 3.1) and the lower bounding heuristics applied to partial/complete outcomes (Sect. 3.2). We also add a *dominance check* that improves efficiency by completely ignoring some nodes (Sect. 3.3).

3 Improvements

3.1 ConcreteSTV Upper Bounds

Blom et al. [1] and Teague and Conway [4] describe a method of computing upper bounds on STV margins that searches for, and tests, actual manipulations of ballots, looking for those that result in changed outcomes. This heuristic, implemented in ConcreteSTV, finds better (smaller) upper bounds than the heuristics used by BST-19.[2]

3.2 New Lower Bound Heuristics

In step 11 of Fig. 1, BST-19 assigns to each prefix π the largest of several lower bounds: the lower bound assigned to the parent of π (if $|\pi| > 1$), an *elimination* lower bound; a quota lower bound; or a lower bound obtained by solving a relaxation of a MINLP. For a given π, an optimal solution to this MINLP was the smallest manipulation that could be made to the cast ballots to realise an

[2] More details are available in the full version of this paper.

outcome that started with π. The MINLP was relaxed in two ways: by merging sequences of eliminated candidates into a 'super candidate' that was eliminated in a single round; and linearising all non-linearities. We use a similar MINLP in this paper, with the super-candidate relaxation, but retain all non-linearities and use the WIGM surplus transfer rules.[3] Our MINLP now yields a *lower bound* on the manipulation required to realise outcomes starting with π.[4]

The remainder of this section is structured as follows. We first describe how relaxation is used to make the minimal manipulation MINLP tractable (*Super candidate relaxation*). We then outline how we improve the existing lower bounding heuristics used by BST-19. We first explain our new approach for computing tighter lower and upper bounds on candidate tallies in varying contexts (*Minimum and maximum tallies*), and then how these new bounds are used in BST-19's *elimination* and *quota* lower bounds. Finally, we present our new lower bounding heuristic, the *displacement lower bound*, which is used together with the elimination and quota lower bounds, and the minimal manipulation MINLP, to assign lower bounds to prefixes in step 11 of Fig. 1.

Super Candidate Relaxation. Solving the minimal manipulation MINLP becomes intractable when dealing with long election orders. We instead solve the MINLP for a relaxation of a given prefix π, denoted $\tilde{\pi}$, in which some of the sequences of eliminations present in π are grouped or merged. By reducing the total number of candidates in the election, by merging some candidates into a single 'super candidate', the number of model variables is considerably reduced.

Consider an election order $\pi = [(\texttt{A}, 0), (\texttt{C}, 1), (\texttt{B}, 0), (\texttt{E}, 0), (\texttt{F}, 0)\ (\texttt{D}, 1)]$. This order is relaxed by grouping candidates `B` and `E` into one 'super candidate' `BE`, producing $\tilde{\pi} = [(\texttt{A}, 0), (\texttt{C}, 1), (\texttt{BE}, 0), (\texttt{F}, 0), (\texttt{D}, 1)]$. Where (`BE`, 0) appears in the order, it represents candidates `B` and `E` being eliminated in some sequence. Formally, we apply candidate merging to sequences of $n \geq 3$ candidate eliminations $c_1, \ldots, c_{n-1}, c_n$ by grouping candidates c_1 to c_{n-1} into a 'super candidate', leaving c_n out of the merge. When merging eliminated candidates, some constraints in the MINLP, concerned with ensuring those candidates have the lowest tally at the point of their elimination, are removed. We leave c_n out of the merge because merging entire sequences of eliminated candidates produces a relaxation that is too aggressive, resulting in poor lower bounds on the margin.

Minimum and Maximum Tallies. BST-19's lower bounding heuristics rely on computing minimum and maximum possible tallies for candidates in each round of a given prefix. If a ballot might have been distributed as part of a surplus transfer before arriving in a candidate c's pile, overly conservative assumptions were used: it was assumed to have a value of 1 when calculating c's maximum tally, and 0 when computing their minimum tally.

As a prefix does not stipulate the precise round that a seated candidate received their quota, and votes skip candidates that already have a quota during

[3] The original MINLP of [2] used Australian Senate surplus transfer rules.
[4] The formulation of this MINLP is provided in the full version of this paper.

surplus distributions, there is some ambiguity as to which pile a vote belongs to in any given round. For prefix π and round r, we define the *tail* of ballot $b = [x_1, \ldots, x_m]$ as the order of remaining candidates that b *can* (but not necessarily will) be transferred through as the tabulation continues starting with round r.

$$\text{tail}_{\pi,b,r} = [x_i \mid 1 \leq i \leq m \text{ and } x_i \notin \{c_{\pi,1}, \ldots, c_{\pi,r-1}\}] \tag{2}$$

where $c_{\pi,i}$ is the candidate being elected or eliminated in position i of order π.

Example 2. Consider prefix $\pi = [(\mathtt{A}, 1), (\mathtt{B}, 1), (\mathtt{C}, 1)]$. For the ballot $b = [\mathtt{A}, \mathtt{B}, \mathtt{C}]$, and round $r = 2$, $\text{tail}_{\pi,b,r} = [\mathtt{B}, \mathtt{C}]$. For the ballot $b' = [\mathtt{C}, \mathtt{A}, \mathtt{B}]$, $\text{tail}_{\pi,b',r} = [\mathtt{C}, \mathtt{B}]$.

The pile that ballot b belongs to at the start of round r will be one of the candidates in $\text{tail}_{\pi,b,r}$ or the exhausted pile (**ex**). The knowledge of what happens in round r (i.e., a candidate is seated or eliminated) gives extra context as to what pile a ballot b could be in. In particular, the only time piles become ambiguous is when two or more seatings occur in a row in π.

We define $\text{pile}_{\pi,b,r}$ as the set of possible piles a ballot b could be in at the start of round r of prefix π. In the following, $a_{\pi,i}$ is 0 when a candidate is eliminated in round i of π and 1 if a candidate is elected.

$$\text{pile}_{\pi,b,r}\& = \begin{cases} \{\mathbf{ex}\} & \text{if } \text{tail}_{\pi,b,r} = \emptyset \\ \{x_1, \ldots, x_m, \mathbf{ex}\} & \text{if } a_{\pi,r-1} = a_{\pi,r} = 1, \text{ where } \text{tail}_{\pi,b,r} = [x_1, \ldots, x_m] \\ \{x_1\} & \text{otherwise, where } \text{tail}_{\pi,b,r} = [x_1, \ldots, x_m] \end{cases} \tag{3}$$

We can now define which ballots *must* be in a given candidate's pile in a given round, and which ballots *may* be in their pile.

$$\mathcal{B}^{\text{must}}_{\pi,c,r} = \{b \mid b \in \mathcal{B} \text{ where } \{c\} = \text{pile}_{\pi,c,r}\} \tag{4a}$$

$$\mathcal{B}^{\text{maybe}}_{\pi,c,r} = \{b \mid b \in \mathcal{B} \text{ where } c \in \text{pile}_{\pi,c,r}\} \tag{4b}$$

Table 2. Example of how tails and piles evolve in Example 3 for different ballots b and rounds r of a prefix π, where **ex** denotes that a ballot has exhausted.

		prefix π = [(A,0), (B,1), (C,1), (D,0)]			
ballot b		$r = 1$	$r = 2$	$r = 3$	$r = 4$
[A, D]	tail:	[A, D]	[D]	[D]	[D]
	pile:	{A}	{D}	{D}	{D}
[A, C, B]	tail:	[A, C, B]	[C, B]	[C]	∅
	pile:	{A}	{C}	{C}	{**ex**}
[A, B, C, D]	tail:	[A, B, C, D]	[B, C, D]	[C, D]	[D]
	pile:	{A}	{B}	{C, D, **ex**}	{D}

Example 3. Consider the prefix $\pi = [(\texttt{A},0),(\texttt{B},1),(\texttt{C},1),(\texttt{D},0)]$. Table 2 shows how $\text{tail}_{\pi,b,r}$ and $\text{pile}_{\pi,b,r}$ are computed for different ballots b and rounds r in π. For ballot [A, C, B] and round 2, for example, the tail is [C, B] while the ballot can only be in one pile, that of candidate C. The ballot [A, C, B] *must* be in candidate A's pile in round 1, and then in C's pile in rounds 2 and 3. The ballot [A, B, C, D] *must* be in candidate A's pile in round 1, B's pile in round 2, but then *may* be in C's, D's, or the exhausted pile in round 3. Our determination of which pile a ballot could be in at round r of π *does not* consider events at rounds $r' > r$.

We have not yet defined *how much* a ballot b contributes to the pile it is in. As finding the pile of a ballot b at the start of round r of a prefix π is sometimes ambiguous, the value of a ballot b is similarly sometimes ambiguous. We denote $B^{\max}_{\pi,b,r}$ and $B^{\min}_{\pi,b,r}$ as the maximum and minimum possible value (between 0 and 1) of ballot b at the start of round r in prefix π. Whenever a candidate c is seated in a round r of a prefix π, there is an associated transfer value $T_{\pi,c,r}$. To compute the minimum and maximum value of a ballot b after it has passed through one or more surplus transfers, we need to establish lower and upper bounds on the transfer value associated with each of those transfers. Let $T^{\min}_{\pi,c,r}$ and $T^{\max}_{\pi,c,r}$ denote a lower and upper bound, respectively, on the transfer value for the seated candidate c in round r of π (defined in Eq. 7).

$$B^{\max}_{\pi,b,1} = B^{\min}_{\pi,b,1} = 1 \tag{5a}$$

$$B^{\max}_{\pi,b,r} = \begin{cases} B^{\max}_{\pi,b,r-1} \times T^{\max}_{\pi,c,r-1} & \text{if } b \in \mathcal{B}^{\text{must}}_{\pi,c,r-1} \text{ and } a_{\pi,r-1} = 1 \\ B^{\max}_{\pi,b,r-1} & \text{otherwise} \end{cases} \tag{5b}$$

$$B^{\min}_{\pi,b,r} = \begin{cases} B^{\min}_{\pi,b,r-1} \times T^{\min}_{\pi,c,r-1} & \text{if } b \in \mathcal{B}^{\text{maybe}}_{\pi,c,r-1} \text{ and } a_{\pi,r-1} = 1 \\ B^{\min}_{\pi,b,r-1} & \text{otherwise} \end{cases} \tag{5c}$$

When π contains no seatings, both the minimum and maximum value of a ballot in any round of π is 1. We compute minimum and maximum tallies as follows:

$$V^{\max}_{\pi,c,r} = \sum_{b \in \mathcal{B}^{\text{maybe}}_{\pi,c,r}} B^{\max}_{\pi,b,r}, \qquad V^{\min}_{\pi,c,r} = \sum_{b \in \mathcal{B}^{\text{must}}_{\pi,c,r}} B^{\min}_{\pi,b,r} \tag{6}$$

We compute bounds on the transfer value for a candidate c, seated in round r of a prefix π, in Eq. 7.

$$T^{\max}_{\pi,c,r} = \frac{\max\left(Q,\, V^{\max}_{\pi,c,r}\right) - Q}{\max\left(Q,\, V^{\max}_{\pi,c,r}\right)}, \qquad T^{\min}_{\pi,c,r} = \frac{\max\left(Q,\, V^{\min}_{\pi,c,r}\right) - Q}{\max\left(Q,\, V^{\min}_{\pi,c,r}\right)} \tag{7}$$

As we are often computing bounds for prefixes that did not arise in practice, i.e., that do not follow from the cast ballots, candidates may be elected in positions without a quota. We take the max of quota and the actual tally in Eq. 7 to arrive at sensible transfer values in these contexts.

Example 4. Consider the example from Table 1 and now with the prefix $\pi = [(\texttt{C}, 1), (\texttt{E}, 1), (\texttt{A}, 0)]$. At the start of the first round, all ballots sit in the pile of

their highest ranked candidate, and have a value of 1. As there is no ambiguity around the location and value of ballots, $\mathcal{B}^{\text{maybe}}_{\pi,c,r=1} = \mathcal{B}^{\text{must}}_{\pi,c,r=1}$ and $V^{\min}_{\pi,c,r=1} = V^{\max}_{\pi,c,r=1}$ for all candidates c. Consequently, we can compute an exact transfer value for C, i.e., $T^{\min}_{\pi,\mathtt{C},r=1} = T^{\max}_{\pi,\mathtt{C},r=1} = 0.396$. At the start of the second round, candidate E will have a minimum tally of $V^{\min}_{\pi,\mathtt{E},r=2} = 350$ votes and a maximum tally of $V^{\max}_{\pi,\mathtt{E},r=2} = 393.56$. The difference arises as the 110 [C, E, D] votes sitting in C's pile in round 1 may or may not skip over E, when transferred at a value of 0.396 each, depending on when E achieves their quota. We can compute lower and upper bounds on E's transfer value in round 2 as follows.

$$T^{\min}_{\pi,\mathtt{E},2} = \frac{\max(308, 350) - 308}{\max(308, 350)} = 0.12, \quad T^{\max}_{\pi,\mathtt{E},2} = \frac{\max(308, 393.56) - 308}{\max(308, 393.56)} = 0.22.$$

Consider candidate D's tally in $r = 3$ of π. Their minimum tally, $V^{\min}_{\pi,\mathtt{D},r=3}$, is equal to the sum of the minimum values of the ballots that *must* be in their pile in round 3. These are the 400 [C, D] ballots, at 0.396 votes each, and the 110 [C, E, D] ballots, at $0.396 \times 0.12 = 0.04752$ votes each. Thus, $V^{\min}_{\pi,\mathtt{D},r=3} = 163.627$. The maximum tally of D in $r = 3$ of π is equal to the sum of the maximum values of the ballots that *maybe* in their pile in round 3 which, in this case, is the same set of ballots that *must* be in their pile. By Eq. 6, the maximum total value of the 400 [C, D] and 110 [C, E, D] ballots in round 3 is $400 \times 0.396 = 158.4$ and $110 \times 0.396 = 43.56$. Note that the latter set of ballots is not in $\mathcal{B}^{\text{must}}_{\pi,\mathtt{E},r=2}$.

$$V^{\min}_{\pi,\mathtt{D},r=3} = 163.627, \qquad V^{\max}_{\pi,\mathtt{D},r=3} = 158.4 + 43.56 = 201.96$$

The minimum and maximum tallies of A and B in round 3 of π are simpler to compute, as the ballots that *must* and *maybe* in their piles across rounds 1 to 3 are the same, and none of these ballots participate in a surplus transfer.

$$V^{\min}_{\pi,\mathtt{A},r=3} = V^{\max}_{\pi,\mathtt{A},r=3} = 250, \qquad V^{\min}_{\pi,\mathtt{B},r=3} = V^{\max}_{\pi,\mathtt{B},r=3} = 120$$

Elimination Lower Bound. For prefix π, the *elimination lower bound*, l^{elim}_{π}, is a lower bound on the number of ballots we need to change to ensure that each eliminated candidate in π has the smallest tally in the round they are eliminated.

For a candidate $c \in \mathcal{C}$, eliminated in round r of π, we compute c's minimum tally at that point, $V^{\min}_{\pi,c,r}$, as per Eq. 6. We also compute the maximum possible tally of each other *remaining* candidate c' (i.e., that is *still standing*) at the start of round r, according to π, as per Eq. 6. For c to be eliminated in r, we need their minimum tally at this point to be *less* than the maximum tally of all other candidates still standing. Otherwise, we need to take votes away from c to make this so. If c's minimum tally is *greater* than the maximum tally of one of the candidates still standing, then they cannot possibly be eliminated in round r. Thus, we need to change *at least* the following number of votes:

$$l^{\text{elim}}_{\pi,c} = \max_{c' \in \mathcal{S}_{\pi,r} \setminus \{c\}} \left(\frac{V^{\min}_{\pi,c,r} - V^{\max}_{\pi,c',r}}{2} \right)^{+} \tag{8}$$

where $(\cdot)^+$ is the positive part function.[5] For each c vs c' comparison, the change involves giving some of the votes that would reside with c to c'.

This forms an elimination lower bound, $l^{\text{elim}}_{\pi,c}$, with respect to candidate c. The overall elimination lower bound for π is obtained by taking the maximum candidate-based elimination lower bound across all candidates eliminated in π. Let $E_\pi \subset \mathcal{C}$ denote the set of candidates eliminated in order π. This gives:

$$l^{\text{elim}}_{\pi} = \max_{c \in E_\pi} l^{\text{elim}}_{\pi,c} \tag{9}$$

Example 5. Consider prefixes $\pi = [(\texttt{C}, 1), (\texttt{E}, 1), (\texttt{A}, 0)]$ and $\pi' = [(\texttt{C}, 1), (\texttt{B}, 1)]$ for the STV election of Table 1. No candidate in the prefix π' has been eliminated, and so its elimination lower bound is 0. In π, A is eliminated in the third round. In this case, $l^{\text{elim}}_{\pi} = l^{\text{elim}}_{\pi,\texttt{A}}$ by Eq. 9, and

$$l^{\text{elim}}_{\pi,\texttt{A}} = \max_{c' \in \{\texttt{B},\texttt{D}\}} \left(\frac{V^{\min}_{\pi,\texttt{A},r=3} - V^{\max}_{\pi,c',r=3}}{2} \right)^+$$

by Eq. 8. To compute l^{elim}_{π}, we need to compute: $V^{\min}_{\pi,\texttt{A},3}$, $V^{\max}_{\pi,\texttt{B},3}$ and $V^{\max}_{\pi,\texttt{D},3}$.

From Example 4, $V^{\min}_{\pi,\texttt{A},3} = 250$, $V^{\max}_{\pi,\texttt{B},3} = 120$ and $V^{\max}_{\pi,\texttt{D},3} = 201.96$. Consequently, $l^{\text{elim}}_{\pi,\texttt{A}}$ is the maximum of $0.5 \times (250 - 120) = 65$ and $0.5 \times (250 - 201.96) = 24.02$ which is 65 votes. Thus, $l^{\text{elim}}_{\pi} = 65$.

Quota Lower Bound. For a prefix π, its *quota lower bound* considers all the candidates that are seated in π. Consider a candidate c that is seated in round r of π. If the maximum tally of c at that point is less than a quota, then c cannot possibly have been seated and we need to give extra votes to c to make it so. The quota lower bound with respect to candidate c in π, and the overall quota lower bound for π is given by:

$$l^{\text{quota}}_{\pi,c} = \left(Q - V^{\max}_{\pi,c,r}\right)^+, \quad l^{\text{quota}}_{\pi} = \max_{c \in W_\pi} l^{\text{quota}}_{\pi,c} \tag{10}$$

Example 6. Consider the prefix $\pi = [(\texttt{C}, 1), (\texttt{E}, 1), (\texttt{D}, 1)]$ for the STV election of Table 1. Three candidates are elected: C in the first round, E in the second, and D in the third. In this example, $W_\pi = \{\texttt{C}, \texttt{E}, \texttt{D}\}$. In this case,

$$l^{\text{quota}}_{\pi,\texttt{C}} = \left(Q - V^{\max}_{\pi,\texttt{C},1}\right)^+, \quad l^{\text{quota}}_{\pi,\texttt{E}} = \left(Q - V^{\max}_{\pi,\texttt{E},2}\right)^+, \quad l^{\text{quota}}_{\pi,\texttt{D}} = \left(Q - V^{\max}_{\pi,\texttt{D},3}\right)^+$$

$V^{\max}_{\pi,\texttt{C},1}$ is equal to C's first preference tally, 510, while $V^{\max}_{\pi,\texttt{E},2}$ is 393.56 as per Example 4. The prefix considered in Example 4 is that same as π here in rounds 1 and 2, but differs in round 3. To compute the maximum tally of D in round 3, we sum the maximum values of the ballots that *may* be in their pile: the 400 $[\texttt{C}, \texttt{D}]$ and 110 $[\texttt{C}, \texttt{E}, \texttt{D}]$. Note that the latter set of ballots are not in $\mathcal{B}^{\text{must}}_{\pi,\texttt{E},r=2}$. This gives us $V^{\max}_{\pi,\texttt{D},3} = 0.396 \times 400 + 0.396 \times 110 = 201.96$. Given $Q = 308$, we have:

$$l^{\text{quota}}_{\pi,\texttt{C}} = 0, \quad l^{\text{quota}}_{\pi,\texttt{E}} = 0, \quad l^{\text{quota}}_{\pi,\texttt{D}} = (308 - 201.96)^+ = 106.4$$

By Eq. 10, $l^{\text{quota}}_{\pi} = 106.4$, the maximum of $l^{\text{quota}}_{\pi,\texttt{C}}$, $l^{\text{quota}}_{\pi,\texttt{E}}$, and $l^{\text{quota}}_{\pi,\texttt{D}}$.

[5] This is defined as $(a)^+ = \max(a, 0)$, which has value a if $a \geqslant 0$ and value 0 if $a < 0$.

Displacement Lower Bound. The elimination and quota lower bounds consider only the eliminations and seatings present in a prefix π. If, by the end of π, no *new* candidate has been elected, we know that *something* must change in future rounds. Some reported loser must be elected in place of a reported winner.

Consider a prefix π, concluding in round $r-1$, where it is clear that at least one reported loser still standing has to displace one of the reported winners still standing. In this case, we need to ensure that at least one of the reported losers will not be eliminated before one of the reported winners.

We compute the *displacement lower bound* for π, l_π^{disp}, as follows. First, we check whether π already changes our reported outcome by seating a reported loser or eliminating a reported winner. In both cases, l_π^{disp} is zero. We then check whether there is scope to change who is elected in subsequent rounds, beyond π. If the number of unfilled seats equals the number of subsequent rounds, all remaining candidates will be automatically seated, and l_π^{disp} is again zero. We then consider each reported loser c that is still standing (not yet elected or eliminated) at the end of π. We denote the set of such candidates as $\mathcal{L}_\pi$. We compute for each $c \in \mathcal{L}_\pi$, the cheapest way that we could elect c in the future at the expense of a reported winner, $l_{\pi,c}^{\text{disp}}$. The displacement lower bound we assign to π, l_π^{disp}, is the smallest of those computed for each reported loser $c \in \mathcal{L}_\pi$.

$$l_\pi^{\text{disp}} = \min_{c \in \mathcal{L}_\pi} l_{\pi,c}^{\text{disp}} \tag{11}$$

To compute the displacement lower bound for a candidate $c \in \mathcal{L}_\pi$, we compute three values: the cheapest way we can make sure c is not eliminated before some reported winner still standing ($\text{DispCost}_{\pi,c}$); the cheapest way we can ensure c achieves a quota ($\text{QuotaCost}_{\pi,c}$); and the cheapest way we can ensure c outlasts enough candidates to be automatically seated in the final round ($\text{LeftAtEndCost}_{\pi,c}$). The displacement lower bound for c is then:

$$l_{\pi,c}^{\text{disp}} = \max\left\{\text{DispCost}_{\pi,c}, \min\{\text{QuotaCost}_{\pi,c}, \text{LeftAtEndCost}_{\pi,c}\}\right\}. \tag{12}$$

To compute $\text{DispCost}_{\pi,c}$, we consider each reported winner w still standing at the end of π, $w \in \mathcal{W} \cap \mathcal{S}_\pi$, and compute the minimum cost required to displace w with c, $\text{DispCost}_{\pi,c,w}$. $\text{DispCost}_{\pi,c}$ is the minimum of these displacement costs.

$$\text{DispCost}_{\pi,c} \leftarrow \min_{w \in \mathcal{W} \cap \mathcal{S}_\pi} \text{DispCost}_{\pi,c,w} \tag{13}$$

Recall that r is the next round of tabulation after π. We compute the maximum tally c could achieve from the end of π onward, in the context where w is still standing, $V_{\pi,c\prec w,r}^{\max}$ (Eq. 15), and the minimum tally of candidate w at the end of π, $V_{\pi,w,r}^{\min}$ (Eq. 6). A lower bound on the cost of displacing w with c is equal to half the difference between these tallies.

$$\text{DispCost}_{\pi,c,w} \leftarrow \max\left\{0, \frac{1}{2}\left(V_{\pi,w,r}^{\min} - V_{\pi,c\prec w,r}^{\max}\right)\right\} \tag{14}$$

$$V_{\pi,c\prec w,r}^{\max} = \sum_{b \in \mathcal{B}} \begin{cases} B_{\pi,b,r}^{\max} & \text{if } c \prec w \text{ in } \text{tail}_{\pi,b,r} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where $c \prec w$ in a list is true if c appears before w or if only c appears.

For a candidate c to be seated, they must either achieve a quota or must never be eliminated. To achieve a quota, their maximum possible tally from π onward must reach a quota. Let $V^{\max}_{\pi,c\prec *,r}$ denote this maximum tally.

$$V^{\max}_{\pi,c\prec *,r} = \sum_{b\in\mathcal{B}} \begin{cases} B^{\max}_{\pi,b,r} & \text{if } c \in \text{tail}_{\pi,b,r} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

We compute the cheapest way for c to achieve a quota ($\text{QuotaCost}_{\pi,c}$), and then for c to be automatically seated in the final round ($\text{LeftAtEndCost}_{\pi,c}$).

$$\text{QuotaCost}_{\pi,c} \leftarrow \max\{0, Q - V^{\max}_{\pi,c\prec *,r}\} \tag{17}$$

For an election $\mathcal{E}$ with N seats, and prefix π, candidate c needs to not be eliminated before $L_{\mathcal{E},\pi} = |\mathcal{S}_\pi| - (N - |W_\pi|)$ other remaining candidates. We compute and sort the displacement costs between c and each remaining alternate candidate, both reported losers and winners, and take the maximum of the first $L_{\mathcal{E},\pi}$ of these displacement costs to form $\text{LeftAtEndCost}_{\pi,c}$.

Let $\mathcal{D}^{<}_{\pi,c}$ denote the set of minimum costs required to displace each remaining candidate $c' \in \mathcal{S}_\pi \setminus \{c\}$ with c, in ascending order, and d_i the ith cost in $\mathcal{D}^{<}_{\pi,c}$.

$$\mathcal{D}^{<}_{\pi,c} \leftarrow [\text{DispCost}_{\pi,c,c'} \mid c' \in \mathcal{S}_\pi \setminus \{c\}] \quad \text{LeftAtEndCost}_{\pi,c} = d_{L_{\mathcal{E},c}} \tag{18}$$

Example 7. Consider the prefix $\pi = [(\texttt{C}, 1), (\texttt{E}, 1)]$ for the STV election of Table 1. This π reflects the start of the reported outcome in which `C` and `E` are elected. In the reported outcome, `A` is elected in round 4. We know that at some point after π we will need to elect some candidate *other* than `A`. The reported losers still standing at the end of π are `B` and `D`. To compute l^{disp}_{π} we need to compute a displacement lower bound for each of these reported losers, $l^{\text{disp}}_{\pi,\texttt{B}}$ and $l^{\text{disp}}_{\pi,\texttt{D}}$.

$$l^{\text{disp}}_{\pi,\texttt{B}} = \max\left\{\text{DispCost}_{\pi,\texttt{B}}, \min\{\text{QuotaCost}_{\pi,\texttt{B}}, \text{LeftAtEndCost}_{\pi,\texttt{B}}\}\right\}.$$

$$\text{DispCost}_{\pi,\texttt{B}} \leftarrow \min_{w\in\{\texttt{A}\}} \max\left\{0, \frac{1}{2}\left(V^{\min}_{\pi,w,r=3} - V^{\max}_{\pi,\texttt{B}\prec w,r=3}\right)\right\}$$

To compute $\text{DispCost}_{\pi,\texttt{B}}$, we need to compute $V^{\min}_{\pi,\texttt{A},r=3}$ and $V^{\max}_{\pi,\texttt{B}\prec\texttt{A},r=3}$. The only ballots that `A` must have in their pile at the end of π are the 250 [`A`] ballots. The only ballots that mention `B` are the 120 [`B`, `A`, `C`] ballots.

$$V^{\min}_{\pi,\texttt{A},r=3} = 250, \quad V^{\max}_{\pi,\texttt{B}\prec\texttt{A},r=3} = 120, \quad \text{DispCost}_{\pi,\texttt{B}} = \max\{0, \frac{1}{2}(250 - 120)\} = 65$$

A lower bound on the cost required to give `B` a quota at some point after π is:

$$\text{QuotaCost}_{\pi,\texttt{B}} \leftarrow \max\{0, Q - V^{\max}_{\pi,\texttt{B}\prec *,r=3}\} = \max\{0, 308 - 120\} = 188$$

To compute $\text{LeftAtEndCost}_{\pi,\texttt{B}}$, a lower bound on the manipulation required to make sure `B` is not eliminated before $L_{\mathcal{E},\pi} = 3-(3-2) = 2$ other remaining candidates, we need to compute $\text{DispCost}_{\pi,\texttt{B},\texttt{A}} = \max\left\{0, \frac{1}{2}\left(V^{\min}_{\pi,\texttt{A},r=3} - V^{\max}_{\pi,\texttt{B}\prec\texttt{A},r=3}\right)\right\} =$

65 and $\text{DispCost}_{\pi,\text{B,D}} = \max\left\{0,\ \frac{1}{2}\left(V^{\min}_{\pi,\text{D},r=3} - V^{\max}_{\pi,\text{B}\prec\text{D},r=3}\right)\right\} = \max\left\{0,\ \frac{1}{2}\left(163.627 - 120\right)\right\} = 43.627$. If we sort these displacements costs in ascending order and take the 2nd element ($L_{\mathcal{E},\pi} = 2$), we find that $\text{LeftAtEndCost}_{\pi,\text{B}} = 65$ and $l^{\text{disp}}_{\pi,\text{B}} = \max\{65, \min\{188, 65\}\} = 65$. Repeating this process to compute $l^{\text{disp}}_{\pi,\text{D}}$, we find that: $\text{DispCost}_{\pi,\text{D}} = 24.02$, $\text{QuotaCost}_{\pi,\text{D}} = 106.04$, and $\text{LeftAtEndCost}_{\pi,\text{D}} = 24.02$. Consequently, $l^{\text{disp}}_{\pi,\text{D}} = 24.02$. The overall displacement lower bound for π is the minimum of $l^{\text{disp}}_{\pi,\text{B}}$ and $l^{\text{disp}}_{\pi,\text{D}}$, by Eq. 11. Thus, $l^{\text{disp}}_{\pi} = 24.02$.

3.3 New Pruning Methods

We introduce a *dominance rule* to maximise the portion of the alternate-outcome search space STV-26 can ignore. We say that node (l, π) is dominated by node (l', π') if $l' \leq l$ and the prefixes have the same super-candidate relaxed representations, $\tilde{\pi} \equiv \tilde{\pi}'$. When deciding whether to add an (l, π) to our frontier, F, we check whether (l, π) is dominated by another node already in F, or one that we have expanded before. If so, we do not add it to the frontier.

If we have seen an order, π', with a given relaxed structure, $\tilde{\pi}'$, in the past, and we see that structure again in order π, *and* we know the lower bound we attached to π', l', is smaller or equal to the lower bound we have attached to π, l, then we know that the smallest lower bound we could find for any descendent of π' will be less than or equal to the smallest lower bound we could find for any descendent of π. The MINLP that we create when we add a given sequence of events π^* to the end of either π or π' will be the same. Moreover, the contribution of each elimination or election event to the evaluation of the bound is not dependent on the precise order of prior elimination subsequences.

4 Results

We report the performance and margin lower bounds obtained by STV-26 across ten 2-seat Australian Senate elections, four US-based 2-seat STV elections, two Irish STV elections of 3 and 4 seats, and 21 3- or 4-seat Glasgow city county elections from 2007. We treated each Australian Senate election as a WIGM contest for the purposes of this evaluation. Our results are in Table 3.

The 2013–16 Australian Senate, Irish and Scottish contests were used to evaluate BST-19 in [2]. We use BST-19 to denote the results published in that paper, and BST-19* for results obtained using our re-implementation of their algorithm. We must emphasise that the original implementation (BST-19) was designed for a slightly different variant of STV (Australian Senate rules) and not for WIGM. The difference lies in how the transfer value for ballots distributed as part of a winner's surplus is computed. There is no difference in transfer value computation for the first elected candidate across rule sets. The differences arises in any subsequent surplus transfer. The two variants do not result in different outcomes on any of the contests in Table 3, with candidates elected and eliminated in the same order, or substantially different tallies per round (≤ 10

Table 3. For STV elections with N winners, $|\mathcal{C}|$ candidates, $|\mathcal{B}|$ validly cast votes, and quota $\mathcal{Q}$, we report: margin upper bound (**UB**), lower bounds by various methods, and their run times. BST-19 refers to results published by [2], and BST-19* to results obtained by STV-26 without the algorithmic improvements (i.e., a modern implementation of BST-19). The best margin and time between STV-26 and BST-19* are in bold. A '**' indicates that the 10ks limit was reached; and '–' that no result was computed for that method. Elections where we computed the exact margin only with STV-26 are shown in bold.

						Margin LB			Time (sec)	
Election	N	$\|\mathcal{C}\|$	$\|\mathcal{B}\|$	$\mathcal{Q}$	**UB**	BST-19	BST-19*	STV-26	BST-19*	STV-26
Australian Senate										
ACT'13	2	27	246742	82248	13391	32	80	**2155**	**	**
ACT'16	2	22	254767	84923	18835	224	343	**9162**	**	**
ACT'19	2	17	95749	90078	12939	–	1635	**9933**	**	**
ACT'22	2	23	285217	95073	11078	–	85	**319**	**	**
ACT'25	2	14	293474	97825	23121	–	12358	**16722**	**	**
NT'13	2	24	102027	34494	2298	96	212	**2298**	**	**1.12**
NT'16	2	19	102027	34010	11244	3105	3120	**7438**	**	**
NT'19	2	18	37869	35010	15890	–	3563	**9322**	**	**
NT'22	2	17	103617	34540	11412	–	474	**5092**	**	**
NT'25	2	17	106807	35603	13256	–	958	**5870**	**	**
Minneapolis, USA										
MN BET'09	2	7	16727	10696	2098	–	2097	**2098**	1.38	**0.67**
MN BET'13	2	5	23949	16286	6713	–	6712	**6713**	0.62	**0.21**
MN BET'17	2	4	48163	23232	16863	–	16863	16863	0.31	**0.10**
MN BET'21	2	5	42672	31876	2703	–	2703	2703	0.4	**0.17**
2002 Irish General Election										
Dublin N.	4	13	43942	8789	211	189	211	211	**165**	174
Dublin W.	3	9	29988	7498	366	260	366	366	**6**	8
2007 Glasgow City Council										
Linn	4	11	9567	1914	218	114	218	218	287	**82**
Newlands	3	9	8654	2164	88	78	85	85	2	2
Greater P.	4	9	8682	1737	237	147	235	235	67	**11**
Craigton	4	10	11052	2211	75	38	72	72	6	6
Govan	4	11	9560	1913	309	145	309	309	2609	**468**
Pollock–s	3	9	9567	2392	3	3	3	3	**1**	2
Langside	3	8	2334	9334	233	223	233	233	10	**5**
Southside C.	4	9	8738	1748	229	187	224	224	625	**172**
Calton	3	10	5199	1300	376	118	364	364	881	**54**
Anderston	4	9	6900	1381	99	93	99	99	8	**7**
Hillhead	4	10	8984	1797	105	59	105	105	**25**	40
Partick W.	4	9	12744	2549	193	185	193	193	5	**4**
Garscadden	4	10	10160	2033	396	172	396	396	2310	**642**
Drumchapel	4	10	8680	1737	443	253	443	443	3222	**319**
Maryhill	4	8	9901	1981	321	223	321	321	62	**26**
Canal	4	11	8624	1725	126	98	125	125	11	**10**
Springburn	3	10	5410	1353	528	74	505	**528**	**	**441**
East Centre	4	13	9078	1816	139	65	134	134	2027	**488**
Shettleston	4	11	8803	1761	353	136	270	**353**	**	**6170**
Baillieston	4	11	10376	2076	105	65	104	104	8	**7**
North East	4	10	8363	1673	421	251	420	420	813	**119**

votes). We expect that the margins for each contest under Senate and WIGM rules would likely be similar. In fact, WIGM and Senate rules are equivalent for 2-seat STV elections as only one surplus distribution is made. This means that tabulation of the Australian Senate and Minneapolis (MN) contests under WIGM and Senate rules is equivalent. The original results are included to give a sense of progress due to advances in solving technology, modern computational power, and improvements to the algorithm itself.

All experiments were run on a server with 32 virtual CPUs and 128 GB of RAM. A 10,000 s time limit was applied for each run of BST-19* and STV-26. [2] applied a 12-hour time limit when generating their results (BST-19). All runs of BST-19* and STV-26 use parallelisation when expanding nodes on the frontier, expanding up to 30 nodes simultaneously.

5 Concluding Remarks

The improvements we have made to the STV margin lower bounding method of [2] have resulted in substantially tighter lower bounds for large real-world STV elections than previously computable. For smaller contests, including many of the non-Australian cases in our dataset, we are able to compute exact margins or near-exact lower bounds in substantially less time. The lower bounds we compute in these cases are sufficient for use in mismatch-based RLAs [3]. For example, for the NT 2022 election assuming a mismatch rate of 0.1%, a mismatch-based audit would require sampling on average about 55 ballots based on the STV-26 lower bound, but more than 1000 ballots based on the BST-19* one. If the mismatch rate was 1% then these numbers jump to 106 and almost 100,000 respectively. Further work is required to scale the method for use on large Australian Senate contests with hundreds of candidates and 6 to 12 seats.

References

1. Blom, M., Conway, A., Stuckey, P.J., Teague, V.J.: Did that lost ballot box cost me a seat? Computing manipulations of STV elections. In: AAAI **34**, 13235–13240 (2020)
2. Blom, M., Stuckey, P.J., Teague, V.J.: Toward computing the margin of victory in single transferable vote elections. INFORMS J. Comput. **31**(4), 636–653 (2019). arXiv:1703.03511
3. Ek, A., Blom, M., Stark, P.B., Stuckey, P.J., Teague, V.J., Vukcevic, D.: Doing more with less: Mismatch-based risk-limiting audits. In: Financial Cryptography and Data Security. FC 2025. Lecture Notes in Computer Science, vol. 15754, pp. 241–255. Springer (2026). arXiv:2503.16104
4. Teague, V.J., Conway, A.: iVote issues: assessment of potential impacts on the 2021 NSW local government elections. In: E-Vote-ID (2022)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





Deniable Vote Updating with Eligibility Verifiability via Trapdoor Chains

Janis Erdmanis(✉)

Valmiera, Latvia
janiserdmanis@protonmail.org

Abstract. Remote e-voting systems that implement coercion resistance via deniable vote updating require a cleansing phase to discard superseded ballots before tallying. However, achieving a cleansing that is simultaneously publicly verifiable, robust to interrupted ballot chains, and efficient without leaking revoting patterns remains an open challenge. We present a publicly verifiable cleansing protocol that uses trapdoor chains to verifiably force the collector to select a single final ballot for each voter from a shuffled ballot list. As a result, each eligible voter who participates contributes exactly one final ballot without leaking information about superseded ballots. The protocol runs in linear time with a single, publicly verifiable cleansing authority and integrates with both mixnet and homomorphic tallying. The system supports revoting, fake credentials, and credential recovery via the trapdoor-chain mechanism. It further provides individual verifiability under signing key and trapdoor secrecy. This enables revoting-based coercion-resistant systems (such as the Estonian i-voting system) to achieve public cleansing verifiability without requiring trust in internal auditors.

Keywords: e-voting · E2E-V · eligibility verifiability · coercion resistance · deniable vote updating · commitment shuffle

1 Introduction

End-to-end verifiable (E2E-V) voting aims to provide voters and observers with cryptographic evidence that election outcomes reflect the recorded ballots and that only eligible voters cast at most one vote [1]. This property is widely viewed as essential for trustworthy remote electronic voting. However, verifiability creates a fundamental tension with *coercion resistance* [12]: the same mechanisms that let voters and observers verify can also reveal information that allows adversaries to confirm how a voter participated. Voters must therefore be able to update or disguise their choices under coercion without leaving a detectable trace. A related emerging concern is *everlasting privacy* [9]—the fear that future advances in computation may eventually break today's encryption. Although present evidence suggests such capabilities remain distant [6,21], the concern is already cited to justify restricting public verifiability.

© The Author(s) 2027
D. Duenas-Cid et al. (Eds.): E-Vote-ID 2026, LNCS 17083, pp. 18–34, 2027.
https://doi.org/10.1007/978-3-032-39512-2_2

Coercion resistance is commonly realized through *revoting* and *fake credentials*. In revoting schemes, voters can repeatedly update their choices in private, ensuring that any coerced vote can later be replaced without detection. In contrast, fake-credential approaches allow voters to generate ballots that appear valid but are ultimately excluded during tallying [2,10,12,14]. However, fake credentials introduce usability challenges, as voters must distinguish genuine credentials from decoys. Alternative strategies modify the voting process—for example, by discounting duplicate votes [5] or allowing voters to override remote ballots with paper ballots cast at polling stations [8,13]. Coercion-resistant systems must also address *forced abstention*, where an adversary attempts to ensure that a voter does not participate, for instance by verifying non-participation on a public bulletin board.

Beyond coercion, remote electronic voting systems must also prevent *impersonation*—unauthorized parties casting ballots on behalf of legitimate voters when authentication mechanisms are weak or credentials are compromised [18]. This requirement reinforces the need for strong eligibility verifiability, for which systems based on auditable hardware credentials—especially smartcards that digitally sign ballots—are widely considered the gold standard [19]. However, in revoting-based systems, this creates a fundamental tension during the *cleansing phase*, where a single ballot must be selected per voter without revealing which ballots were retained or discarded. Otherwise, revoting behavior becomes observable, undermining deniability. Achieving public eligibility verifiability without compromising coercion resistance remains a fundamental open problem in remote electronic voting.

1.1 Our Contribution

We introduce a novel cleansing protocol that ensures each eligible voter contributes at most one final ballot to the tally, while revealing no information about revoting behavior or discarded ballots. The protocol enables observers to detect ballot box stuffing and vote suppression without relying on trusted auditors or privileged access.

At a high level, ballots include commitments to voter public keys and to vote-update trapdoors. In the cleansing phase, the collector shuffles the ballots and matches the blinded public keys of the selected ones against a shuffled, reblinded registration roll, which ensures eligibility. For each dropped ballot, the collector proves knowledge of the committed trapdoor, derived from a fresher ballot of the same voter through a smartcard-seeded trapdoor chain. This mechanism also enables fake credentials and supports credential recovery. The cleansing phase runs in linear time and relies on a single publicly verifiable cleansing authority, while remaining compatible with both mixnet and homomorphic tallying. The protocol also supports practical everlasting privacy and individual verifiability, provided the smartcard secrets remain uncompromised.

Beyond the core mechanism, we also address forced abstention as a coercion strategy. We show how relay networks can provide alternative submission channels and independently verify ballot eligibility, enabling misbehavior to be

detected and attributed. These properties reduce a coercer's remaining strategy to physically preventing participation, for example by separating the voter from their smartcard.

1.2 Related Work

Remote e-voting systems typically compute the final tally using either verifiable reencryption mixnets [7,22,23], in which talliers successively shuffle and reencrypt ElGamal ciphertexts with zero-knowledge proofs of correct permutation, or homomorphic tallying [3], which aggregates encrypted votes and reveals the decrypted tally. Recent advances revisit homomorphic tallying with SNARK-based proofs of the result function [11], enabling publicly tally-hiding for complex ballot types at the cost of a trusted setup phase. These tallying approaches can be combined with a prior cleansing phase that ensures at most one vote in the final tally per voter.

JCJ and Encrypted Credential Schemes. The most influential coercion-resistant construction targeting distributed trust is JCJ (Juels–Catalano–Jakobsson) [12], which underlies many modern coercion-resistant voting protocols [2,18]. JCJ publishes ElGamal-encrypted credentials on a public registration roll; voters submit encrypted votes together with encrypted credentials and zero-knowledge proofs of correct formation. Submitted credential–vote pairs are shuffled and cleansed using plaintext equivalence tests that match submitted credentials against the registration roll.

This design introduces privacy, verifiability, and efficiency challenges that have motivated substantial follow-up work. Cortier et al. [2] show that JCJ's cleansing leaks per-credential revote counts, enabling the "1009 attack"; their CHide variant closes this leak with MPC-based cleansing that reveals only the total number of ballots removed while achieving quasi-linear tallying complexity. Additional proposals [18] augment credentials with proofs of logarithm knowledge to prevent colluding cleansing authorities from decrypting registration credentials and impersonating voters. However, these improvements often come at the cost of increased computational or protocol complexity and are not easily combined without sacrificing efficiency. Moreover, the anonymous channel assumption still permits ballot box flooding attacks that can disrupt tallying.

Ledger-Row Schemes. KTV-Helios [14] and DeVoS [15] represent a class of systems in which each voter is associated with a dedicated ledger row, where their ballots are hidden among dummy ballots posted by trustees using disjunctive zero-knowledge proofs. This mechanism prevents coercers from detecting revoting behaviour while also mitigating ballot-box flooding.

In KTV-Helios, the final vote is computed as the product of all ciphertexts in a voter's ledger row. However, this approach is incompatible with homomorphic tallying techniques and requires an uninterrupted chain of recorded ballots for the final vote to remain valid. DeVoS avoids cumulative updates by allowing each

new ballot to supersede the previous one, but at the cost of a two-round-trip casting protocol and a submission phase divided into micro-phases. This inflates the audit trail and makes vote casting sensitive to micro-phase boundaries, requiring late-arriving ballots to be regenerated.

Estonia's Revoting System. Estonia's internet voting system [10,16,20], operational since 2005, provides the closest practical precedent for our construction. Both systems rely on smartcard-based digital identity, revoting for coercion resistance, and a single authority performing ballot cleansing. Voters may revote an unlimited number of times, with receipt-freeness following from update deniability. To address a coercer who observes the voter at the last moment, remote voting closes two hours before polling stations, allowing voters to cast superseding paper ballots [13].

A limitation of Estonia's system is that cleansing is not publicly verifiable. To ensure that a malicious cleansing authority has not dropped, added, or manipulated ballots, internal audits with privileged access to pre-mix data are employed. However, a compromised auditor can observe revoting patterns and thereby violate deniable vote updating, creating a fundamental tension between auditing and coercion resistance. Our construction resolves this tension by making the cleansing phase publicly verifiable while revealing no information about superseded ballots beyond the final tally, thereby eliminating the need for privileged auditors and strengthening coercion resistance.

2 System Overview

This section presents the system architecture and the trust assumptions underlying our security claims. The protocol involves the following principals:

- **Smartcard** $\mathcal{S}$: A tamper-resistant device held by the voter that generates trapdoor commitments and produces proof-of-knowledge signatures for ballots.
- **Voter** $\mathcal{V}$: The voter together with their chosen voting client, responsible for vote encryption, interacting with the smartcard, and submitting the constructed ballot to the collector.
- **Registrar** $\mathcal{R}$: Manages voter registration and publishes the registration roll listing each eligible voter's identity and corresponding public key.
- **Collector** $\mathcal{C}$: Collects ballots, verifies their validity, records the public components on the bulletin board, and performs the verifiable cleansing phase.
- **Tallier** $\mathcal{T}$: A distributed entity that produces the tally from ballots selected in the cleansing phase via homomorphic or mixnet-based tallying.
- **Bulletin Board** $\mathcal{BB}$: An append-only public ledger on which all ballot public components, cleansing and tallying transcripts, and supporting proofs are recorded.

Beyond these principals, an administrator coordinates the setup phase, which falls outside the protocol itself. Any observer may act as an **auditor** by verifying

Table 1. Security properties and trust assumptions. Security properties follow the definitions of [1]. Each row lists the minimal trust requirements for the corresponding property. Where multiple rows are given for the same property, each row represents an alternative set of trust assumptions under which the property can be achieved.

Property	Smartcard	Voter	Registrar	Collector	Tallier	Bulletin Board
Eligibility Verifiability	●	○	●	○	○	○
Universal Verifiability	○	○	○	●	○	○
	●	○	○	○	○	○
Individual Verifiability	●	●	○	○	○	○
Vote Anonymity	○	●	○	●	○	○
	○	●	○	○	T(●)	○
Coercion Resistance	●	○	○	●	○	○
Participation Privacy	●	○	○	●	○	○

Notation: ●– required to be honest; ○– may be corrupt; T(·) – threshold trust.

the public evidence on the bulletin board. A **relay network** may also emerge spontaneously when the collector is unavailable, providing an alternative ballot submission channel while holding the collector accountable. The collector nonetheless remains a liveness dependency for cleansing, as it alone holds the private openings; its failure can stall the tally but not enable undetected manipulation.

Table 1 summarizes the trust assumptions underlying our security claims, with each row indicating which parties are required to be honest to guarantee the corresponding property. Compared to the Estonian system, *universal verifiability*[1] can also be achieved even if the collector is corrupted, provided that the smartcard remains honest—an assumption already required for eligibility verifiability. *Individual verifiability* requires trusting the smartcard, as compromised secrets could be used to modify or invalidate a vote in a deniable manner. It also assumes that the voter performs verification on a separate device and therefore the voter is honest although the casting device does not need to be.

Vote Anonymity that prevents a voter from being linked to their final counted vote is achieved either by the collector performing a shuffle, which additionally provides everlasting privacy (first row), or by the talliers performing either mixnet-based tallying or homomorphic tallying, where anonymity relies on a threshold of the talliers remaining honest (second row). Since the voter necessarily reveals their selection to the casting device, a compromised device can leak the vote—directly or kleptographically through its choice of randomness—

[1] Eligibility and universal verifiability additionally assume the smartcard is not used without the voter's authorization, whether surrendered voluntarily or invoked covertly; this threat and its mitigations are discussed in [16].

so vote anonymity requires trusting the device that cast the final counted vote. In contrast, *coercion resistance* and *participation privacy* do not require trusting the voter's device: since the voter may recast from another device, evidence from any single compromised device—an observed vote, abstention, or forced participation superseded by a null vote—remains inconclusive to the coercer[2].

System Timeline. The protocol proceeds in three phases. In the *setup phase*, the registrar issues smartcards and publishes the registration roll. In the *voting phase*, voters cast ballots using their smartcards; after submission, each voter may independently verify that their ballot appears on the bulletin board and is cast as intended. In the *tallying phase*, the collector performs the cleansing procedure and the tallier produces the final tally, both publishing their evidence on the bulletin board for subsequent public audit.

3 Preliminaries

Notation. Let $\mathbb{Z}_q$ be the integers modulo q, $\mathbb{G}_q$ a group of order q in which the discrete logarithm problem is hard, and $\mathbb{S}_n$ the symmetric group. We fix generators $(\mathsf{h}, \mathsf{g}) \in \mathbb{G}_q^2$ such that $\log_{\mathsf{g}} \mathsf{h}$ is unknown. Commitments to $y \in \mathbb{G}_q$ are $\mathrm{Com}(y) = \mathsf{h}^{\xi_y} y$ for $\xi_y \in_R \mathbb{Z}_q$. ElGamal encryption of $m \in \mathbb{Z}_q$ under $\mathsf{pk} = \mathsf{g}^x$ is $\mathrm{Enc}_{\mathsf{pk}}(m) = (\mathsf{g}^r, \mathsf{pk}^r \mathsf{g}^m)$ for $r \in_R \mathbb{Z}_q$; reencryption is $\mathrm{ReEnc}_{\mathsf{pk}}((a, b)) = (a\mathsf{g}^r, b\mathsf{pk}^r)$ and decryption $\mathrm{Dec}_x((a, b)) = b/a^x$. We model $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$ as a random oracle.

We also introduce an encryption scheme for field elements based on a Diffie–Hellman key encapsulation mechanism. For $t \in \mathbb{Z}_q$, encryption under the public key $\mathsf{pk} = \mathsf{g}^x$ is defined as $\mathrm{Enc}^\star_{\mathsf{pk}}(t; r) = (\mathsf{g}^r,\ t + \mathcal{H}(\mathsf{pk}^r) \pmod q)$, where $r \in_R \mathbb{Z}_q$. Given a ciphertext (a, s), decryption computes the shared Diffie–Hellman secret a^x and outputs $\mathrm{Dec}^\star_x((a, s)) = s - \mathcal{H}(a^x) \pmod q$. Correctness follows since $a = \mathsf{g}^r$ implies $a^x = (\mathsf{g}^r)^x = \mathsf{pk}^r$, and therefore $\mathrm{Dec}^\star_x(\mathrm{Enc}^\star_{\mathsf{g}^x}(t; r)) = t$.

A relation $\mathcal{R} \subseteq \mathbb{G}_q^* \times \mathbb{Z}_q^*$ associates a vector of public group elements (statement) $\mathbf{s} \in \mathbb{G}_q^*$ with a vector of witnesses $\mathbf{w} \in \mathbb{Z}_q^*$. We write $\pi = \mathrm{PoK}\{\mathbf{w} : (\mathbf{s}, \mathbf{w}) \in \mathcal{R}\}$ to denote a non-interactive proof of knowledge of witnesses $\mathbf{w}$ satisfying the relation with respect to statement $\mathbf{s}$, and $\mathrm{Verify}_{\mathcal{R}}(\mathbf{s}, \pi)$ for verification (e.g., $\mathrm{PoK}\{x : y = \mathsf{g}^x\}$). Witnesses generated internally during proof construction and never leaving the prover's trust boundary (e.g., blinding or reencryption factors) are treated as implicit prover randomness and omitted from the relation and witness list for clarity. Auxiliary data is written $\mathrm{PoK}^{(A;B)}\{\cdot\}$, where A and $\mathcal{H}(B)$ are included in the Fiat–Shamir challenge $\mathcal{H}(\mathbf{s}, A, \mathcal{H}(B), \ldots)$; A is public, while only $\mathcal{H}(B)$ is revealed (e.g., in π), so B remains hidden but bound to the proof. Verification is denoted $\mathrm{Verify}^{(A;B)}_{\mathcal{R}}(\mathbf{s}, \pi)$.

[2] A realistic example is a workplace where employees are provided with company-managed laptops or mobile phones that may be subject to monitoring. A voter wishing to oppose the coercer's instructions may cast the requested vote using the monitored device and subsequently recast their intended vote using a personal device.

4 Protocol Specification

Design Rationale. Our construction reverses the burden of proof in cleansing: rather than proving the selected ballots correct, the collector proves that every *discarded* ballot was superseded, by opening its trapdoor commitment. The smartcard makes this possible—and unavoidable—by stepping backward along a trapdoor chain: each ballot commits to a trapdoor t while handing the collector, encrypted, the value $\mathcal{H}(t)$—precisely the trapdoor committed in the voter's previous ballot. Each ballot thus opens the one it supersedes while the newest stays one preimage out of reach—a voter committing t_5, t_4, t_3 lets the collector open the first two but not t_3—forcing every final ballot into the selection. Eligibility is established by first shuffling accepted ballots and the registration roll independently, and then proving that every selected ballot's public-key commitment matches a reblinded registration-roll entry. Since each reblinded commitment is unique, this simultaneously proves that every selected ballot belongs to a registered voter and that at most one final ballot is selected per voter, without linking ballots to voters.

4.1 Setup Phase

The registrar issues a smartcard to each eligible voter and publishes the registration roll listing each voter's identity and corresponding public key $((\mathsf{Id}_i, \mathsf{pk}_i))_i$, upon which all parties agree. Each smartcard contains a trapdoor seed $\mathsf{seed}_i \in_R \mathbb{Z}_q$ from which the vote-update trapdoors are derived by iterated hashing, forming the voter's trapdoor chain. The collector generates a secret key $y \in_R \mathbb{Z}_q$ and publishes the corresponding public key $\mathsf{pk}_\mathcal{C} = \mathsf{g}^y$, under which smartcards encrypt trapdoors. The tallier generates a secret key and publishes a public key $\mathsf{pk}_\mathcal{T}$ under which voters encrypt their votes (Fig. 1).

4.2 Voting Phase[3]

Let N denote the number of voting options. The voter selects a voting option and encodes it as an element $v \in \mathbb{Z}_N$. The voter then samples a randomization factor $r \in_R \mathbb{Z}_q$ and commitment blinding factors $(\alpha, \beta) \in_R \mathbb{Z}_q^2$, computes the ElGamal ciphertext $V = \mathrm{Enc}_{\mathsf{pk}_\mathcal{T}}(v; r)$, and sends (α, β, V) to the smartcard.

The smartcard computes a vote-update trapdoor $t_n = \mathcal{H}^{(n)}(\mathsf{seed})$, and encrypts the previous trapdoor for the collector as $T = \mathrm{Enc}^\star_{\mathsf{pk}_\mathcal{C}}(t_{n+1}) = \mathrm{Enc}^\star_{\mathsf{pk}_\mathcal{C}}(\mathcal{H}(t_n))$. It then constructs the credential commitment $E = \mathsf{h}^\beta \mathsf{pk}$ and the trapdoor commitment $C = \mathsf{h}^\alpha \mathsf{g}^{t_n}$. To bind these values it produces the proof

$$\pi_E = \mathrm{PoK}^{(C,V;\alpha,T)}\{(\beta, x) : E = \mathsf{h}^\beta \mathsf{g}^x\}. \tag{1}$$

The smartcard returns (C, E, T, π_E) to the voter.

[3] Voter Indices Are Suppressed Throughout This Subsection, as Each Voter's Interaction with the Smartcard and Collector Is Independent.

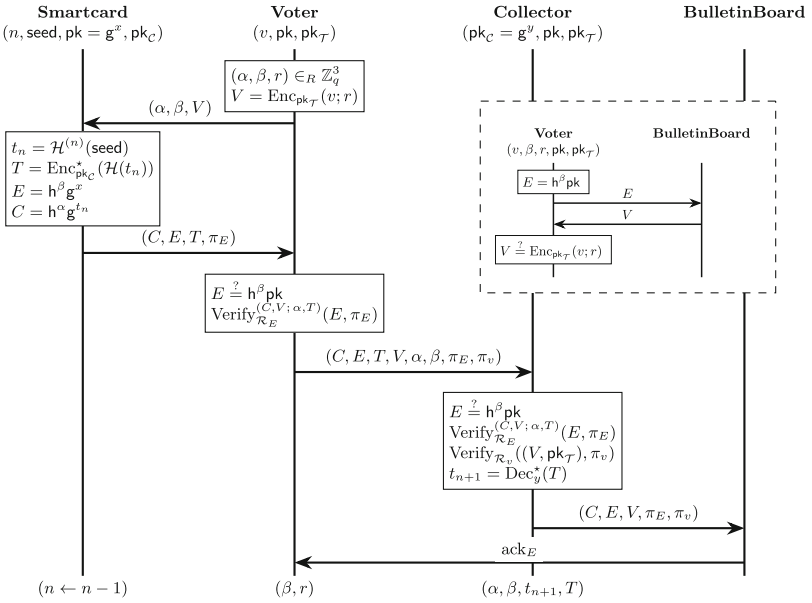


Fig. 1. The vote casting protocol with individual verifiability (inset). The protocol starts with the voter selecting a choice $v \in \mathbb{Z}_N$ and forming its encryption, which is sent to the smartcard for signing. The voter checks the validity of the signature locally before forwarding the ballot to the collector. If the ballot is valid, the collector records the public part on the bulletin board and stores $(\alpha, \beta, t_{n+1}, T)$ required for the cleansing protocol. Once the voting protocol ends, the voter retains (β, r) with which it can verify its ballot from another device using the verification protocol shown in the inset.

The voter verifies that $E \stackrel{?}{=} \mathsf{h}^\beta \mathsf{pk}$ and checks $\mathrm{Verify}_{\mathcal{R}_E}^{(C,V;\alpha,T)}(E, \pi_E)$. If all checks succeed, the voter submits the ballot to the collector:

$$\mathsf{ballot} = (C, E, T, V, \alpha, \beta, \pi_E, \pi_v) \tag{2}$$

Here π_v is a proof constructed by the voter demonstrating knowledge of the ciphertext randomization (to prevent the Pfitzmann attack [17]), or a range proof when homomorphic tallying is employed [3] or when forced abstention through vote randomization is a concern.

The collector repeats the verification steps of the ballot and checks the vote encryption $\mathrm{Verify}_{\mathcal{R}_v}(V, \pi_v)$. If all checks succeed, it performs trapdoor decryption $t_{n+1} = \mathrm{Dec}^\star_y(T)$, stores the private openings and trapdoor encryption $(\alpha, \beta, t_{n+1}, T)$, and publishes the public ballot components (C, E, V, π_E, π_v) on the bulletin board. Since π_E binds $\mathcal{H}((\alpha, T))$ into the Fiat–Shamir challenge, a substituted trapdoor encryption invalidates the ballot at submission; hence a trapdoor encryption T inconsistent with the trapdoor commitment C can only originate from the smartcard, to which the fault is attributed when detected during cleansing (Sect. 4.3). The bulletin board then returns an acknowledgment

ack_E to the voter, certifying that the blinded public key E (and consequently the associated public ballot components) has been recorded[4].

Relay-Verifiable Ballots. If the voter cannot reach the collector directly, the ballot may instead be submitted through a relay. A relay receiving the ballot performs the same verification steps as the collector, allowing it to confirm the ballot's eligibility before attempting to record it on the bulletin board. If submission fails, the relay may forward the ballot to other relays, each of which can independently verify and attempt submission. This relay network makes the collector accountable and ensures that voters can cast their ballots even if the collector becomes unreachable.

Individual Verifiability. The voter verifies their ballot from another device as follows. They reconstruct the credential commitment $E = \mathsf{h}^{\beta}\mathsf{pk}$ from their public key pk and blinding factor β, and use it to locate their ballot on the bulletin board. They then check, using r and v, that $V \stackrel{?}{=} \text{Enc}_{\mathsf{pk}_T}(v; r)$.

4.3 Tallying Phase

The tallying phase begins with cleansing, which is performed entirely by the collector. The collector first applies a shuffle to the list of tuples $((C_i, E_i, V_i))_i$ using a secret permutation $\psi \in_R \mathbb{S}_I$. It then publishes the shuffled list $((C'_i, E'_i, V'_i))_i$ together with a zero-knowledge proof of the correctness of the shuffle:

$$\pi_{CEV} = \text{PoK}\{\psi : C'_i = \text{Com}(C_{\psi(i)}) \wedge E'_i = \text{Com}(E_{\psi(i)}) \wedge V'_i = \text{ReEnc}(V_{\psi(i)})\} \tag{3}$$

where the statement holds for all recorded ballots $i \in [I]$ and the blinding factors are chosen so that every E'_i is unique.

From the shuffle output, the collector selects the index set $L \subseteq [I]$ corresponding to each voter's last cast ballot. To verify eligibility of each selected index $i \in L$, a new shuffle is formed from the registration roll public keys $(\mathsf{pk}_k)_k$ via a permutation $\phi \in_R \mathbb{S}_K$. The shuffle produces blinded public keys $(E''_k)_k$ with blinding factors chosen so that $\{E'_i\}_{i \in L} \subseteq \{E''_k\}_k$, and is supported by a zero-knowledge proof:

$$\pi_{\text{Elig}} = \text{PoK}\{(\phi, \{\beta_{\psi(i)}\}_{i \in L}) : E''_k = \text{Com}(\mathsf{pk}_{\phi(k)}) \wedge \{E'_i\}_{i \in L} \subseteq \{E''_k\}_k\} \tag{4}$$

where the statement holds for all registered public keys $k \in [K]$.

Finally, to demonstrate that the selected indices L correspond to each voter's last cast ballot and that no eligible ballot has been omitted, a proof of knowledge is constructed for the trapdoor associated with each dropped ballot:

$$\pi_{\text{Drop}} = \bigwedge_{i \notin L} \text{PoK}\{(\alpha'_i, t_i) : C'_i = \mathsf{h}^{\alpha'_i}\mathsf{g}^{t_i}\} \tag{5}$$

[4] The acknowledgment may simply consist of the bulletin board's digital signature on E, or of a commitment to a batch of published entries, such as a digitally signed Merkle tree root containing E as one of its leaves.

Together, these proofs form a publicly verifiable transcript of the cleansing phase published on the bulletin board:

$$\mathsf{tr} = \big(((C_i, E_i, V_i, \pi_{E_i}, \pi_{v_i}))_i \,,\, ((C'_i, E'_i, V'_i))_i \,,\, (E''_k)_k \,,\, \pi_{CEV},\, \pi_{\text{Elig}},\, \pi_{\text{Drop}} \big)$$

which anyone can verify using the registration roll $((\mathsf{Id}_k, \mathsf{pk}_k))_k$. The selected vote ciphertexts $\{V'_i\}_{i \in L}$ are then passed through a reencryption mixnet [22] or homomorphically tallied [3].

Trapdoor Reduction. The drop proof π_{Drop} requires the collector to know a full opening (α'_i, t_i) of each shuffled commitment C'_i with $i \notin L$. The blinding factor α'_i is always known to the collector, being the sum of the α_i received at submission and its own reblinding factor from the shuffle. The trapdoor component t_i, however, must be derived; the collector obtains it by finding the trapdoor of lowest index in each voter's trapdoor chain among the recorded ballots.

For a given trapdoor t_*, define the set of reachable trapdoors

$$F(t_*) = \{\mathcal{H}^{(n)}(t_*) : n \in [M]\}, \tag{6}$$

where M is an upper bound on the trapdoor-chain length that the smartcard is allowed to use. For each public key, the collector considers the trapdoor values decrypted from that key's ballots and selects the value t_* of lowest index, i.e., the unique decrypted value such that every other decrypted value lies in $F(t_*)$. Such a value exists because revotes with the genuine credential step backward through the chain: each new ballot commits the preimage of the previously committed trapdoor, and its encrypted trapdoor decrypts to that previously committed value, so the decrypted values all lie on a single chain, one step ahead of the corresponding commitments. Every superseded ballot therefore has its trapdoor $t_j \in \{t_*\} \cup F(t_*)$, and its commitment $C_j = \mathsf{h}^{\alpha_j} \mathsf{g}^{t_j}$ can be opened. The remaining commitment $C_i = \mathsf{h}^{\alpha_i} \mathsf{g}^{t_i}$ commits to the preimage t_i of t_*, i.e., $\mathcal{H}(t_i) = t_*$, which the collector cannot obtain without inverting $\mathcal{H}$; it is therefore forced to select ballot i as the final one.

If trapdoor reduction fails, the collector publishes all $(\alpha_i, \beta_i, t_i, T_i, \pi_{\text{Dec}^\star_i})$ associated with the given public key, where $\pi_{\text{Dec}^\star_i} = \text{PoK}\{y : t_i = \text{Dec}^\star_y(T_i)\}$ proves that t_i was correctly derived from the encrypted trapdoor T_i. This provides verifiable evidence that the corresponding voter's smartcard has been compromised, allowing the vendor to be held accountable.

Fake Credentials. The scheme supports fake credentials implemented as smartcard PIN codes by assigning each a higher trapdoor-chain index than the genuine credential. Each PIN behaves as a genuine credential until a higher-precedence PIN is used, after which it behaves as a fake credential relative to that PIN while remaining genuine with respect to lower-precedence PINs. Consequently, this provides a weaker form of coercion resistance than [2,12,14]; however, it still defends against coercion strategies in which a coercer arrives at the last moment to observe the voter casting the final ballot.

The genuine credential traverses the trapdoor chain backward ($n \leftarrow n-1$), making each new trapdoor commitment $\mathsf{g}^{t_i} = C_i/\mathsf{h}^{\alpha}$ unopenable and ensuring that the last genuine ballot survives. Fake credentials traverse the chain forward ($n \leftarrow n+1$), making their commitments openable once a higher-precedence credential has been used. This permits an arbitrary number of revotes, limited only by the trapdoor-chain length.

Interleaving credentials requires care. Trapdoors must never be reused, as losing a ballot with a lower-index trapdoor could leave multiple commitments unopenable, preventing the collector from identifying a unique surviving ballot. Since ballot delivery is not guaranteed, the smartcard advances a fake credential to the highest unused chain position after each genuine invocation. The resulting ballot remains openable from the preceding fake ballot's encrypted trapdoor, preserving first-ballot precedence among fake credentials.

A useful side effect of this design is that it enables credential recovery without exposing the voter to additional coercion risks. A recovery credential can be printed on paper and used if the voter forgets their genuine credential. The genuine credential remains secret and can still be used to evade coercion if the coercer demands the printed credential. This approach also improves usability: since remembering multiple PIN codes can be error-prone, the voter need only remember one credential, from which fake credentials can be derived whenever needed.

5 Security Analysis

This section provides an overview of the security mechanisms that ensure the properties outlined in Table 1 under the specified trust assumptions.

Eligibility Verifiability. Eligibility verifiability ensures that every selected ballot for tallying is authorized by an eligible voter who contributes at most one ballot. Assuming the smartcard's signing key has not been compromised and the smartcard has not been used without the voter's authorization [16], the proof proceeds as follows. The proof π_{E_i}, being a proof-of-knowledge signature over the ballot, establishes that the submitter knows the private key underlying $E_i = \mathrm{Com}(\mathsf{pk}'_i)$ and binds this credential to the vote encryption V_i. The shuffle proof π_{CEV} ensures that the shuffled commitments E'_i open to the same public keys as the originals. The registration-roll shuffle $\mathcal{R}_{\mathrm{Elig}}$ produces commitments $E''_k = \mathrm{Com}(\mathsf{pk}_{\phi(k)})$ to the eligible keys. The verified subset relation $\{E'_i\}_{i\in L} \subseteq \{E''_k\}_k$ then forces each selected ballot's credential to be an eligible public key. Since the registration roll contains one key per voter, at most one selected commitment E'_i maps to each registration key, giving at most one selected ballot per voter for tallying.

Universal Verifiability. Universal verifiability ensures that the final tally consists only of accepted ballots that have not been manipulated or dropped, with

each voter contributing at most one valid vote. Here we focus on cleansing verifiability with corrupt collector, as tallying verifiability of the selected ballots is well understood [3,7,11,22]. We show that the selected set L contains the final ballot of every voter, as determined by the smartcard's trapdoor chain, and that no eligible voter's final ballot is omitted, assuming the smartcard neither leaks the trapdoor seed to the collector nor is used without the voter's authorization [16], preventing an adversary from generating a nullifying trapdoor.

The shuffle proof π_{CEV} ensures that each shuffled commitment C'_i opens to the same trapdoor value g^{t_i} as its pre-shuffle counterpart. The proof π_{Drop} establishes that the collector knows an opening of C'_i for every $i \notin L$. Since each ballot's encrypted trapdoor T_i reveals $\mathcal{H}(t_i)$ to the collector, the trapdoors of superseded ballots are derivable by iterating $\mathcal{H}$, so an honest collector can always open the commitments outside L.

For the final ballot, however, obtaining t_i from the known value $\mathcal{H}(t_i)$ would require inverting $\mathcal{H}$. Under the one-wayness of $\mathcal{H}$ and the binding property of Pedersen commitments, the collector cannot open C'_i and is therefore forced to place each voter's final ballot in L. Because every commitment opened in π_{Drop} carries the credential of a voter whose final ballot is thereby in L, no eligible voter's cast final ballot is dropped, and the collector cannot substitute a superseded ballot for the final one.

If no consistent trapdoor chain exists within the bound M, the collector publishes the stored openings together with proofs of correct trapdoor decryption, providing publicly verifiable evidence of smartcard misbehavior and allowing the affected ballots to be discarded.

Individual Verifiability. Individual verifiability requires that an adversary cannot substitute the voter's verification target with another voter's ballot matching the same vote selection, and that the verified ballot is the final one tallied. In the proposed system, these properties hold assuming the smartcard is uncompromised and has not been used without the voter's authorization [16].

The voter locates their ballot (E, V) on the bulletin board and knows from their smartcard invocation history (genuine or fake) whether it will be selected for tallying. To verify ownership, the voter opens their credential commitment $E = \mathsf{h}^{\beta}\mathsf{pk}$ using their known blinding factor β, where pk is listed on the registration roll. The proof π_E establishes knowledge of (β, x) such that $E = \mathsf{h}^{\beta}\mathsf{g}^{x}$, and since g and h are independent generators, the commitment cannot be opened to a different public key under the discrete logarithm assumption, even with knowledge of x. This binds the ballot to the voter's credential, and the voter confirms their selection by checking the vote ciphertext $V \stackrel{?}{=} \text{Enc}_{\mathsf{pk}_{\mathcal{T}}}(v; r)$ using their secret randomization factor r and chosen vote v.

Participation Privacy. Participation privacy comprises two properties: participation anonymity, where no observer can determine from public evidence whether a voter participated, and participation receipt-freeness, where the voter cannot prove to a third party whether they participated or abstained. A ballot

(C, E, V, π_E, π_v) posted on the bulletin board contains no information identifying the voter. The credential commitment $E = \mathsf{h}^{\beta}\mathsf{pk}$ is perfectly hiding, so without knowledge of β every public key on the registration roll is an equally plausible opening. The proof π_E is a proof of knowledge that reveals nothing beyond the submitter's ability to open the commitment, and the shuffled commitments $\{E'_i\}_i$ are never opened. A coercer therefore cannot determine from the public record whether a given voter's key appears among the posted ballots, and the voter cannot produce a convincing proof of abstention. Defense against forced participation, where a coercer compels the voter to cast a specific vote, is addressed under coercion resistance via deniable vote updating and null vote casting.

Coercion Resistance. Coercion resistance requires that a voter can execute a counter-strategy such that the coercer cannot distinguish whether the voter complied with their demands or deviated in favour of their own intent. The proposed system supports two mechanisms: deniable vote updating and fake credentials. Both assume an honest smartcard and collector. We additionally require that the voter's chosen plaintext is not unique in the final tally if plaintext-based tallying is used.

Receipt-Freeness. A voter who wishes to sell their vote or prove compliance to a coercer can disclose the ballot contents $(C, E, T, V, \alpha, \beta)$ and demonstrate that this ballot appears on the bulletin board. However, to issue a conclusive receipt the voter must also prove that this ballot is included in the final tally rather than superseded by a ballot with a lower order trapdoor commitment. This requires establishing a link between the pre-shuffle ballot and a post-shuffle entry in L which is not possible as π_{CEV} and π_{Drop} are zero knowledge. Moreover, the cleansing phase reveals no revoting statistics—the set L discloses only the number of distinct voters, not how many times any voter cast a ballot—thereby excluding the "1009 attack" [2]. The voter therefore cannot produce a convincing receipt that their disclosed ballot is the tallied one. A separate receipt channel arises if the voter's plaintext choice is unique on the tally board (the Italian attack); homomorphic tallying is inherently immune to this as it hides individual votes, whereas mixnet-based tallying must employ additional countermeasures.

Vote Update Deniability. By receipt-freeness, the coercer cannot verify that any observed ballot is included in the final tally. The voter's counter-strategy is therefore to comply with the coercer and later recast their vote in private using the genuine credential. The coercer cannot detect the voter's subsequent session from the public record, since each session uses fresh blinding factors (α, β) and a new trapdoor t_n, producing indistinguishable credential and trapdoor commitments making cast ballots unlinkable. The trapdoor index n is hidden by encrypting the trapdoor. Consequently, the smartcard outputs do not allow a coercer to determine whether the voter has already voted or submitted a revote.

Fake Credentials in the proposed system can still cast valid ballots, making them a weaker form than those in [2,12], where ballots cast with fake credentials are discarded entirely. A fake credential occupies a higher position in the trapdoor

chain than the genuine credential, so a ballot cast with the genuine credential always takes precedence during cleansing. This protects against coercion scenarios in which a coercer arrives at the last moment and forces the voter to cast the final ballot under observation: if the voter uses a fake credential under coercion, they can cast a ballot with their genuine credential at any time during the voting phase, which then takes precedence. A residual attack vector arises if the coercer demands both the genuine and fake credentials; however, this is mitigated because the coercer does not know how many fake credentials the voter may have generated and therefore always faces the possibility of being one credential short of the genuine one.

Everlasting Privacy. The system offers practical everlasting privacy [9] assuming an honest collector and that the submission channel resists external timing analysis. The channel need not hide the voter's identity from the collector, who necessarily learns it through the private openings (α, β, t_{n+1}). A possible mechanism for providing such timing-analysis resistance is discussed in Sect. 6.

A ballot posted on the bulletin board contains the commitments (C, E) and vote ciphertext V. A future adversary with unlimited computational power could decrypt the vote ciphertext V. However, the Pedersen commitments $E = \mathsf{h}^{\beta}\mathsf{pk}$ and $C = \mathsf{h}^{\alpha}\mathsf{g}^{t_n}$ are perfectly hiding: for any public key pk' on the registration roll, a suitable blinding factor β' exists such that $E = \mathsf{h}^{\beta'}\mathsf{pk}'$, and similarly for C. The shuffled commitments (C', E') inherit the same property. Since the proofs π_E, π_{CEV}, π_{Drop} are zero-knowledge and therefore of no additional help, a future adversary cannot find out who cast the corresponding encrypted vote V.

6 Discussion and Implementation

Strengthening Individual Verifiability. Currently individual verifiability depends on the smartcard's secrets remaining confidential and on no unauthorized ballots having been issued with it [16]. These trust assumptions could be relaxed by supplementing the protocol with vote trackers. Since cast-as-intended verifiability is already assured, the tracker need only prove that a ballot was selected for tallying, requiring only the publication of tracker values. This can be achieved with tracker-based systems that offer receipt-freeness [4]. A further goal would be to assign trackers to non-participating voters, enabling them to verify non-participation in a deniable manner, but this necessitates further research.

Resisting Metadata Analysis. Everlasting privacy can be strengthened by releasing ballots in a shuffled order. To make this accountable, the collector first publishes blinded commitments $E_i^* = \mathsf{g}^{\rho_i} E_i$ during the voting phase and returns (ρ_i, E_i^*) as an acknowledgment to the voter. Once the voting phase closes, the collector performs a shuffle with permutation $\tau \in \mathbb{S}_I$ supported by a proof $\pi_{\text{Anon}} = \text{PoK}\{\tau, \{\rho_i\}_i : E_i = \mathsf{g}^{-\rho_i} E^*_{\tau(i)}\}$ and releases $(E_i)_i$ together with the remaining ballot components.

Implementation. Ballot creation requires two Pedersen commitments with one opening proof and one ElGamal encryption, amounting to seven group exponentiations and three multiplications in $\mathbb{G}_q$, which is well within the capabilities of existing smartcards used for digital signatures. A single trapdoor chain is sufficient when voting is non-concurrent; for concurrent elections, multiple chains can be installed. With 128-bit trapdoor security and 1000 revotes shared across genuine and fake credentials, the full chain requires up to 15 kB of storage.

The shuffle proofs can be implemented using existing ElGamal reencryption shuffle proofs such as Verificatum [7,22,23], with the generator and public key for commitment shuffles set equal to the blinding generator, and the vote reencryption shuffle using the standard generator and tallier's public key combination. The drop proof is linear in the number of dropped ballots, and trapdoor root finding scales with the chain length. The cleansing phase therefore scales linearly, making it practical for large-scale elections.

7 Conclusion

We have presented a novel construction to provide public verifiability for the ballot cleansing phase. The protocol offers perfect cleansing privacy, scales linearly with a footprint approaching two shuffle proofs, imposes minimal smartcard requirements, and decouples from the tallying scheme, enabling integration with both homomorphic and mixnet-based tallying.

The scheme enables Estonian-style voting systems to achieve public cleansing verifiability without exposing internal data to auditors, eliminating a potential access point for coercers. Although votes are ElGamal encrypted, blinding the public keys provides practical everlasting privacy, enabling the publication of election evidence. Resilience against corrupt collectors is provided by relay networks that independently verify ballot eligibility and offer alternative submission channels. Individual verifiability is also strengthened: voters verify their ballot directly from the bulletin board, including the signer's public key, precluding verification of another voter's ballot.

Future work should address smartcard trust assumptions by coupling the protocol with tracker constructions, which may also enable verifiable and deniable proof of non-participation.

Acknowledgment. We are particularly grateful to the reviewer who suggested replacing the span-proof machinery of an earlier version with direct proofs of knowledge of the dropped ballots' trapdoor openings—a simplification that substantially improved this paper. Reviewer feedback also prompted us to simplify the trapdoor-chain construction. We also thank Véronique Cortier and Jan Willemson for helpful feedback on an earlier draft of this manuscript.

References

1. Benaloh, J., Rivest, R.L., Ryan, P.Y.A., Stark, P.B., Teague, V., Vora, P.L.: End-to-end verifiability. ArXiv (2015)

2. Cortier, V., Gaudry, P., Yang, Q.: Is the JCJ voting system really coercion-resistant? In: IEEE Computer Security Foundations Symposium (2024)
3. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Advances in Cryptology — EUROCRYPT '97. Springer (1997)
4. Damodaran, A., Rastikian, S., Rønne, P.B., Ryan, P.Y.A.: Hyperion: Transparent end-to-end verifiable voting with coercion mitigation, (2024). IACR ePrint
5. 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)
6. Gutmann, P., Neuhaus, S.: Replication of quantum factorisation records with an 8-bit home computer, an abacus, and a dog. IACR Cryptology ePrint Archive, (2025)
7. Haenni, R., Locher, P., Koenig, R., Dubuis, E.: Pseudo-code algorithms for verifiable re-encryption mix-nets. In: Financial Cryptography and Data Security, Springer (2017)
8. Haenni, R., Spycher, O.: Secure internet voting on limited devices with anonymized DSA public keys. In: USENIX EVT/WOTE (2011)
9. Haines, T., Mosaheb, R., Müller, J., Pryvalov, I.: SOK: Secure e-voting with everlasting privacy. In: Proceedings on Privacy Enhancing Technologies (2023)
10. Heiberg, S., Martens, T., Vinkel, P., Willemson, J.: Improving the verifiability of the Estonian internet voting scheme. In: Electronic Voting, Springer (2017)
11. Huber, N., et al.: Kryvos: Publicly tally-hiding verifiable e-voting. In: ACM CCS (2022)
12. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: ACM Workshop on Privacy in the Electronic Society (2005)
13. Krips, K., Willemson, J.: On practical aspects of coercion-resistant remote voting systems. In: Electronic Voting, Springer (2019)
14. Kulyk, O., Teague, V., Volkamer, M.: Extending Helios towards private eligibility verifiability. In: E-Voting and Identity, Springer (2015)
15. Mueller, J., Pejó, B., Pryvalov, I.: Devos: Deniable yet verifiable vote updating. IACR Cryptol. ePrint Arch. **2023**, 1616 (2024)
16. Pereira, O.: Individual verifiability and revoting in the Estonian internet voting system. In: Financial Cryptography and Data Security, Springer (2023)
17. Pfitzmann, B.: Breaking an efficient anonymous channel. In: EUROCRYPT. LNCS, vol. 950, pp. 332–340. Springer (1995)
18. Rønne, P.B.: JCJ with improved verifiability guarantees. In: EVoteID (2016)
19. Smyth, B., Frink, S., Clarkson, M.R.: Computational election verifiability: Definitions and an analysis of Helios and JCJ. IACR Cryptology ePrint Archive (2015)
20. Springall, D., et al.: Security analysis of the Estonian internet voting system. In: ACM Conference on Computer and Communications Security (2014)
21. Waintal, X.: The quantum house of cards. In: Proceedings of the National Academy of Sciences of the United States of America (2023)
22. Wikström, D.: A sender verifiable mix-net and a new proof of a shuffle. In: ASIACRYPT. Springer (2005)
23. Wikstrom, D.: How to implement a stand-alone verifier for the verificatum mix-net. verificatum.org (2011)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



It Doesn't Take a Thief: Optical-Scan Voting Systems Fail Even Without Adversaries

Aleksander Essex[1(✉)] and Philip B. Stark[2]

[1] Western University, London, Canada
aessex@uwo.ca
[2] University of California, Berkeley, USA
pbstark@berkeley.edu

Abstract. Optical-scan voting systems and their supporting ecosystem of people, processes, and technology are fallible. While a substantial body of work examines adversarial threats to such systems, we have encountered jurisdictions where the possibility of tabulator error is not fully internalized. Stakeholders there often find hypothetical attacks unconvincing, but some are persuaded by real-world accounts of equipment and procedural failures. This paper introduces a taxonomy of non-adversarial failure modes organized into intuitive categories: recording votes on paper, reading votes from the paper, combining votes as read into a reported outcome, and testing and verifying, all illustrated with documented incidents. We map common verification mechanisms against this taxonomy, identifying gaps that no paper-based audit can detect or correct, most notably failures that compromise the trustworthiness of the paper trail, such as giving voters the wrong ballot style (omitting contests they are eligible for, or including ones they are not), using ballot-marking devices to record votes, or failing to keep voted ballots secure and organized.

Keywords: optical-scan tabulators · non-adversarial failure modes · risk-limiting audits

1 Introduction

Paper ballots are the most widely trusted medium for recording votes, for good reason: they create a durable, tamper-evident, human-readable record of voter selections independently of the method used to count the votes. Hand counting votes from paper ballots is still common globally, providing a robust, immediate means of delivering on a universal design requirement of democratic elections to allow "independent scrutiny of the voting and counting process ... so that electors have confidence in the security of the ballot and the counting of the votes" [45].

Some jurisdictions (most notably the US) use optical-scan tabulators to speed up tabulation and mitigate human error in vote counting. In the United States,

© The Author(s) 2027
D. Duenas-Cid et al. (Eds.): E-Vote-ID 2026, LNCS 17083, pp. 35–54, 2027.
https://doi.org/10.1007/978-3-032-39512-2_3

over 95% of voters live in jurisdictions using paper ballots [50]. However, a paper ballot is not a counted vote. Between the moment a voter expresses their intent and the moment the official results are published, votes are subject to loss, alteration, substitution, and fabrication through hardware, software, and human procedures.

While these vulnerabilities are well known among academics, in our experience, they are not widely acknowledged by election officials and legislators, resulting in reluctance to take countermeasures such as physical accounting for voted ballots and conducting risk-limiting audits. This paper aims to persuade such stakeholders that these measures are needed.

1.1 Optical Scan Systems Are Fallible

There are many documented failures, with root causes ranging from unexpected 'inputs' such as fold-lines in ballots being incorrectly interpreted as voter marks by inadequately maintained scanners [22], to procedural failures such as including test ballots in the election results [20], to ballot printers that are inadequate for the paper stock used [31], to seemingly mundane environmental factors such as humid air at polling places causing paper jams [27]. Detailed incident corpora list examples dating back at least to the early 2000 s [34].

A substantial body of work examines adversarial threats to voting systems, including demonstrated attacks on memory cards [16], comprehensive security reviews [8,51], and ongoing hardware vulnerability assessments [24]. Although accounting for and designing to adversarial threat models is an essential activity, our experience engaging with election management bodies suggests that stakeholders are generally more receptive to real-world accounts of equipment and procedural failures than to hypothetical attack scenarios. We have even encountered institutional settings in which the very possibility of tabulator error has not been fully internalized.

To the extent that election stakeholders are the audience for this work, we believe a non-adversarial framing may be more effective at building the case for meaningful, routine post-election audits.

Trust and Institutional Context. A non-adversarial framing serves both ends of the institutional trust spectrum. In low-trust environments, it helps distinguish ordinary fallibility from unfounded fraud claims; in high-trust environments (as in Canada), it helps stakeholders internalize that trust in outcomes should rest on verifiable processes rather than assumed equipment reliability.

1.2 Contribution and Outline

This paper presents a systematic model of non-adversarial tabulator failure modes organized around a four-stage pipeline: vote recording, covered in Sect. 4; vote reading, covered in Sect. 5; vote aggregation, covered in Sect. 6; and verification and testing, covered in Sect. 7. Unlike prior taxonomies organized by root cause, we structure failures by *where* they manifest in the pipeline rather than

what caused them to better align with the vantage point from which an election official actually experiences them. We map selected incidents from the United States onto this framework, demonstrating that failures have been observed at every stage of the pipeline, across vendors, jurisdictions, and election types. Section 8 matches common verification mechanisms to the taxonomy, identifying gaps that no paper-based audit can cover on its own.

While prior work has systematized some sub-categories, such as optical mark detection [5,25], memory card reliability [1,11], and L&A testing [53], as well as incident collections, we believe this is the first paper to propose a pipeline-structured taxonomy of non-adversarial failures spanning the full tabulation process.

2 Background

The US leads global adoption of optical-scan ballot tabulators. According to Verified Voting's 2026 polling-place equipment data [50], 96% of US voters are in jurisdictions that use ballot tabulators: 69.5% in jurisdictions that offer hand-marked paper ballots, with electronic ballot marking devices (BMDs) and/or direct-recording electronic touch-screens (DREs) provided as an accessibility option, and 26.5% in jurisdictions that require all in-person voters to use BMDs. Only 3.9% of US voters are in jurisdictions that use DREs as the sole in-person voting method.

Canada uses optical-scan tabulators only sub-nationally—in Ontario municipal (since the early 2000 s) and provincial (since 2018) elections, and in some British Columbia, New Brunswick, and Manitoba contests—while federal elections are still counted entirely by hand. Adoption elsewhere is limited: the Philippines has used precinct-count scanners nationwide since 2010, South Korea uses central-count scanners for hand-marked ballots, and the UK has used optical scanning only occasionally (e.g., the 2004 London Assembly and 2007 Scottish Parliament elections).

There are two main optical-scan voting technologies: mark-sense and imaging scanners. Mark-sense scanners, an older technology still deployed in some US states, use an array of photoreceptors to detect reflectance at fixed positions along the ballot, interpreting marks at vote-target coordinates as votes and reading ballot-style information encoded the same way. Imaging scanners instead capture a digital image of the ballot and process it to read the votes, either from the voter's hand marks (together with encoded ballot-style information) or, for BMD printouts, from a machine-readable code; some recent systems use optical character recognition.

There are likewise two main system configurations: precinct-count (PCOS) and central-count (CCOS). In PCOS, the voter inserts the ballot into a combined scanner and ballot box. In CCOS, officials feed collected ballots (from precincts, vote centers, and mail) into a scanner at a central location. CCOS systems use high-speed scanners that are generally more accurate than the "consumer-grade" scanners typical of PCOS. Many US jurisdictions combine the two—for instance, PCOS for in-precinct voting and CCOS for vote-by-mail (VBM).

2.1 Related Work

Vulnerabilities of optical-scan ecosystems are well studied in the academic literature, which generally falls along three lines.

1. **Non-adversarial**: Threats to elections stemming from both potential risk and observed instances of procedural lapses and equipment failures [6,11,25,51].
2. **Adversarial**: Threats to the elections stemming from cybersecurity vulnerabilities and the potential for intentionally malicious efforts to compromise integrity or ballot secrecy [10,23,24,28].
3. **Verifiability**: Efforts to make election results produced by optical-scan tabulators efficiently and independently verifiable through statistical checks like risk-limiting audits (RLAs) [26], or ballot secrecy-preserving cryptographic proofs like end-to-end verifiability (E2E-V) [9,14,37].

To our knowledge, our paper is the first to propose a pipeline-structured taxonomy of non-adversarial failures covering the full tabulation stack, but there is related work. Norden [34] lists nearly 200 incidents, organized by vendor and year, rather than by failure type. Walker et al. [53] address logic and accuracy (L&A) testing practices across 50 states; that work is related but focuses on a single verification mechanism rather than the full failure surface. Jones [25] categorizes optical mark-sense scanning mechanisms and image capture failure modes. Bajcsy et al. [5] systematize marginal mark types as a subcategory of mark interpretation failures. Davtyan et al. [11] and Antonyan et al. [1] categorize memory card failures. A NIST workshop [33] produced a preliminary taxonomy organized around attack vectors rather than operational failure modes. The EAC used attack trees to decompose and systematize threats [46].

2.2 Methodology

The incidents presented here were collected from the academic literature on voting system security and reliability and institutional reports from the EAC, state election authorities, and civil society organizations such as the Brennan Center for Justice and Verified Voting, targeted web searches for documented incidents, discussions with domain experts, and our own direct experience auditing elections and testifying regarding contested election results. We also used large language models as a search aid to identify candidate incidents, which we verified against primary or secondary sources. While the incidents and analysis in this paper are drawn from the US, the failure modes we describe are inherent to the technology, not specific to any jurisdiction.

The incidents below are not an exhaustive list, but they show that failures have occurred at every stage of the pipeline, across vendors, jurisdictions, and election types within the US. Incident-based analysis carries inherent selection bias, since failures may be undetected (especially in jurisdictions without rigorous canvasses and audits) or unreported.

3 Tabulator Ecosystem and Failure Modes

Optical-scan voting systems operate in a broader ecosystem of hardware, software, and human procedures that spans the full pipeline from ballot production to result certification. Any stage can introduce errors that may be invisible to subsequent stages.

3.1 Tabulation Pipeline and Failure Modes

We organize failures into four broad stages of collecting votes and reporting results:

1. **F1: Vote Recording (Sect. 4).** A voter marks a ballot by hand or uses a ballot-marking device to print one; the output is a marked paper ballot, human-readable and possibly also machine-readable (e.g., a QR code). Failures present the voter the wrong contests or record something other than their expressed preferences.
2. **F2: Vote Reading (Sect. 5).** The ballot is scanned and interpreted by software configured for the election, producing a cast vote record (CVR) or simply incrementing contest vote counters. Failures record or use the selections differently from what a reasonable human observer would read from the ballot.
3. **F3: Vote Aggregation (Sect. 6).** The votes as-read, possibly partially aggregated, are transmitted to a central election management system (EMS) and combined into official results. Failures turn otherwise correct interpretations into incorrect totals and outcomes.
4. **F4: Verification and Testing (Sect. 7).** Checks before the election (logic and accuracy testing) and after it (ballot accounting, pollbook reconciliation, chain-of-custody and seal checks, post-election audits) can prevent or recover from F1–F3 failures, unless the marked ballots themselves are untrustworthy, or the tests are missing, poorly designed, or poorly executed.

Vulnerabilities of optical-scan systems extend well beyond the tabulator hardware and software: the election definition software (F2) and the EMS (F3) are external systems whose failures are indistinguishable from tabulator failures in the final result.

Relationship to Security Definitions. *Software independence* (SI) requires that an undetected error in a voting system's software cannot cause an undetectable change in an election outcome [36], and is a formal requirement of the VVSG 2.0 [48]. Optical-scan systems can be software-independent: hand-marked paper ballots provide a human-readable record independent of the software used to count them. An election is *defensible* [3] if the election official can provide affirmative public evidence that the reported outcomes are correct, despite any errors or malfeasance that might have occurred. *Evidence-based elections* [4,39] provide affirmative public evidence that elections were decided correctly by combining demonstrably trustworthy vote records with risk-limiting audits (F4). SI, defensibility, and being evidence-based are properties of a run of the complete system (an election), not just the hardware. The primary failure modes (F1–F3) indicate what an audit should protect against.

3.2 Terminology

We distinguish among three related notions that are often conflated:

1. **Voter intent**. The candidate or option the voter intends to select. Voter intent is known only to the voter.
2. **Expressed preference**. The selection the voter *actually makes* by marking a ballot by hand or indicating a selection on a BMD using the touchscreen or an audio interface. Expressed preferences differ from voter intent if the voter errs in expressing their intent [3].
3. **Marked selection**. The human-readable selection physically recorded on the paper ballot. In some BMD systems, the tabulator reads votes encoded in QR codes or barcodes, not the human-readable votes, but we consider the human-readable text to be the marked selection.

In the context of this discussion, the *marked selection* is the ground truth against which tabulator output is evaluated. In rare cases, the marked selection is ambiguous to a reasonable human observer (for instance, the marked selection was ambiguous in only a handful of ballots in the 2008 Franken–Coleman state-wide recount [32]). The distinction between (2) and (3) arises in the case of ballot-marking devices. With hand-marked ballots (HMPBs), the voter's expressed preference is the marked selection. With BMD-printed ballots, the expressed preference may differ from the marked selection as the result of an F1 vote recording failure, as the DeKalb County 2022 incident in the next section illustrates.

4 Vote Recording (F1)

4.1 Components

The correct output of vote recording is a paper ballot that accurately reflects the voter's expressed selections in every contest the voter is eligible to vote in (and no other contests), in human-readable form and optionally also machine-readable. F1 begins with a printed blank ballot or a BMD configured for the election and ends with a marked paper ballot:

Ballot printing and/or BMD configuration. BMDs sometimes fail to function entirely, resulting in 'denial-of-voting.' Jurisdictions such as the state of Georgia and Los Angeles County, CA, that require all in-person voters to use BMDs are especially vulnerable to such failures. Both HMPB and BMDs are subject to *ballot presentation* failures: the printed ballot or the BMD interface may omit or include some contests or candidates erroneously.

BMDs. BMDs translate the voter's expressed preferences in the presented contests into selections on a printed ballot. The voter then has the opportunity to inspect the printed ballot before inserting it into the scanner. All-in-one BMDs (e.g., ES&S ExpressVote XL) may (at the voter's instigation) deposit the printout directly into a ballot box, bypassing the opportunity to review the printed selections.

HMPB. With hand-marked paper ballots, the paper record begins as an accurate record of the voter's expressed preferences in the contests presented on the ballot.

Curation of cast ballots. The cast ballots are the physical ground truth for post-election audits and recounts. Maintaining secure custody of the ballots is an artifact-integrity matter: losing, adding, damaging, or altering cast ballots undermines the trustworthiness of the election, regardless of whether other processes functioned correctly.

Table 1. Vote recording incidents (F1). P = Process failure; A = Artifact integrity failure.

Ref.	Incident	Component	Type	Outcome
[15]	Georgia, 2020	BMD	P, A	BMDs did not function
[18]	Georgia, 2020	BMD	P, A	BMD does not display all candidates in a contest
[21]	Louisville, KY, 2021–2026	HMPB	P, A	Voters assigned to the wrong precinct, presented the wrong contests
[12,13]	DeKalb County, GA, 2022	BMD	P	Incorrect ballot def. Selections assigned to the wrong candidate
[52]	Northampton County, PA, 2023	BMD	P	Expressed preferences swapped for two candidates

4.2 Selected Incidents

Table 1 summarizes selected F1 incidents. We briefly elaborate on three cases.

Voters given incorrect ballot style in Louisville, KY, 2021–2026. [21] A clerical error in assigning voters to precincts resulted in giving voters ballots that omitted some contests they were eligible to vote in. Some errors that result in giving voters the wrong collection of contests are recoverable and some are not. In general, if contests (or candidates) are omitted, the error is unrecoverable because voters' selections were never solicited and recorded. Depending on details, it can be possible to recover from errors that erroneously add a contest to ballots, for instance, if the error occurs for every ballot of a given style.

BMD fails to display some candidates in Georgia, 2020 A bug in Dominion voting system software (not a configuration error) caused BMDs to fail to display about half the candidates in a US Senate special election [18]. The accidental discovery of the bug during L&A testing triggered Dominion to issue a software patch that was deemed a *de minimis* change by the US Election Assistance Commission[1] and therefore exempted from additional testing—even though the original bug had slipped through VVSG certification testing.

BMDs in DeKalb County, GA, 2022. Following the withdrawal of a candidate in the Democratic primary for DeKalb County Commission District 2, a ballot definition error in Dominion BMD touchscreens caused votes intended

[1] See also https://www.eac.gov/voting-equipment/engineering-change-orders/dvs-100714, last accessed 26 July 2026.

for one remaining candidate to be recorded as votes for another [12,13]. In this instance, the error was introduced when the BMD committed the voter's marked selections to paper (the tabulators themselves interpreted and counted ballots correctly). This case illustrates a characteristic of BMD failures: because the paper ballot reflects what the BMD printed rather than what the voter selected, software independence holds only if the voter verified the printed ballot before casting it, which studies have shown most voters do not perform in practice [3,7].

5 Vote Reading (F2)

5.1 Components

F2 failures include mark-detection errors, stray marks or paper folds read as votes, ballot-definition errors, discrepancies between a vote's human-readable and barcode/QR encoding, and incorrect application of vote-validity rules. The F2 pipeline scans a ballot and uses ballot-definition information to interpret the scan, creating a (possibly ephemeral) cast vote record that is stored or used to increment contest vote counters. This interpretation depends on a chain of logical mappings that must be mutually consistent across the ballot definition, the tabulator, and the EMS:

Scanning. The scanner ingests the ballot under the prevailing environmental conditions. Process failures include mechanical feed problems such as jams, double-feeds, and skipped ballots or batches of ballots [25,27,42], and optical failures such as sensor degradation, dirty scanner heads [22], image skew or distortion, print-through from the opposite side of a ballot sheet, insufficient resolution and insufficient image bit depth. Humidity can disrupt feed mechanisms and degrade image quality [25]. Sensor contamination accumulates over time if maintenance is inadequate [22].

Ballot definition (process). The ballot definition process encompasses authoring and configuration to present the correct collection of contests with the correct choice options to each voter and to read marked ballots. For HMPB, this generally includes mapping physical target positions in the ballot geometry to candidate identifiers in the tabulation system, depending on ballot-style information encoded on the printed ballot. Process failures include mapping selections to the wrong contest or candidate, or applying the wrong voting rules to the contest. Because a single mapping error affects every ballot of a given style, these failures tend to be systematic and are often undetectable from aggregate results.

Ballot definition file. The ballot definition file is a digital artifact that carries the ballot geometry and candidate/position mappings into each tabulator. For interpretation to be correct, three things must be mutually consistent: (1) the physical positions of vote targets on the printed ballot, (2) the definition file loaded on the tabulator, and (3) the version used by the EMS to decode CVRs. Any pairwise mismatch produces a failure. A last-minute ballot revision may be propagated to tabulators and the EMS, but leave already-printed ballots

inconsistent with the new definition. Alternatively, a revision may update the EMS but not all tabulators, as in the 2020 Antrim County incident [19].

Mark interpretation. The mark interpretation process evaluates the digitized ballot image against the ballot definition to determine which ovals contain valid votes. Process failures arise from threshold calibration errors. Over-sensitive configurations count stray marks as votes, and under-sensitive ones miss valid marks. Process failures also arise from physical ballot artifacts such as fold-lines or smudges that mimic filled ovals, and from sensor-to-sensor variation that causes identical marks to be evaluated differently across machines [22].

Cast vote record (CVR). The CVR is a data artifact produced by mark interpretation, encoding the tabulator's per-ballot vote assignments for downstream aggregation for some optical-scan systems. Artifact integrity failures include file corruption between the tabulator and the EMS and unauthorized modification of CVRs prior to aggregation.

Table 2. Vote reading incidents. P = Process failure; A = Artifact integrity failure.

Ref.	Incident	Component	Type	Outcome
[31]	Maricopa County, AZ, 2022	Scanning	P	Timing marks too faint to read
[27]	New York City, NY, 2018	Scanning	P	Paper jams (humidity)
[35]	Wake County, NC, 2018	Scanning	P	Ballots rejected (humidity)
[29]	Maryland, 2016	Scanning	P	Double-feeds. Scratched lens caused false overvotes
[30]	Cambria County, PA, 2024	Scanning	P	Printing error made ballots unreadable
Fig. 1	San Francisco County, CA, 2024	Scanning	P	Scanner erases light marks
[19]	Antrim County, MI, 2020	Ballot definition file	A	Mismatched definition. Votes mapped to incorrect candidates
[22]	Windham, NH, 2020	Mark interpretation	P	Fold-lines incorrectly interpreted as votes
[2]	Los Angeles County, CA, 2023	Mark interpretation	P	Stray marks outside ovals read as overvotes

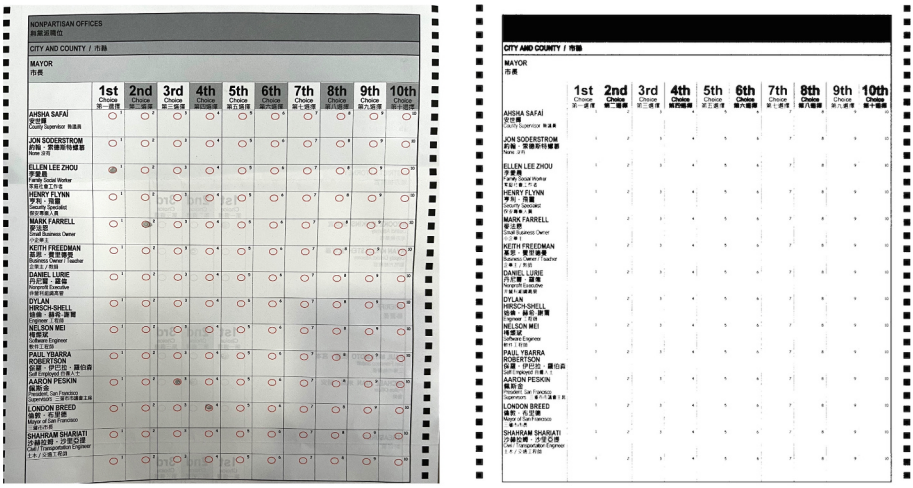
5.2 Selected Incidents

Table 2 summarizes selected F2 incidents. We elaborate on four, with two of them illustrating failures at opposite ends of the F2 pipeline: sensor-level mark interpretation (Windham) and ballot definition files (Antrim).

Scanning in Maricopa County, AZ, 2022. Ballot-on-demand printers failed to maintain sufficient fuser temperature, causing timing marks to print too faintly for Dominion ICP tabulators to read [31]. Ballots were later re-tabulated on central-count scanners, an outcome made possible only because paper ballots were retained and a recovery procedure was in place. The incident illustrates both the sensitivity of scanning to upstream printing conditions and the importance of contingency procedures for F2 faults.

Scanning in San Francisco, CA, 2024. Figure 1 shows a digital picture of a voted ballot and a Dominion optical-scan system scan of the same ballot, from an election in San Francisco in November 2024. The voter's marks are clear in

the picture, but there is no trace of the marks in the scanned image because the Dominion scanner records the image in black-and-white (not grayscale or color) and the mark is below the threshold that renders as black.



(a) Author-taken photograph of a ballot. (b) What the Dominion scanner captured.

Fig. 1. Example F2 Failure. Two conflicting images of a ballot cast in San Francisco in November 2024. **(a)** Photo taken by one of us (Stark) during the post-election audit showing valid voter-made marks. **(b)** Image created by the scanner failing to capture marks. Observe the print-through from the other side is visible in (a). Lowering the image's black threshold could result in print on the back side being interpreted by the scanner as a marked oval on the front side.

Mark Interpretation in Windham, NH, 2020. AccuVote mark-sense scanners misinterpreted fold-lines on absentee ballots as voters' marks [22]. A letter-folding machine had been repurposed to handle a surge in absentee ballots, and the resulting creases intersected candidate vote ovals, creating erroneous votes and overvotes. Inconsistent maintenance left the scanners with varying sensor contamination, so the same creased ballots were read differently across machines.

Ballot Definition File in Antrim County, MI, 2020. A last-minute ballot revision added a write-in blank to one township's ballot, causing the Dominion EMS to reassign sequential vote-target IDs across all ballot styles. Many tabulators were not reprogrammed with the updated definition, leaving them operating with a stale ballot definition file. When the EMS decoded the resulting memory cards against the new IDs, votes were mapped to the wrong candidates [19]. A full hand recount confirmed the correct result, illustrating the three-way consistency requirement. Notably, all three independent records disagreed: the poll tapes reflected the tabulator's totals under the old definition, the EMS totals

reflected decoding under the new definition, and the paper ballots retained the ground truth. The divergence illustrates why retaining paper ballots *and* printed tally tapes is important. The hand count corrected the outcome, while the tape-vs-EMS discrepancy identified where in the pipeline the failure occurred.

6 Vote Aggregation (F3)

6.1 Components

F3 failures tend to be systematic: a single procedural lapse or software error can affect entire batches, precincts, or jurisdictions. The F3 pipeline carries per-ballot CVRs through accumulation and transmission to produce jurisdiction-wide totals:

Vote accumulation. The tabulator accumulates votes across all ballots processed in a session. Process failures include arithmetic errors, counter overflow, and software bugs that cause votes to be dropped or double-counted. One infamous category of aggregation failure involves *negative* vote totals, which can arise from arithmetic faults or corrupted counters [34].

Results artifact (memory card). The vote totals results artifact is typically implemented as a removable memory card, which carries per-tabulator totals and CVRs from the polling place to the EMS. Memory cards can be lost, uploaded multiple times, initialized with corrupt data, or simply never uploaded. The 2020 Georgia hand recount identified both failure modes: approximately 5,900 ballots across four counties were found uncounted because memory cards had never been transferred to the EMS, while in Fulton County some ballot cards were included in the tally multiple times in the original count and in a machine recount due to procedural failures [42].

Election management system (EMS). The EMS aggregates results from all tabulators into a jurisdiction-wide result. Process failures include software bugs that drop or duplicate batches, misconfigured aggregation logic, and (as in the NYC 2021 incident [20]) failure to clear test data before tabulation. Because the EMS process operates on results from multiple tabulators simultaneously, failures here tend to have the widest scope of any component in the pipeline.

6.2 Selected Incidents

Table 3 summarizes selected F3 incidents. We elaborate on two cases that illustrate both process and artifact integrity failures at this stage.

Results Artifact in Volusia County, FL, 2000. In this landmark incident, a memory card in Precinct 216 reported *negative* 16,022 votes for Al Gore [34], which the EMS did not range-check. The erroneous total contributed to premature reporting, calling Florida for George W. Bush. The VVSG 2.0 now explicitly

Table 3. Vote Aggregation incidents (F3). P = Process failure; A = Artifact integrity failure.

Ref.	Incident	Component	Type	Outcome
[34]	Volusia County, FL, 2000	Results artifact	A	Memory card reported negative vote totals
[1]	Connecticut, 2007–09	Results artifact	A	Up to 15% of memory cards contained random data after battery replacement
[17]	St. Lucie County, FL, 2012	Results artifact	A	Procedural errors caused double-scanned and unscanned ballots
[20]	NYC Mayoral Primary, 2021	EMS	P	Test ballots included in tabulation, skewing results
[38,44]	Monmouth County, NJ, 2022	EMS	P	Omitted software patch led to double-counted votes
[42]	Fulton County, GA, 2020	Results artifact	A	Ballots counted multiple times or omitted due to inadequate tracking
[43]	Alameda County, CA, 2022	EMS	P	Ranked-choice voting rules for wrong jurisdiction used. Wrong winner reported

requires that electronic devices detect and prevent the accumulation of negative votes [49], a requirement that did not exist at the time.

EMS Process Failure in Monmouth County, NJ, 2022. In July 2022, an ES&S technician reinstalled election management software on Monmouth County's server to resolve a network connectivity issue, but omitted a software patch designed to prevent duplicate ballots [38]. Some votes in four municipalities were double-counted, leading to the wrong candidate being certified until a recount identified the error some months later [44]. The incident illustrates how the absence of reconciliation checks between precinct-level totals and EMS aggregates allowed the error despite system certification.

7 Verification and Testing (F4)

7.1 Components

Unlike F1–F3, where the failure surface is defined by hardware, software, and data artifacts, the F4 failure surface involves the *design* of the verification process, including what it checks and how thoroughly it covers these failure surfaces relative to the paper ballot ground truth. It also involves the *execution* of the audit by human auditors. For instance, post-election audits involve retrieving ballot sheets, manually interpreting the votes on those sheets, and entering the interpretation into the audit software. The F4 pipeline checks results against independent artifacts:

Logic and Accuracy (L&A) testing. L&A testing is the primary pre-election check in most jurisdictions. A test deck of ballots is processed by each tabulator, and the tabulator results are compared against the ground-truth totals of the deck. Process failures in L&A testing arise from insufficient test deck design, such as not testing all relevant vote combinations, ballot styles, or environmental factors. A fifty-state review found that L&A implementations vary dramatically in scope and rigorousness across jurisdictions [53].

Inventory and chain-of-custody procedures. Maintains a complete accounting of physical ballots, memory cards, and tally tapes between pipeline stages, and protects artifact integrity in transit. Process failures include not physically accounting for ballot cards (relying on the voting system to know how many cards were cast), inadequate tracking of which

memory cards have been used and whether they have been uploaded, which ballot boxes have been sealed and by whom, and which tabulators have been accounted for. Detection of missing artifacts requires reconciliation against an independent expected count (e.g., poll books and other participation records, ballot accounting, and VBM processing records). The Georgia 2020 hand recount uncovered approximately 5,900 uncounted ballots in four counties attributable to memory cards never uploaded to the EMS [42], which proper reconciliation ought to have caught.

Post-election audits. Post-election audits are the primary mechanism by which F1–F3 failures can be detected and corrected after an election. Their effectiveness depends on the integrity of the paper trail and the design of the audit. If the record of the votes is incomplete, incorrect, or adulterated (some F1 failures), little can be done; an audit could even confirm a wrong result. An audit that checks only a fixed percentage of precincts or that relies on the same software or scans used for tabulation provides little assurance. Risk-limiting audits (RLAs) [41] of a trustworthy paper trail provide statistical guarantees of correcting incorrectly reported outcomes. Relatively few US jurisdictions conduct RLAs, and those that do generally audit at most a few contests, which may create a false sense of confidence in the overall election because auditing some contests cannot confirm any contest that was not audited.

Software and firmware integrity checks. Some jurisdictions require verification that tabulators are running certified software before an election, typically through cryptographic hash comparison. Like L&A testing, these are pre-election checks, but they establish which software was installed rather than whether any reported outcome is correct. Process failures include hash scripts that report success regardless of whether a valid hash file is present [47] and self-attestation in which a device reports its own hash.

7.2 Selected Incidents

Table 4 summarizes selected F4 incidents. We elaborate on two.

L&A Testing in Windham, NH, 2020. Pre-election L&A testing did not catch the Windham fold-line failure (Sect. 5) because the test deck did not include folded ballots [22]. This is a characteristic L&A failure: successful verification under sanitized test conditions, failure under operational ones.

Audit Design Limitations in Northampton County, PA, 2023. In this case, the supposed paper ballot ground truth was itself internally inconsistent: the barcode used for tabulation was correct while the human-readable summary was not, a divergence undetectable without independently decoding the barcode.

Table 4. Verification and Testing incidents (F4). P = Process failure.

Ref.	Incident	Component	Type	Outcome
[22,53]	Windham, NH, 2020	L&A testing	P	Fold-line failure not caught under operational conditions
[47]	ES&S hash verification, 2020	Software integrity check	P	Hash script reported success with missing hash file
[52]	Northampton County, PA, 2023	Post-election audit	P	Barcode correct, but hand count of human-readable text showed incorrect outcome
[42]	Fulton County, GA, 2020	Post-election audit/handcount	P	Audit tally sheets omitted from audit results

Table 5. Coverage of failure modes by verification mechanism. Symbols address detection, not recoverability. ● = detects. ◐ = partially detects (see notes). ○ = does not detect. L&A = *logic and accuracy testing*. BA/CoC = *ballot accounting and chain-of-custody reconciliation*. BP-RLA = *ballot-polling RLA*. BC-RLA = *ballot-comparison RLA*. Batch = *batch-comparison audit*. HR = *hand recount*.

	Failure mode	L&A	BA/CoC	BP-RLA	BC-RLA	Batch	HR
	Ballot printing/BMD config.	◐[a]	◐[b]	○	○	○	○
F1	BMD marking fidelity	○	○	○	○	○	◐[c]
	Curation of cast ballots	○	●	○	○	○	○
	Scanning	◐[a]	○	◐[d]	◐[d]	◐[d]	●
	Ballot definition (process)	◐[a]	○	●	●	●	●
F2	Ballot definition file	◐[e]	○	●	●	●	●
	Mark interpretation	◐[a]	○	●	●	◐[f]	●
	CVR integrity	○	○	○	●	◐[f]	●
	Vote accumulation	◐[g]	○	●	○	●	●
F3	Results artifact (memory)	○	●	◐[d]	◐[d]	◐[d]	●
	EMS aggregation	○	○	●	○	●	●

[a] Only what the test deck checks (e.g. not folded ballots, marginal marks, untested styles). [b] Reconciliation against participation records; omitted contests are unrecoverable. [c] Only if printed text and its encoding disagree. [d] Only failures large enough to alter outcome. [e] Only if re-run on every tabulator after the last definition change. [f] Whole batches only. [g] In-tabulator only; EMS out of scope.

8 Implications for Audits

The F1–F4 taxonomy provides a structured basis for evaluating which verification mechanisms can detect which failure classes. A recurring theme of the incident analysis is that no single verification mechanism covers the full pipeline. Meaningful post-election audits, therefore, must take the form of a layered system of checks whose combined coverage spans the first three failure categories.

8.1 What Each Verification Mechanism Covers

Table 5 maps six verification mechanisms against the F1–F3 sub-components of Sects. 4–6. L&A coverage is bounded by test deck design and fails under operational conditions the deck omits Recommendation 3. Ballot accounting and chain-of-custody reconciliation is the only mechanism that detects missing, duplicated, or never-uploaded artifacts; the audits assume its result rather than establishing it, which is why the F1 curation and F3 results-artifact rows depend on it Recommendation 4. Ballot-polling RLAs detect only outcome-altering failures [41]. Ballot-comparison RLAs check CVR integrity between tabulator and EMS, but are blind to errors in combining the votes as-read, where the CVRs faithfully match the ballots. Batch comparison detects only whole-batch failures. Hand recounts give the strongest F2–F3 coverage but are typically triggered only by margin thresholds or discovered discrepancies. This creates a paradox in jurisdictions where tabulators are used without RLAs, since the evidence needed to trigger a recount may be demonstrable only by a recount. Every Canadian jurisdiction using tabulators is caught in it, and none has signalled a move toward RLAs.

The F1 rows that matter most for the paper trail are uncovered. Votes never solicited (wrong ballot style) or never recorded (BMD unavailable) cannot be recovered from the ballots cast, and no paper-based mechanism detects a BMD printout that misstates the voter's expressed preference, since all of them treat the printed text as ground truth. Northampton County 2023 (Sect. 7.2) is the exception, and only because the text and the barcode disagreed Recommendations 5 and 6.

8.2 Recommendations

The following recommendations follow directly from our taxonomy, incident corpus, and verification coverage analysis in the preceding sections:

1. **Post-election audits should be universal and mandatory.** Our analysis demonstrates that failures have occurred across all pipeline stages, across all major vendors, and in jurisdictions with well-resourced election administrations. Importantly, the absence of a *detected* failure in a given election is not evidence that the outcome was correct.
2. **Audit design should be informed by the full failure taxonomy.** Jurisdictions should evaluate their audit procedures against each of F1–F3 explicitly and identify gaps in coverage.

3. **L&A test decks should be designed to exercise known failure modes.** Test decks should include folded ballots, ballots with marginal marks, and all vote combinations for each contest, including split-ticket and RCV patterns. As commonly practiced, L&A checks none of these, nor mark-density thresholds or sensor sensitivity.
4. **Inventory reconciliation should be completed before certification.** The Georgia 2020 memory card failures and the Monmouth County double-counting were detectable by reconciling precinct-level records and ballot counts against independent expected counts (poll books, VBM processing). This reconciliation should be a required pre-certification step in every jurisdiction, rather than left to ad hoc post-election discoveries.
5. **BMD deployments require explicit acknowledgment of fragility and audit limitations.** Jurisdictions using BMDs for all voters (especially all-in-one designs where the voter cannot inspect the printout) should explicitly recognize that (1) Election day failures may prevent voting altogether, so HMPB backup ballots are required, and (2) Audits cannot detect errors in recording votes originating in BMDs, so genuine RLAs are impossible. These facts should be publicly acknowledged.
6. **New verification mechanisms for BMD F1 failures are needed.** Ensuring BMD printouts reflect voters' expressed preferences is an open, evidently intractable problem [40].

9 Conclusion

Optical-scan elections have failed repeatedly without any adversary, in ways that are well documented. No stage, vendor, or jurisdiction is immune. The F1–F4 taxonomy organizes those failures by where in the pipeline they manifest. Hand-marked paper ballots kept secure and accounted for throughout make it possible to catch outcome-changing errors and correct wrong outcomes—that is, to have software independence. But the possibility of checking is not the same as *actually* checking. Every contest should be audited. Audits should be designed with the full failure taxonomy in mind. In low-trust settings, audits of a trustworthy paper trail answer unfounded claims of manipulation; in high-trust settings, they replace assumed reliability with demonstrated correctness.

Acknowledgements. This work was funded in part by NSERC Discovery Grant RGPIN-2026-06401 (Essex) and NSF grant SaTC–2228884 (Stark).

References

1. Antonyan, T., Nicolaou, N., Shvartsman, A.A., Smith, T.: Determining the causes of AccuVote optical scan voting terminal memory card failures. In: EVT/WOTE '10. USENIX Association (2010)
2. Appel, A.W.: Willful disregard of voter intent in Los Angeles. Center for Information Technology Policy Blog, Princeton University (2023). https://blog.citp.princeton.edu/2023/05/08/willful-disregard-of-voter-intent-in-los-angeles/
3. 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)
4. Appel, A., Stark, P.: Evidence-based elections: create a meaningful paper trail, then audit. Georgetown Law Technol. Rev. **4**, 523–541 (2020)
5. Bajcsy, A., Li-Baboud, Y.S., Brady, M.: Systematic measurement of marginal mark types on voting ballots. NIST Report 8069, National Institute of Standards and Technology (2015)
6. Barney-Smith, E.H., Nagy, G., Lopresti, D.: Mark detection from scanned ballots. In: Document Recognition and Retrieval XVI, vol. 7247, p. 72470P. SPIE (2009)
7. Bernhard, M., et al.: Can voters detect malicious manipulation of ballot marking devices? In: IEEE Symposium on Security and Privacy, pp. 679–694. IEEE (2020)
8. Butler, K.R.B., et al.: Systemic issues in the Hart InterCivic and Premier voting systems: reflections on Project EVEREST. In: EVT '08. USENIX Association (2008)
9. Chaum, D., et al.: Scantegrity: end-to-end voter-verifiable optical- scan voting. IEEE Secur. Priv. **6**(3), 40–46 (2008)
10. Crimmins, B.L., Narayanan, D.Y., Springall, D., Halderman, J.A.: DVSorder: Ballot randomization flaws threaten voter privacy. In: USENIX Security 24, pp. 6525–6541 (2024)
11. Davtyan, S., et al.: Pre-election testing and post-election audit of optical scan voting terminal memory cards. In: EVT '08. USENIX Association (2008)
12. Decaturish: Editorial: Release the results of the District 2 county commission race now. Decaturish (2022). https://www.decaturish.com/archives/editorial-release-the-results-of-the-district-2-county-commission-race-now/article_7c32ff8f-afcf-54ac-91fa-1bb0e83de028.html
13. DeKalb County Voter Registration and Elections: DeKalb board of registration and elections certifies primary elections, focuses on upcoming runoff contests. DeKalb County, Georgia (2022). https://www.dekalbcountyga.gov/news/dekalb-board-registration-and-elections-certifies-primary-elections-focuses-upcoming-runoff
14. Dzieduszycka-Suinat, S., et al.: The future of voting: end-to-end verifiable internet voting - specification and feasibility study. US Vote Foundation (2015). https://www.usvotefoundation.org/E2E-VIV
15. Fausset, R., Epstein, R., Rojas, R.: 'I refuse not to be heard': Georgia in uproar over voting meltdown (2020). https://www.nytimes.com/2020/06/09/us/politics/atlanta-voting-georgia-primary.html
16. Feldman, A.J., Halderman, J.A., Felten, E.W.: Security analysis of the Diebold AccuVote-TS voting machine. In: EVT '07. USENIX Association (2007)
17. Florida Department of State, Division of Elections: Election observation report: St. Lucie County, Florida, November 14–18, 2012. Tech. rep., Florida Secretary of State (2012). https://files.floridados.gov/media/693816/st-lucie-county-election-observation-report-nov-14-18-2012.pdf

18. Fowler, S.: State outlines fix for error that halted election testing (2020). https://www.gpb.org/news/2020/09/29/state-outlines-fix-for-error-halted-election-testing
19. Halderman, J.A.: The Antrim County 2020 election incident: an independent forensic investigation. In: USENIX Security 22, pp. 589–605. USENIX Association (2022)
20. Hernandez, J.: The 'human error' that's snarling the New York City mayor's race. NPR (2021). https://www.npr.org/2021/06/30/1011747612/the-human-error-thats-snarling-the-new-york-city-mayors-race
21. Horsely, M.: Clerical error stopped Democrats from voting in narrow primary, Jefferson clerk says (2026). https://kentuckylantern.com/2026/07/01/clerical-error-stopped-democrats-from-voting-in-narrow-primary-jefferson-clerk-says/
22. Hursti, H., Lindeman, M., Stark, P.B.: New Hampshire SB 43 forensic audit report. Tech. rep., State of New Hampshire (2021). https://mm.nh.gov/files/uploads/doj/remote-docs/20210713-sb43-forensic-audit-report.pdf
23. Hursti, H., Moss, J., DEF CON Voting Village: DEF CON 25 voting machine hacking village report on cyber vulnerabilities in U.S. election equipment. Tech. rep., DEF CON (2017). https://www.defcon.org/images/defcon-25/DEF%20CON%2025%20voting%20village%20report.pdf
24. Hursti, H., et al.: DEF CON 27 voting machine hacking village report. Tech. rep., DEF CON / Nordic Innovation Labs/Georgetown University (2019). https://media.defcon.org/DEF%20CON%2027/voting-village-report-defcon27.pdf
25. Jones, D.W.: On optical mark-sense scanning. In: Chaum, D., et al. (eds.) Towards Trustworthy Elections. LNCS, vol. 6000, pp. 175–190. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12980-3_10
26. Lindeman, M., Stark, P.B.: A gentle introduction to risk-limiting audits. IEEE Secur. Priv. **10**(5), 42–49 (2012)
27. MacDougall, I.: What went wrong at New York City polling places? It was something in the air. literally. ProPublica (2018). https://www.propublica.org/article/new-york-city-polling-places-midterms-2018-humidity
28. Mahmood, K., et al.: Busting the paper ballot: voting meets adversarial machine learning. In: ACM CCS 2025, pp. 3132–3146 (2025)
29. Maryland State Board of Elections: 2016 post election audits in Maryland. Presentation at NIST Voting Standards Workshop (2017). https://www.nist.gov/system/files/documents/2017/02/16/day1-2017-02-13-09_15-03-natasha-post_election_audits_final.pdf
30. McCullough, C.: Ballot scanning errors caused voting delays in one Pennsylvania county, but no evidence of a 'steal'. PolitiFact (2024). https://www.politifact.com/factchecks/2024/nov/05/tweets/ballot-scanning-errors-caused-voting-delays-in-one/
31. McGregor, R.V.: Maricopa County 2022 general election ballot-on-demand printer investigation. Tech. rep., Independent investigation (2023). https://www.maricopa.gov/DocumentCenter/View/82599/Maricopa-Election-Report-and-Attachments
32. Minnesota Office of the Secretary of State: Minnesota's historic 2008 election. Tech. rep., Minnesota Office of the Secretary of State (2009). https://www.sos.mn.gov/media/3078/minnesotas-historic-2008-election.pdf
33. National Institute of Standards and Technology: Developing an analysis of threats to voting systems: Preliminary workshop summary. Tech. rep., National Institute of Standards and Technology (2005). https://www.nist.gov/system/files/documents/itl/vote/threatworksummary.pdf, workshop held October 7, 2005, Gaithersburg, Maryland

34. Norden, L.: Voting system failures: a database solution. Tech. rep., Brennan Center for Justice, New York University School of Law (2010). https://www.brennancenter.org/our-work/research-reports/voting-system-failures-database-solution
35. North Carolina State Board of Elections: Elections officials respond to humidity issues. NCSBE (2018), https://www.ncsbe.gov/news/press-releases/2018/11/06/elections-officials-respond-humidity-issues
36. Rivest, R.L.: On the notion of "software independence" in voting systems. Phil. Trans. R. Soc. A **366**(1881), 3759–3767 (2008)
37. Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt À Voter: a Voter-Verifiable Voting System. IEEE Trans. Inf. Forensics Secur. **4**(4), 662–673 (2009)
38. Shore News Network: Software malfunction led to miscount that declared wrong winner in NJ election. Shore News Network (2023), https://shorenewsnetwork.com/?p=837866
39. Stark, P., Wagner, D.: Evidence-based elections. IEEE Secur. Priv. **10**, 33–41 (2012)
40. Stark, P., Xie, R.: They may look and look, yet not see: BMDs cannot be tested adequately. In: Krimmer, R., et al. (eds.). E-Vote-ID 2022. LNCS, vol. 13553, pp. 122–138. Springer (2022)
41. Stark, P.B.: Super-simple simultaneous single-ballot risk-limiting audits. In: EVT/WOTE '10. USENIX Association, (2010)
42. Stark, P.B.: When audits and recounts distract from election integrity: The 2020 U.S. presidential election in Georgia. In: E-Vote-ID 2024. Gesellschaft für Informatik (2024)
43. Stone, J.: Alameda Co. finds error in ranked-choice voting system, investigating Oakland school board race. https://abc7news.com/post/ranked-choice-voting-oakland-school-board-director-district-4-race-mike-hutchinson-alameda-county-registrar-of-voters/12626221/ (2022)
44. Sullivan, S.: Lack of safeguards led to double-counted votes in NJ county, AG probe finds. NJ.com (2023), https://www.nj.com/monmouth/2023/09/lack-of-safeguards-led-to-double-counted-votes-in-nj-county-ag-probe-finds.html
45. United Nations: International covenant on civil and political rights. General Comment No. 25: Participation in public affairs and the right to vote. UN Office on Human Rights (1996), https://digitallibrary.un.org/record/221930?v=pdf
46. U.S. Election Assistance Commission: Election operations assessment: Threat trees and matrices and threat instance risk analyzer (TIRA). Tech. rep., U.S. Election Assistance Commission (2009), https://www.eac.gov/sites/default/files/eac_assets/1/28/Election_Operations_Assessment_Threat_Trees_and_Matrices_and_Threat_Instance_Risk_Analyzer_(TIRA).pdf
47. U.S. Election Assistance Commission: Letter to ES&S re: ExpressVote 1.0 trusted build hash validation. EAC (2020), https://www.eac.gov/sites/default/files/document_library/files/ESS%20Bitmap%20Records%20Redacted.pdf
48. U.S. Election Assistance Commission: Voluntary voting system guidelines version 2.0 (VVSG 2.0). Tech. rep., U.S. Election Assistance Commission (2021), adopted February 2021
49. U.S. Election Assistance Commission: Voluntary voting system guidelines version 2.0 (VVSG 2.0), test assertions version 1.2. Tech. rep., U.S. Election Assistance Commission (2021)

50. Verified Voting: The verifier: Polling place equipment map. Verified Voting (2026), https://verifiedvoting.org/verifier/#mode/navigate/map/voteEquip/mapType/ppEquip/year/2026
51. Wagner, D., Bishop, M., et al.: California secretary of state top-to-bottom review of voting systems. Tech. rep, California Secretary of State (2007)
52. Walker, C.: Voting machines in Northampton County printed ballots with errors in two judicial races. Spotlight PA / Votebeat (2023), https://www.spotlightpa.org/news/2023/11/pennsylvania-election-2023-northampton-county-voting-machine-ballot-issue/
53. Walker, J., Bajaj, N., Crimmins, B.L., Halderman, J.A.: Logic and accuracy testing: A fifty-state review. In: E-Vote-ID 2022. LNCS, vol. 13553, pp. 157–184. Springer (2022)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Risk-Limiting Audits for Parliamentary Majorities

Jack Freestone[1], Dennis Leung[2], and Damjan Vukcevic[3(✉)]

[1] School of Mathematical and Physical Sciences, Macquarie University, Sydney, Australia
[2] School of Mathematics and Statistics, University of Melbourne, Melbourne, Australia
[3] Department of Econometrics and Business Statistics, Monash University, Melbourne, Australia
damjan.vukcevic@monash.edu

Abstract. Existing methods for risk-limiting audits typically focus on certifying individual contests. In parliamentary elections, however, the politically relevant outcome is often whether a party has won enough seats to form government, not whether every reported seat outcome is correct. Extending on the work of Mohanty et al. [9], we formulate the certification of a parliamentary majority as a partial conjunction testing problem: it is enough to verify that the reported winning party truly won at least a majority of its reported seats. Building on the SHANGRLA auditing framework, we construct a sequential audit statistic for the majority outcome by combining seat-level statistics. We then propose adaptive sampling strategies that allocate auditing effort across seats, including variants that learn to avoid spending excessive effort on seats that appear unlikely to have been truly won. Using simulations based on synthetic and real data, from the 2014 Indian Lok Sabha election, we show that auditing the parliamentary majority can substantially reduce the number of ballots inspected (by almost a thousand-fold) compared to certifying every reported winning seat.

1 Introduction

A *risk-limiting audit* (RLA) is a post-election procedure that sequentially samples ballots, accumulating statistical evidence until either the reported outcome can be certified[1] or the audit escalates to a full hand count [8]. Its guarantee is that if the reported outcome is wrong, the audit certifies it with probability at most α (the *risk limit*), a pre-specified value. If the reported outcome is correct, certification typically needs only a small fraction of the ballots. The SHANGRLA

Authors listed alphabetically. This work was supported by the Australian Research Council (OPTIMA ITTC IC200100009).

[1] We use 'certify' (and 'certification') throughout in a purely *statistical* sense: to verify the result of the election with risk limit α. This is distinct from *certification* as a *legal* act, whose meaning vary across jurisdictions.

© The Author(s) 2027
D. Duenas-Cid et al. (Eds.): E-Vote-ID 2026, LNCS 17083, pp. 55–69, 2027.
https://doi.org/10.1007/978-3-032-39512-2_4

framework of Stark [13] reduces the auditing of any electoral system (a *social choice function*) to testing a finite collection of *assertions* about population means; the ALPHA *test supermartingale* [14] provides a flexible sequential test for each. These tools, however, were developed primarily for single-contest elections. In a parliamentary system, government is determined not by any single race but by whether a party controls a majority of seats, a joint outcome whose auditing has received comparatively little attention.

As first observed by Mohanty et al. [9], certifying a parliamentary majority does not require certifying every individual seat. We formalise this as a *partial conjunction* hypothesis test [1]: a procedure that rejects at least some (but not necessarily all) null hypotheses from a large collection. This pools evidence across seats, certifying the majority even when some seats have not individually accumulated enough evidence, and can substantially reduce the number of ballots that must be hand-inspected.

We develop a suite of adaptive sampling schemes that dynamically allocate auditing effort across seats. These include *greedy* strategies that concentrate sampling on weaker seats (where stronger evidence is required) and *filtering* strategies that redirect effort away from seats where the reported outcome seems to be false. We evaluate our schemes via simulation across a range of parliamentary configurations, including scenarios based on the 2014 Indian Lok Sabha election, and show they require observing orders of magnitude fewer ballots than audits that aim to certify all reported winning seats.

2 RLA for a Parliamentary Majority by Sequentially Testing a Partial conjunction Hypothesis

Let $\mathcal{S}$ denote the set of all seats in a parliament, each corresponding to a separate contest. As a concrete example, the Lok Sabha (Lower House) of the Indian Parliament has $|\mathcal{S}| = 543$ seats corresponding to 543 constituencies [9]. Whichever party controls the majority, that is,

$$r \equiv \lfloor |\mathcal{S}|/2 \rfloor + 1$$

of the $|\mathcal{S}|$ seats, can form the government for the next term. Suppose $\mathcal{W} \subset \mathcal{S}$ is the subset of seats reportedly won by the winning party after a given election (necessarily $|\mathcal{W}| \geqslant r$). To confirm that the new government can be rightfully formed, we need to verify that at least r of the seats in $\mathcal{W}$ have been truly won by the winning party, which can be formulated as testing the null hypothesis

$H_{r/\mathcal{W}}$: fewer than r seats in $\mathcal{W}$ have been truly won by the winning party.

This is the complementary null of the assertion that the winning party won at least r seats. If $H_{r/\mathcal{W}}$ is rejected, the majority outcome is considered certified.

To conduct an audit with risk limit α, we can sample the ballots cast for the seats in $\mathcal{W}$ sequentially, until their revealed evidence suggests that $H_{r/\mathcal{W}}$ can be rejected at level α. We conceptualise such sampling as happening iteratively

in 'rounds', indexed by a time variable t. Before any sampling has taken place we have $t = 0$, and at each subsequent time $t \in \{1, 2, \dots\}$, the auditor has the option to draw one cast ballot for each seat $s \in \mathcal{W}$ without replacement.[2] Let

$$D_{s,t+1} \equiv \begin{cases} 1, & \text{if a ballot for seat } s \text{ is drawn at time } t+1, \\ 0, & \text{otherwise,} \end{cases}$$

be the auditor's sampling decision for time $t+1$ for seat s. The decision $D_{s,t+1}$ can therefore only depend on the information from the ballots sampled up to time t; we denote this information as $\mathcal{F}_t$.

To evaluate the evidence against a given null hypothesis, H, we need to define a suitable test statistic, U_t. We will calculate this iteratively based on the information revealed in $\mathcal{F}_t$, giving rise to a sequence of values, $U \equiv (U_t)_{t \geqslant 0}$, referred to as a *(test) process*. We want processes that satisfy the following:

1. We say that U is *anytime-valid for testing* H if

$$\Pr\left(\sup_{t \geqslant 0} U_t \geqslant 1/\alpha\right) \leqslant \alpha \text{ for all } \alpha \in (0,1) \text{ when } H \text{ is true.} \tag{1}$$

   This allows us to reject H, with risk limit α, once U exceeds $1/\alpha$.
2. If H is false, we want to reject it efficiently. We seek a process U that will grow rapidly in response to new observations, so it can quickly exceed $1/\alpha$.

In the remainder of this section we construct a process ($E_{r/\mathcal{W}}$ in Theorem 1) that is anytime-valid for testing $H_{r/\mathcal{W}}$. Then in Sect. 3 we explore various sampling schemes that guide the drawing decisions $D_{s,t}$ across time, aiming to minimize the total number of ballots sampled before $H_{r/\mathcal{W}}$ can be rejected.

2.1 A Test Process for Certifying a Single Seat

As a starting point, we define for each seat $s \in \mathcal{W}$ the null hypothesis

$$H_s\text{: the winning party has } \textit{not} \text{ truly won the seat } s,$$

and let N_s be the total number of ballots cast for seat s, which we label $b_{s,1}, \dots, b_{s,N_s}$. We assume the social choice function for this seat can be decomposed into a finite set of J_s *assertions* using the SHANGRLA framework [13], i.e., there exist J_s non-negative *assorter* functions $\{A_{s,j}\}_{j=1}^{J_s}$ such that, by letting $x_{s,j,i} \equiv A_{s,j}(b_{s,i})$, H_s is false if and only if

$$\bar{x}_{s,j} \equiv \frac{1}{N_s} \sum_{i=1}^{N_s} x_{s,j,i} > 1/2 \quad \text{for all } j = 1, \dots, J_s. \tag{2}$$

[2] This choice is primarily for clear exposition. In practice, a real audit would aim to have very few rounds, and would draw a much larger sample of ballots in each round. The methods we develop here are applicable to any such sampling scheme.

We can therefore write

$$H_s = \bigcup_{j=1}^{J_s} H_{s,j}, \tag{3}$$

where

$$H_{s,j} \colon \bar{x}_{s,j} \leqslant 1/2$$

is the null hypothesis that the jth assertion is *not* true. As an example, consider a seat contested by just two candidates, Alice (the reported winner) and Bob. Here there is a single assertion ($J_s = 1$), whose assorter $A_{s,1}$ scores a ballot for Alice as 1, a ballot for Bob as 0, and any other (e.g. invalid) ballot as 1/2. The assertion mean $\bar{x}_{s,1}$ is then the average of these scores across all N_s ballots, which exceeds 1/2 if and only if Alice received more votes than Bob.

Suppose $(B_{s,i})_{i=1}^{N_s}$ represents the full sequence of ballots that would be drawn randomly without replacement for seat s, which is necessarily a random permutation of $\{b_{s,i}\}_{i=1}^{N_s}$. Let $(X_{s,j,i})_{i=1}^{N_s} = (A_{s,j}(B_{s,i}))_{i=1}^{N_s}$ represent the corresponding (random) sequence of values from $\{x_{s,j,i}\}_{i=1}^{N_s}$. Since a ballot for seat s isn't necessarily drawn in every round, one should keep in mind that $B_{s,i}$ is only the ith ballot drawn for seat s and its sampling will take place at a time $t \geqslant i$.

Let $n_s(t)$ denote the number of ballots drawn for seat s up to and inclusive of time t. Define the process $M_{s,j} \equiv (M_{s,j,t})_{t \geqslant 0}$ by letting $M_{s,j,0} \equiv 1$ and

$$M_{s,j,t} \equiv \prod_{i=1}^{n_s(t)} (1 + \lambda_{s,j,i}(X_{s,j,i} - \mu_{s,j,i})) \quad \text{for all } t = 1, 2, \ldots, \tag{4}$$

where $\mu_{s,j,i} \equiv \frac{N_s/2 - \sum_{k=1}^{i-1} X_{s,j,k}}{N_s - i + 1}$ is the hypothesised mean of remaining assorter values just before the ith ballot is drawn assuming $\bar{x}_{s,j} = 1/2$ (the boundary value for $H_{s,j}$ to hold), and $\lambda_{s,j,i} \in (0, \mu_{s,j,i}^{-1})$ is a parameter depending only on $\mathcal{F}_{T_s(i)-1}$, where $T_s(i) \equiv \min\{t : n_s(t) = i\}$ is the first time that i ballots have been sampled for seat s. Each $\lambda_{s,j,i}$ controls how aggressively the process responds to the value $X_{s,j,i} - \mu_{s,j,i}$, and can be set based on the information available from all the revealed ballots before the ith ballot is drawn for seat s. Larger values of $\lambda_{s,j,i}$ produce faster growth of $M_{s,j,t}$ when $H_{s,j}$ is false.

Note that $M_{s,j}$ is an example of a *betting process* [18], introduced for election auditing by the ALPHA process of Stark [14]. Unlike the original ALPHA, $M_{s,j}$ has the variation that its value may not update at every time step because sampling doesn't necessarily occur for seat s at all t. Alternatively, we can write

$$M_{s,j,t} = M_{s,j,t-1} \cdot \left\{ D_{s,t} \cdot \left(1 + \lambda_{s,j,n_s(t)}(X_{s,j,n_s(t)} - \mu_{s,j,n_s(t)})\right) + 1 - D_{s,t} \right\} \tag{5}$$

which gives $M_{s,j,t} = M_{s,j,t-1}$ when $D_{s,t} = 0$ and elucidates how the betting process evolves from $t-1$ to t.

Importantly, $M_{s,j}$ is a *test supermartingale* under $H_{s,j}$. Specifically, it is non-negative ($M_{s,j,t} \geqslant 0$ for all $t \geqslant 0$), has initial value 1 ($M_{s,j,0} = 1$), and

$$\mathbb{E}[M_{s,j,t} \mid \mathcal{F}_{t-1}] \leqslant M_{s,j,t-1} \tag{6}$$

when $H_{s,j}$ is true. We establish these facts in Appendix A. Ville's inequality [15] then implies that $M_{s,j}$ is anytime-valid for testing $H_{s,j}$.

To test the seat as whole, we combine these together into a *seat-level* process

$$E_{s,t} \equiv \min_{1 \leqslant j \leqslant J_s} M_{s,j,t}. \tag{7}$$

By virtue of the 'min' construction, this is anytime-valid for testing H_s (see Theorem 1, below). However, our goal is not test each seat on its own, but to use these processes as a building block for testing more complex hypotheses.

2.2 A Test Process for Certifying a Parliamentary Majority

Let $\mathcal{C} \subset \mathcal{W}$ be a subset of the reported winning seats, and consider

$$H_{1/\mathcal{C}} \equiv \bigcap_{s \in \mathcal{C}} H_s, \tag{8}$$

the *intersection null hypothesis* that H_s is true for all seats in $\mathcal{C}$. Note that $H_{1/\mathcal{C}}$ is true when *no* (< 1) seats in $\mathcal{C}$ are truly won by the winning party, and hence its notational similarity to $H_{r/\mathcal{W}}$. Now define

$$E_{1/\mathcal{C},t} \equiv \prod_{s \in \mathcal{C}} E_{s,t}. \tag{9}$$

This provides a stepping-stone to a process for testing $H_{r/\mathcal{W}}$, because we can write that hypothesis as the following union

$$H_{r/\mathcal{W}} = \bigcup_{\substack{\mathcal{C} \subset \mathcal{W}: \\ |\mathcal{C}| = |\mathcal{W}| - r + 1}} H_{1/\mathcal{C}}. \tag{10}$$

Our test process for a parliamentary majority can now be defined as follows:

$$E_{r/\mathcal{W},t} \equiv \prod_{s=1}^{|\mathcal{W}|-r+1} E_{(s),t} = \min\{E_{1/\mathcal{C},t} : |\mathcal{C}| = |\mathcal{W}| - r + 1\} \tag{11}$$

where $E_{(1),t} \leqslant \cdots \leqslant E_{(|\mathcal{W}|),t}$ are the *order statistics* of $\{E_{s,t}\}_{s \in \mathcal{W}}$ at time t.

In other words, $E_{r/\mathcal{W},t}$ is the product of the smallest $|\mathcal{W}| - r + 1$ values among $\{E_{s,t}\}_{s \in \mathcal{W}}$, and the second equality in (11) follows naturally. It is easy to check that all of $E_{s,t}$, $E_{1/\mathcal{C},t}$ and $E_{r/\mathcal{W},t}$ are computable from $\mathcal{F}_t$. The following theorem proven in Appendix B states that they indeed define processes that are anytime-valid for testing H_s, $H_{1/\mathcal{C}}$ and $H_{r/\mathcal{W}}$.

Theorem 1. *The processes* $E_s \equiv (E_{s,t})_{t \geqslant 0}$, $E_{1/\mathcal{C}} \equiv (E_{1/\mathcal{C},t})_{t \geqslant 0}$ *and* $E_{r/\mathcal{W}} \equiv (E_{r/\mathcal{W},t})_{t \geqslant 0}$ *are anytime-valid for testing* H_s, $H_{1/\mathcal{C}}$ *and* $H_{r/\mathcal{W}}$, *respectively.*

Intuitively, using $E_{r/\mathcal{W}}$ to test $H_{r/\mathcal{W}}$ makes sense: In light of the union representation (10), $H_{r/\mathcal{W}}$ is false if and only if all intersection nulls $H_{1/\mathcal{C}}$ with $|\mathcal{C}| = |\mathcal{W}| - r + 1$ are false; the 'min' representation in (11) says that $E_{r/\mathcal{W},t}$ is large enough to reject $H_{r/\mathcal{W}}$ only when, for all $\mathcal{C}$ with cardinality $|\mathcal{W}| - r + 1$, the value of $E_{1/\mathcal{C},t}$ is large enough to reject the corresponding hypothesis $H_{1/\mathcal{C}}$.

2.3 Related Works

In the statistics literature, $H_{r/\mathcal{W}}$ is a *partial conjunction* hypothesis, testing whether at least r of the seat-level nulls $\{H_s\}_{s\in\mathcal{W}}$ are false. The notion is due to Benjamini and Heller [1] in the context of *replicability analysis*; Bogomolov and Heller [2] provide a recent survey. Our construction sits in the *e-value* and *e-process* framework [11,16]: each of $E_{s,t}$, $E_{1/\mathcal{C},t}$ and $E_{r/\mathcal{W},t}$ is an e-process for its null, and combining the seat-level ones into $E_{r/\mathcal{W},t}$ is a sequential Benjamini–Heller-type construction [6,7]. As shown in Appendix B, it stays valid even when sampling induces cross-seat dependence, extending Vovk and Wang [17].

Mohanty et al. [9, Section 4] first proposed certifying the overall winning party rather than every seat, for a more efficient audit, and sketched a test of $H_{r/\mathcal{W}}$, but did not implement nor evaluate it. Like us, they built an anytime-valid $E_{s,t}$ for each H_s, but then transformed it into a p-value $P_{s,t} \equiv 1/E_{s,t}$ and combined these across each subset $\mathcal{C} \subset \mathcal{W}$ by Fisher's method [5]. This is the stratified union-intersection idea of SUITE [10], applied across seats rather than across strata of a single contest. Working directly with the e-processes instead allows us to use arbitrary sampling schemes $D_{s,t}$ with easy theoretical justification, including the adaptive schemes in Sect. 3 which induce dependence across s, and is uniformly more efficient than the p-value approach; see Appendix C.

A final pair of RLA methods also test a union-of-intersections null, but within a *single* contest. AWAIRE [3] audits a single instant-runoff election. Its component tests share the same population of ballots, so they are not conditionally independent and must be combined by *averaging* rather than as a product. In

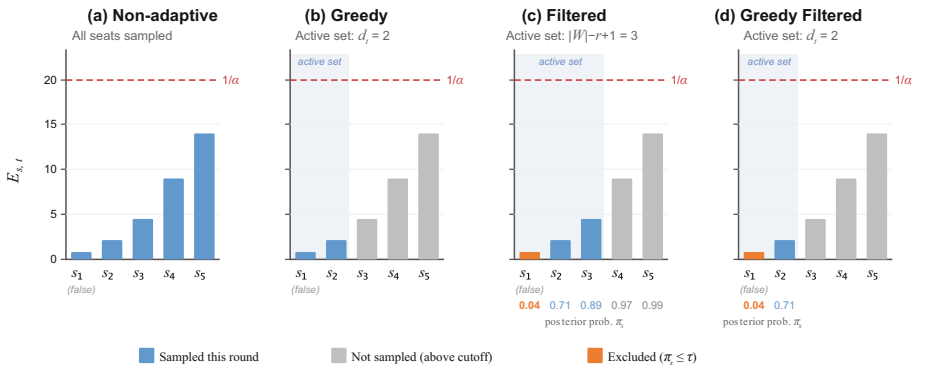


Fig. 1. Illustration of the four sampling schemes at a single round, with $|\mathcal{W}| = 5$ seats and $r = 3$. Bars show the current values of $E_{s,t}$; the dashed red line is $1/\alpha$. Blue bars indicate seats sampled in the next round; grey bars indicate seats above the active-set cutoff; orange bars indicate seats excluded by the posterior filter ($\pi_s \leqslant \tau$). Seat s_1 is falsely reported. (a) Non-adaptive: every seat is sampled. (b) Greedy: only the d_t weakest seats are sampled. (c) Filtered: only seats among the $|\mathcal{W}| - r + 1$ weakest with $\pi_s > \tau$ are sampled. (d) Greedy Filtered: combines the d_t active-set window of the Greedy scheme with the $\pi_s > \tau$ filter of the Filtered scheme.

our setting, since each seat is a separate contest, their e-processes are conditionally independent and can be multiplied. Spertus et al. [12] stratify a single contest and, like us, combine by products. They pool evidence about one contest-wide margin across strata. In contrast, we pool evidence about many separate contests, of which only r need to be verified.

3 Adaptive Sampling Schemes

Given how $E_{r/\mathcal{W},t}$ is defined in (11), the ballot drawing decisions $D_{s,t}$ should ideally be made so that the product of the smallest $|\mathcal{W}| - r + 1$ values among $\{E_{s,t}\}_{s\in\mathcal{W}}$ can exceed $1/\alpha$ with minimal sampling. We now describe several schemes for adaptively determining $D_{s,t}$. These are illustrated in Fig. 1.

Non-adaptive. This scheme sets $D_{s,t} = 1$ for all $s \in \mathcal{W}$ and all t. The auditor makes no attempt to reduce the number of ballots to be sampled.

Greedy. Since the winning party can be immediately certified at time t as long as the product of the smallest $|\mathcal{W}| - r + 1$ values among $\{E_{s,t}\}_{s\in\mathcal{W}}$ exceeds $1/\alpha$, a natural heuristic is to 'greedily' sample ballots for the seats with the lowest seat-level processes, in the hope that the additional ballots will increase these values. Let $d_t \in \{1, \ldots, |\mathcal{W}|\}$ be a cutoff sample size. The Greedy scheme sets

$$D_{s,t+1} = \begin{cases} 1, & \text{if } E_{s,t} \leqslant E_{(d_t),t}, \\ 0, & \text{if } E_{s,t} > E_{(d_t),t}, \end{cases}$$

where $E_{(1),t} \leqslant \cdots \leqslant E_{(|\mathcal{W}|),t}$ are the order statistics of $\{E_{s,t}\}_{s\in\mathcal{W}}$ at time t.

The question is how to choose d_t. Define

$$\delta_t = \max\left\{ i : i \in \{1, \ldots, r\} \text{ and } \prod_{s=i}^{|\mathcal{W}|-r+i} E_{(s),t} < 1/\alpha \right\}, \tag{12}$$

with the convention that $\delta_t = 0$ if the maximum is taken over an empty set (meaning the winning party can already be certified at time t). Let

$$d_t = \begin{cases} |\mathcal{W}|, & \text{if } \delta_t = r, \\ \delta_t + a, & \text{if } \delta_t < r, \end{cases} \tag{13}$$

for a constant $a \in \{0, 1, \ldots, |\mathcal{W}| - r\}$. When $\delta_t = r$, even the product of the largest $|\mathcal{W}| - r + 1$ elements among $\{E_{s,t}\}_{s\in\mathcal{W}}$ is below $1/\alpha$, so we sample all $|\mathcal{W}|$ seats. When $\delta_t < r$, the order statistics $E_{(1),t}, \ldots, E_{(\delta_t),t}$ are precisely those for which the product of any consecutive set of $|\mathcal{W}| - r + 1$ containing them would not exceed $1/\alpha$, and at a minimum we should gather more evidence for the δ_t seats corresponding to these processes. The parameter a controls how aggressively the

scheme minimises per-round sampling: $a = 0$ is the most aggressive, while larger values of a hedge by sampling additional seats beyond the strict minimum.

Since sampling is without replacement, it is possible for every one of the d_t selected weakest seats to have already exhausted its ballot supply (i.e., all N_s ballots have been drawn) without achieving certification ($E_{r/\mathcal{W},t} \geqslant 1/\alpha$). In this case, the round would issue no new samples and the audit would stall. To avoid this, we apply the following fallback: if all d_t weakest seats are exhausted, we incrementally increase d_t by one—extending the selection to the next-weakest seat—and repeat until either at least one selected seat has remaining ballots or $d_t = |\mathcal{W}|$. This guarantees that in each round the sampling effort is directed at the weakest seats whose ballot supply has not yet been exhausted.

Filtered. The Greedy scheme concentrates all sampling effort on the seats with the weakest seat-level processes. While intuitive, this can fail when some of the weakest seats correspond to *falsely reported* outcomes, i.e., seats where the winning party did not actually prevail. For such a seat s, the process E_s won't grow, which will lead to $E_{r/\mathcal{W}}$ stalling or growing very slowly.

We define a Bayesian inference approach that aims to filter out such seats. For each seat $s \in \mathcal{W}$, we monitor the observed ballots and update a posterior distribution on the true ballot-type proportions. After each round, we use this to calculate a posterior probability, π_s, that we use to exclude unpromising seats from further sampling.

Our approach uses a canonical Dirichlet–multinomial conjugate setup. We omit most of the details (they are standard) and just sketch out the main steps.

For seat s with L_s ballot types, set a prior on the ballot-type probabilities $\mathbf{p}_s = (p_{s,1}, \ldots, p_{s,L_s}) \sim \text{Dirichlet}(\boldsymbol{\alpha}_{s,0})$, with hyperparameters $\boldsymbol{\alpha}_{s,0} \equiv \hat{\mathbf{p}}_{s,0}\, \kappa_{s,0}$ based on an initial value $\hat{\mathbf{p}}_{s,0}$ and concentration parameter $\kappa_{s,0}$. After observing $n_s(t)$ ballots at time t, use conjugacy[3] to give the posterior $\mathbf{p}_s \mid \mathcal{F}_t \sim \text{Dirichlet}(\boldsymbol{\alpha}_{s,n_s(t)})$. Let $\theta_{s,j}$ be the true assorter mean for assertion j. The posterior on $\mathbf{p}_s$ induces a posterior on $\theta_{s,j}$ via its assorter function. The joint posterior on $\theta_{s,j}$ across all j could be used to calculate the posterior probability that seat s was truly won ($\theta_{s,j} > 1/2$ for all j). For convenience, and for a fixed $\epsilon \geqslant 0$, we instead define

$$\pi_s \equiv \min_{1 \leqslant j \leqslant J_s} \Pr(\theta_{s,j} > 1/2 + \epsilon \mid \mathcal{F}_t), \tag{14}$$

so that π_s is small when at least one assertion has substantial posterior evidence that its margin does not exceed ϵ. We investigate only $\epsilon = 0$, which deprioritises seats that appear falsely reported. A positive ϵ would additionally deprioritise seats won by too small a margin to certify feasibly. (Determining an efficient, adaptive choice of ϵ is left for future work.) We calculate each $\Pr(\theta_{s,j} > 1/2 + \epsilon \mid$

[3] The conjugacy relies on the sampled ballots being treated as multinomial, an approximation since sampling is without replacement. This model is used only to choose where the filter samples next, which is a predictable decision, so by Theorem 1 the risk limit holds regardless of how accurate the model is.

$\mathcal{F}_t$) via a normal approximation to the posterior, or an exact calculation in the case of $L_s = 2$ (two ballot types).

The Filtered scheme restricts attention to the $|\mathcal{W}| - r + 1$ seats with the smallest current seat-level processes $E_{s,t}$ since only these seats enter the product defining $E_{r/\mathcal{W},t}$ in (11). Among this active set, it samples those whose posterior probability exceeds a threshold τ:

$$D_{s,t+1} = \begin{cases} 1, & \text{if } E_{s,t} \leqslant E_{(|\mathcal{W}|-r+1),t} \text{ and } \pi_s > \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

Choosing τ small (we set $\tau = 0.01$) makes the filter liberal: a seat is excluded only when there is substantial evidence that it was *not* truly won. If no seat in the active set satisfies both $\pi_s > \tau$ and has remaining ballots, the $\pi_s > \tau$ condition is dropped and $D_{s,t+1} = 1$ is set for every seat in the active set that still has ballots to draw.

Greedy Filtered. The Greedy scheme is efficient in benign cases (no falsely reported seats), while the Filtered scheme is more robust when the audit encounters false seats. We combine them together by using both the adaptive window d_t (the 'active set') and the $\pi_s > \tau$ selection filter.

To ensure we always draw at least one ballot, we use the following fallback rules: (1) The active set is expanded one seat at a time—adding the next-weakest seat—first to ensure it contains a seat with remaining ballots, and, if necessary, further (up to a maximum size of $|\mathcal{W}| - r + 1$) to ensure it contains a seat that also satisfies $\pi_s > \tau$. (2) Similar to the fallback of the Filter method, if the above rule selects no seats, the $\pi_s > \tau$ filter is dropped and the scheme sets $D_{s,t+1} = 1$ for every seat in that window with remaining ballots.

4 Results

We evaluate the sampling schemes on plurality elections audited by ballot-polling[4] without replacement. Throughout, we assume there are no invalid ballots[5] and set a risk limit of $\alpha = 0.05$. Section 4.1 shows results from controlled two-candidate simulations (with further results in Appendix D), while in Sect. 4.2 we simulate audits based on the 2014 Indian parliamentary election. Code and data to reproduce our results are available at: https://github.com/freejstone/RLA_parliamentary_majorities_code.

[4] Ballot-polling audits require only the paper trail and the reported outcome, so can be used for any election with a paper trail. This is the simplest scenario and the only one we investigate here. More efficient audits are possible if electronic records of the ballots are available; only some jurisdictions would have these.

[5] This is purely for simplicity. The methods can cope with invalid ballots: the assorters would score these as 1/2 (see Sect. 2.1).

Audit Strategies. We compare six strategies for certifying the parliamentary majority (testing $H_{r/\mathcal{W}}$): Non-adaptive; Greedy with $a = 0$ or $a = 3$ in (13); Filtered with $\tau = 0.01$; and Greedy Filtered with $a = 0$ or $a = 3$, and $\tau = 0.01$.

For comparison, we include two further strategies to act as benchmarks: (1) *Top-r seats* is a two-phase rule that first samples from $\mathcal{R}$, the r winning seats with the largest reported seat margins, switching to the remaining $|\mathcal{W}| - r$ winning seats if the first r seats are exhausted. During the first phase, it effectively attempts to certify all seats in $\mathcal{R}$, i.e., to reject $H_{r/\mathcal{R}}$. (2) *All seats* aims to certify all reported winning seats, rather than just a majority. It uses the same sampling rule as *Non-adaptive*, but sets $r = |\mathcal{W}|$, giving it the stricter target of rejecting $H_{|\mathcal{W}|/\mathcal{W}}$.

Tuning Parameters. For testing each assertion, we use $M_{s,j}$ from (4) and the truncated-shrinkage strategy of ALPHA [14] to set $\lambda_{s,j,i}$. This uses tuning parameters d and η_0, which we set to $d = 200$ and $\eta_0 = 0.51$ as per Ek et al. [4].

For the Dirichlet prior used in the filtering strategies, we set the hyperparameters such that the prior mean for each assertion was 0.51 and $\kappa_{s,0} = 200$ (for plurality voting this involves solving a set of linear equations). This was to mimic the role of η_0 and d in the ALPHA truncated-shrinkage estimator.

4.1 Simulated Plurality Contests

We first evaluated the audit strategies using synthetic two-candidate plurality contests. Consider a parliament with $|\mathcal{S}| = 100$ seats, so a majority requires $r = \lfloor |\mathcal{S}|/2 \rfloor + 1 = 51$ seats. We fixed the number of reported winning seats at $|\mathcal{W}| = 60$ and each seat to have $N_s = 5000$ ballots.

For each seat s, we set a target proportion of ballots for the reported winner, $p_{\text{target},s}$. The reported winning candidate receives $\lfloor N_s \times p_{\text{target},s} \rfloor$ ballots and the remaining ballots are assigned to the opposing candidate.

We allowed some reported winning seats to be false. If $n_{\text{false}} > 0$, then n_{false} of the seats in $\mathcal{W}$ were generated with $p_{\text{target},s} = 0.48$, so that the reported winner did not truly win that seat. The remaining $|\mathcal{W}| - n_{\text{false}}$ seats are correctly reported; we considered two scenarios for setting $p_{\text{target},s}$ for these seats:

1. (Homogeneous margins) All truly won seats share the same target vote share $p_{\text{target},s} = p_{\text{target}}$. We varied $p_{\text{target}} \in \{0.52, 0.55, 0.60\}$.
2. (Heterogeneous margins) The target vote share for each truly won seat was drawn independently as

$$p_{\text{target},s} \sim \text{Beta}(\bar{p}_{\text{target}}\kappa_{\text{het}}, (1 - \bar{p}_{\text{target}})\kappa_{\text{het}}),$$

and then truncated to lie in $[0.51, 0.999]$, ensuring that the reported winner truly wins the seat. We fixed $\kappa_{\text{het}} = 30$, corresponding to moderate heterogeneity, and varied $\bar{p}_{\text{target}} \in \{0.52, 0.55, 0.60\}$.

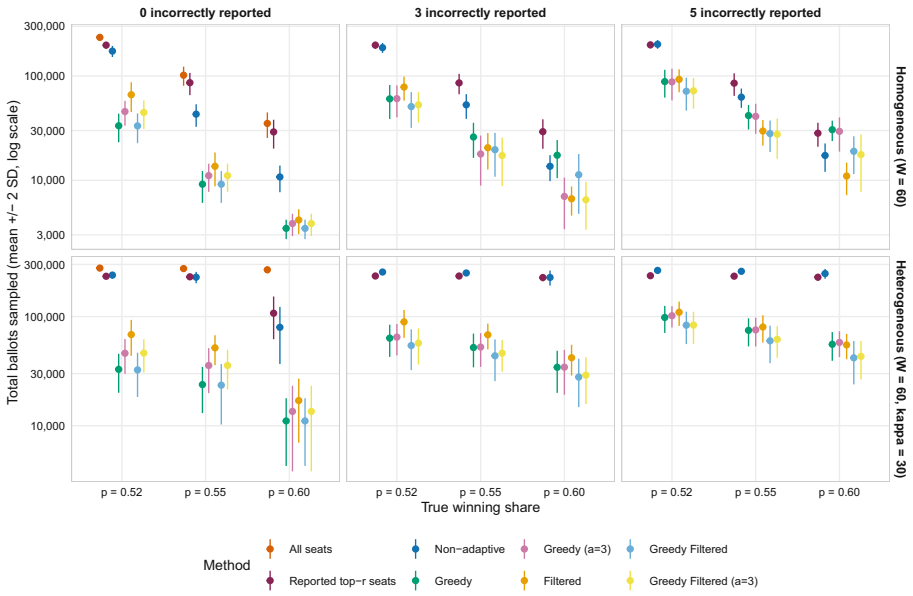


Fig. 2. Simulated two-candidate plurality contests. The total number of ballots sampled to certification: each point shows the mean across 100 replicates and error bars denote ±2 standard deviations. Columns correspond to the number of falsely reported seats, n_{false}. Rows correspond to different ways to set the margins: top row shows Scenario 1, bottom row shows Scenario 2. Results for the *All seats* method are only reported for $n_{\text{false}} = 0$, since for $n_{\text{false}} > 0$ it tests $H_{|\mathcal{W}|/\mathcal{W}}$, which is a true null and requires a full recount with probability at least $1 - \alpha$.

We also varied $n_{\text{false}} \in \{0, 3, 5\}$, giving $3 \times 3 = 9$ configurations for each of the two scenarios. We replicated each one 100 times. Figure 2 shows our results, in terms of the number of ballots sampled until certification.

All six strategies for certifying the parliamentary majority were markedly more efficient than the two benchmarks. *All seats* necessarily uses the most ballots: certifying $H_{|\mathcal{W}|/\mathcal{W}}$ requires *every* reported seat to individually exceed the threshold. *Top-r seats* is handicapped in the same way: during its first phase it effectively certifies $H_{r/\mathcal{R}}$, which is again driven by the minimum process. In contrast, certifying $H_{r/\mathcal{W}}$ depends on the product of the bottom $|\mathcal{W}| - r + 1$ seat-level processes, so stronger seats can carry the minimum. Among the majority schemes, *Non-adaptive* was least efficient, since it samples every seat regardless of how informative each is. The 'filtered' variants outperform their pure-greedy counterparts once $n_{\text{false}} > 0$, as expected, by redirecting effort away from seemingly false seats.

We provide some further results in Appendix D, in which we also vary the number of reported winning seats $|\mathcal{W}| \in \{51, 60, 80\}$ and the heterogeneity parameter $\kappa_{\text{het}} \in \{10, 30, 100\}$. Our broad conclusions remained the same.

4.2 Indian General Election

To evaluate the audit framework on a realistic parliament-scale election, we simulated audits using data from the 2014 Indian general election of the Lok Sabha (Lower House). We used candidate-level vote totals for all 543 constituencies from a community-scraped ECI dataset[6], originally sourced from the Election Commission of India's results portal. The Bharatiya Janata Party (BJP) won $|\mathcal{W}| = 282$ of the 543 seats, forming a single-party majority. Certifying a BJP majority requires verifying that at least $r = \lfloor 543/2 \rfloor + 1 = 272$ of these 282 reported winning seats were truly won.

Across the 282 BJP seats, the number of candidates ranged from $L_s = 5$ to $L_s = 43$ (median 15), and the number of ballots per seat ranged from approx. 87,000 to 1,512,000 (median 1,010,000), summing to a total of about 282 million ballots. The winner's vote share ranged from 0.264 to 0.758 (median 0.495). Because each L_s-candidate plurality contest has $J_s = L_s - 1$ SHANGRLA assertions, the minimum assorter mean across these head-to-head comparisons is the bottleneck for certifying a seat. In the BJP seats, this minimum assorter mean ranged from 0.5002 to 0.7812 (median 0.5831).

We generated n_{false} falsely reported seats by swapping the vote totals of the reported winner and the runner-up. The BJP remains the reported winner, but the runner-up is the true winner in the underlying ballot population. We apply this perturbation to the n_{false} most marginal BJP seats, where marginality is measured by the two-candidate margin between the BJP candidate and the runner-up. We vary $n_{\text{false}} \in \{0, 3, 5\}$. We replicated each of these three configurations 100 times.

Figure 3 reports the total number of ballots sampled before certification. We see qualitative behaviour similar to Sect. 4.1. All adaptive schemes dominated the benchmarks by one to three orders of magnitude: the *All seats* baseline required roughly 250 million ballots, and the *Reported top-r seats* benchmark required around 5 million across all n_{false}, whereas the others typically required only a few hundred thousand. Within the parliamentary-majority schemes, the *Greedy* ($a = 0$) scheme degraded sharply as we increased n_{false} (going from about 400k to 3.8M ballots), demonstrating the price of pouring effort into false seats. Widening the active set to $a = 3$ restored robustness at $n_{\text{false}} = 0$, where it is the most efficient scheme overall. However, once $n_{\text{false}} > 0$, the 'filtered' schemes took over, showing consistently superior performance when $n_{\text{false}} = 5$.

5 Discussion

We have developed a risk-limiting audit for parliamentary majorities. We formalised this task as a partial conjunction hypothesis test and constructed a sequential parliament-level audit statistic by combining seat-level statistics. The

[6] https://github.com/datameet/india-election-data/tree/master/parliament-elections/election2014, accessed on 2026-04-19.

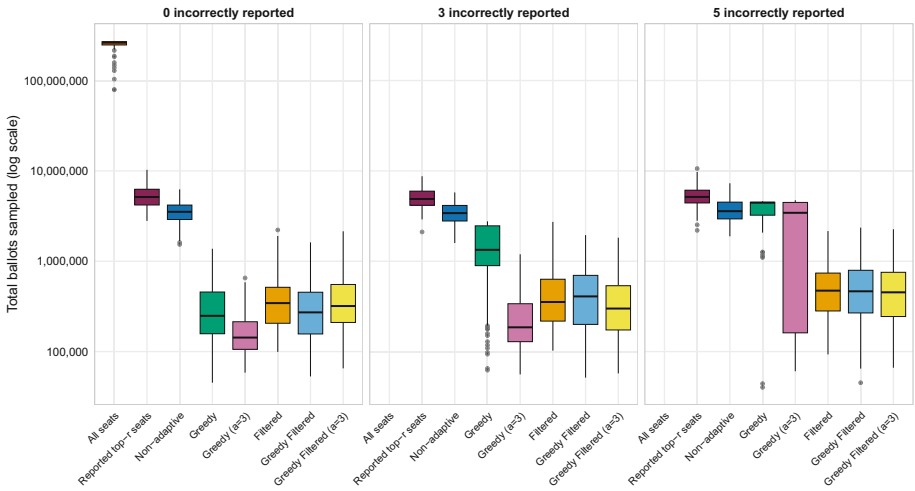


Fig. 3. India 2014. Box plots of the total number of ballots sampled to certification across 100 replicates, with one panel per number of falsely reported seats n_{false}. Results for the *All seats* method are omitted for $n_{\text{false}} > 0$, like in Fig. 2.

resulting procedure controls the risk of certifying an incorrect majority outcome, while allowing the auditor to adaptively allocate sampling effort across seats.

Although our experiments focused on ballot-polling audits of plurality elections, the methodology is not limited to this setting. Because it is built on SHANGRLA assertions, it inherits that framework's generality, covering a wide variety of social choice functions, other audit styles such as comparison audits, and settings with missing ballots. It can stretch further still, targeting *any* hypothesis of the form $H_{r/\mathcal{W}}$ for any value of r, not just majorities.

In practice, each constituency needs the same infrastructure as any single-contest RLA. The extra burden imposed by the majority audit is cross-seat coordination: a central authority must aggregate data from the sampled ballots, update $E_{r/\mathcal{W},t}$, and issue the next round's instructions. This overhead favours a few large-batch rounds rather than the single-ballot rounds we used for exposition (n.b., the risk limit holds for any batch size). Designing this coordination to run efficiently in practice is an important problem in its own right, one we leave for future work. The burden is nonetheless far lighter than auditing every seat: the total sample is orders of magnitude smaller, and the adaptive schemes concentrate effort on a few seats, allowing most constituencies to stop early.

In our experiments we only considered ballot-polling audits. When the relevant election infrastructure is available, more sophisticated and efficient audit designs are possible, such as ballot-level comparison audits. The sample size required for such audits typically grows linearly with respect to the reciprocal of the margin, whereas for ballot-polling it grows quadratically [8]. Thus, com-

bining our method with a comparison audit should lead to even smaller sample sizes, in the order of thousands of ballots for the Indian election example.

Our audit certifies that a reported winning party won a majority of seats in its own right. This is the pertinent question when a single party, or a pre-declared alliance, wins an outright majority, as in our Indian example. When no party/alliance wins such a declared majority, our method is *not* relevant: if a governing coalition is negotiated only after the election, the majority to certify is not known while the ballots remain available, and such elections are better served by auditing the individual seats. However, because our audit is built from ordinary seat-level assertions, it can be run in tandem with a full seat-by-seat audit: the majority can be certified 'midstream', as soon as the parliament-level statistic $E_{r/\mathcal{W},t}$ crosses $1/\alpha$, while the seat-by-seat audit continues towards certifying every individual seat or recounting those that appear miscounted.

Several directions remain for future work: (1) It would be useful to compare our sampling strategies in diverse scenarios, such as other election and audit types. (2) The efficiency of the audit depends on the choice of sampling scheme and ALPHA tuning parameters; future work could study alternative choices. This includes investigating the benefit of setting them based on reported vote totals/margins, in scenarios where these are available. (3) Determining optimal ways to set sample sizes per seat for each round would be useful for practice. (4) The process $E_{r/\mathcal{W}}$ is anytime-valid for testing $H_{r/\mathcal{W}}$, but might not be optimal. Similarly, the process $E_{s,t}$ is one option for summarizing seat-level information, but might not be optimal. Investigating if there are more efficient alternatives would be of interest.

Appendices

The appendices to this paper are available online at: https://arxiv.org/abs/2607.21082.

References

1. Benjamini, Y., Heller, R.: Screening for partial conjunction hypotheses. Biometrics **64**(4), 1215–1222 (2008)
2. Bogomolov, M., Heller, R.: Replicability across multiple studies. Stat. Sci. **38**(4), 602–620 (2023). Preprint arXiv:2210.00522
3. Ek, A., Stark, P.B., Stuckey, P.J., Vukcevic, D.: In: Adaptively weighted audits of instant-runoff voting elections: AWAIRE. In: E-Vote-ID 2023. LNCS, vol. 14230, pp. 35–51. Springer (2023) Preprint arXiv:2307.10972
4. Ek, A., Stark, P.B., Stuckey, P.J., Vukcevic, D.: Efficient weighting schemes for auditing instant-runoff voting elections. In: Financial Cryptography and Data Security. FC 2024. LNCS, vol. 14746, pp. 18–32. Springer (2025). Preprint arXiv:2403.15400
5. Fisher, R.A.: Statistical Methods for Research Workers. Fourteenth Hafner Publishing Co., New York (1973)

6. Gablenz, P., Sabatti, C.: Catch me if you can: signal localization with knockoff e-values. J. Royal Stat. Soc. Ser. B Stat. Methodol **87**(1), 56–73 (2025). Preprint arXiv:2306.09976
7. Hoang, A.T., Dickhaus, T.: Combining independent p-values in replicability analysis: a comparative study. J. Stat. Comput. Simul. **92**(10), 2184–2204 (2022). Preprint arXiv:2104.13081
8. Lindeman, M., Stark, P.B.: A gentle introduction to risk-limiting audits. IEEE Secur. Priv. **10**(5), 42–49 (2012)
9. Mohanty, V., Culnane, C., Stark, P.B., Teague, V.: Auditing Indian elections. In: E-Vote-ID 2019. LNCS, vol. 11759, pp. 150–165. Springer (2019). Preprint arXiv:1809.04235
10. Ottoboni, K., Stark, P.B., Lindeman, M., McBurnett, N.: Risk-limiting audits by stratified union-intersection tests of elections (SUITE). In: E-Vote-ID 2018. LNCS, vol. 11143, pp. 174–188. Springer (2018). Preprint: arXiv:1809.04235
11. Ramdas, A., Grünwald, P., Vovk, V., Shafer, G.: Game-theoretic statistics and safe anytime-valid inference. Stat. Sci. **38**(4), 576–601 (2023). Preprint arXiv:2210.01948
12. Spertus, J.V., Sridhar, M., Stark, P.B.: Sequential stratified inference for the mean (2026). arXiv:2409.06680
13. Stark, P.B.: Sets of half-average nulls generate risk-limiting audits: SHANGRLA. In: Financial Cryptography and Data Security. FC 2020. LNCS, vol. 12063, pp. 319–336. Springer (2020). Preprint arXiv:1911.10035
14. Stark, P.B.: ALPHA: Audit that learns from previously hand-audited ballots. Ann. Appl. Stat **17**(1), 641–679 (2023). Preprint arXiv:2201.02707
15. Ville, J.: Etude critique de la notion de collectif, Monographies des Probabilites, vol. 3. Gauthier-Villars Paris (1939)
16. Vovk, V., Wang, R.: E-values: calibration, combination and applications. Ann. Stat. **49**(3), 1736–1754 (2021). Preprint arXiv:1912.06116
17. Vovk, V., Wang, R.: Merging sequential e-values via martingales. Electron. J. Stat. **18**(1), 1185–1205 (2024). Preprint: arXiv:2007.06382
18. Waudby-Smith, I., Ramdas, A.: Estimating means of bounded random variables by betting. J. Royal Stat. Soc. Ser. B Stat. Methodol. **86**(1), 1–27 (2024). Preprint: arXiv:2010.09686

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





A Simple Protocol for Tracker Based Voting

Rosario Giustolisi[1(✉)], Peter Y. A. Ryan[2], and Maryam Sheikhi Garjan[3]

[1] IT University of Copenhagen, Copenhagen, Denmark
rosg@itu.dk
[2] University of Luxembourg, Esch-sur-Alzette, Luxembourg
peter.ryan@uni.lu
[3] University of Hamburg, Hamburg, Germany
maryam.sheikhi.garjan@uni-hamburg.de

Abstract. We present the simplest tracker based voting protocol to date that combines receipt-free and transparent verification, and immediate ballot-casting assurance. Existing schemes achieve these goals only partially or with additional assumptions: Selene and Hyperion rely on a private channel to deliver tracker information after tallying, while Surtr removes that assumption at the cost of more complex voter interaction. Our protocol avoids these trade-offs. Each voter encrypts the chosen candidate and all remaining candidates, each paired with a tracker. The system then immediately shuffles and decrypts these candidate-tracker pairs and publishes them in the voter's bulletin-board row. This allows the voter to verify in cleartext that the chosen vote is associated with the tracker, while a coercer observes only indistinguishable pairs. Tallying uses one re-encrypted real-vote ciphertext per voter, enabling direct tally extraction.

1 Introduction

A common way to let voters check that their ballot is counted is to give each voter a tracker that eventually appears on the bulletin board. State-of-the-art tracker-based systems [3,11,13,15,19] make this verification transparent: voters can identify their vote in cleartext. Some of these schemes [11,13,19] also provide receipt-free verification by allowing a coerced voter to equivocate about trackers.

However, the current designs involve important trade-offs. Selene [19] and Hyperion [11] require a private channel so that tracker information can be delivered privately (i.e. hidden from the coercer) to the voter after tallying. Surtr [13] removes this private-channel assumption, but requires more complex voter interaction to preserve coercion resistance, as voters are required to keep track of any coercer-chosen tracker until tallying.

In this paper, we present a simple tracker-based protocol that provides transparent and receipt-free verification without a private channel and without additional voter burden. Compared with prior systems, we avoid: (i) verified generation, encryption, and mixing of trackers as in Selene; (ii) dual keys and exponentiation mixnets as in Hyperion; and (iii) extra dual keys and assumptions

© The Author(s) 2027
D. Duenas-Cid et al. (Eds.): E-Vote-ID 2026, LNCS 17083, pp. 70–86, 2027.
https://doi.org/10.1007/978-3-032-39512-2_5

on coerced voters as in Surtr. In addition to privacy, verifiability, and receipt-free verification, our protocol also gives immediate ballot-casting assurance [1], whereas prior tracker-based systems focus on post-tally verification.

The core idea is simple: the voter encrypts the chosen candidate and associates it with an encrypted commitment to the tracker, and does the same for each of the other $N - 1$ candidates, each with a distinct tracker commitment. Importantly, the chosen candidate-tracker commitment encryption pair is positioned first in the ballot. The tally servers then immediately mix and decrypt all N candidate-tracker commitment pairs and publish the shuffled plaintext pairs in that voter's bulletin-board row. This gives immediate voter confirmation: the voter can find the chosen candidate next to the corresponding tracker commitment by using the tracker associated with that commitment. At the same time, it still hides the real choice from a coercer, who sees only N candidate-tracker commitment pairs. After voting closes, the first encrypted pairs from all voters' bulletin-board rows are included in tallying, enabling direct tally extraction.

Contributions. We present a simple tracker-based voting protocol that achieves transparent and receipt-free verification, and ballot-casting assurance. The protocol is simple both cryptographically and operationally: it relies only on standard mixnet constructions and can be instantiated with any threshold encryption scheme and any verifiable parallel shuffle. From the voter's perspective, the vote-tracker commitment association is revealed immediately on the bulletin board, publicly and in the clear, so the voter does not have to remember any coercer-chosen tracker until after the election closes. We also give a formal security analysis proving that the protocol satisfies ballot privacy and verifiability under standard cryptographic assumptions.

2 Related Work

End-to-end verifiable voting systems have been a major focus of research over the past decade. Pioneering work on optical-scan voting includes Scantegrity [7], which demonstrated end-to-end voter-verifiable elections in a real-world setting. Remotegrity [22] extended this approach to remote voting, providing end-to-end verifiable Internet voting. Both systems rely on voter-verifiable paper records and cryptographic proofs, setting the foundation for transparent verification.

Selections [8] tackled the challenge of combining Internet voting with coercion-resistance through over-the-shoulder coercion-resistance mechanisms. SureVote [6] introduced a code-voting method allowing voters to use pre-distributed codes for response verification without cryptographic operations. Pretty Good Democracy [20] improved this by distributing acknowledgment code management among a group of trustees, decreasing the reliance on a single trusted server. More recently, post-quantum secure end-to-end verifiable schemes like Epoque [4] have extended these ideas to resist quantum adversaries. These works collectively demonstrate that achieving transparent, verifiable voting with user-friendly mechanisms remains an active research challenge. Our protocol contributes to this line of work by combining immediate ballot-casting assurance

with transparent and receipt-free verification, while maintaining the simplicity and directness that voters expect.

Many voting protocols offer strong coercion resistance [2,9,10,12,14,16,21], but without transparent verification. Only a small set of systems combine receipt-free verification with plaintext-style vote tracking. Early work includes the boardroom scheme of [3], where trackers are generated by voters, and sElect [15], which uses partially voter-generated trackers for accountability. However, these designs do not provide receipt-free verification, because a voter can reveal the tracker to a coercer before tallying.

Selene [19] provides receipt-free verification by disclosing tracker information only after tally publication, enabling a coerced voter to equivocate. Its notification mechanism, however, relies on a private channel to preserve that equivocation in front of a coercer. Hyperion [11] replaces trackers with Diffie–Hellman terms that let voters recover their vote in plaintext, but it still requires a private notification channel. Surtr [13] achieves receipt-free verification without the need for a private notification channel. However, as in the previous schemes, verification is post-tally, and a coerced voter needs to write down a coercer-chosen tracker until the tally is published in order to equivocate.

Our scheme achieves transparent voter verification immediately for *recorded-as-intended*, since the vote-tracker association is revealed on the bulletin board after ballot casting, allowing the voter to verify in clear that their vote is correctly associated with the tracker. In Selene and Hyperion, transparent voter verification also covers *tallied-as-recorded*, but no transparent verification is possible before tallying and without a private notification channel.

3 A Simple Protocol for Tracker Based Voting

We first give an informal overview of the protocol. Its full specification is described in Sect. 3.3.

3.1 Overview

Ballot Casting. Let $\mathbb{V} = \{v_1, \ldots, v_N\}$ be the candidate set and $\mathbb{I} = \{i_1, \ldots, i_M\}$ the electoral roll. The voter i generates N random trackers $x_0, x_1, \ldots, x_{N-1}$ and computes the corresponding tracker commitments $g^{x_0}, g^{x_1}, \ldots, g^{x_{N-1}}$. The voter chooses their candidate $v_o \in \mathbb{V}$ which is associated with the tracker x_0. The voter's ballot β_i then contains N ciphertext pairs, one per candidate, each encrypting a candidate alongside an encryption of the tracker commitment, with the real vote pair $([\![v_o]\!], [\![g^{x_0}]\!])$ placed first in the ballot. The ballot is posted on the voter's row $\mathcal{BB}_i$ of the bulletin board $\mathcal{BB}$.

Ballot Recording. Right after β_i is posted on the bulletin board, the tally servers retain the first ciphertext $[\![v_o]\!]$ as the ciphertext that will later be tallied. Then, the servers run a verifiable pairwise re-encryption mix over all N candidate-tracker commitment ciphertext pairs from β_i. This shuffles the pairs into a

random order, publishing a proof of correct re-encryption. Then, they apply a decryption mix to reveal the N plaintext pairs, publishing in the voter's row $\mathcal{BB}_i$ the list $(v_{k_1}, g^{x_{k_1}}), \ldots, (v_{k_N}, g^{x_{k_N}})$ in shuffled order, together with a proof of correct decryption.

Voter Verification. The voter can now check their vote on the bulletin board immediately after casting, without waiting for tallying. Knowing v_o and x_0, the voter device computes g^{x_0} and scans the published plaintext pairs to find the entry (v_o, g^{x_0}). Finding it confirms that the correct candidate is associated with the correct tracker: the vote was cast as intended and is recorded on the board. Because decrypted candidate-tracker pairs appear after ballot casting, this constitutes immediate ballot-casting assurance.

Receipt-Freeness. All N candidates appear in the decrypted list, each associated with a distinct commitment, so the coercer cannot determine which pair corresponds to the real vote by inspection. If the voter is coerced into revealing their vote, the voter can point to any entry (v_k, g^{x_k}) in the published list and produce the corresponding tracker x_k as a convincing demonstration that v_k was their real choice. Since the commitment g^{x_k} is on the board and x_k opens it, a coercer has no way to refute this equivocation. Thus, a coerced voter can always claim to have voted for the coercer-preferred candidate regardless of their true choice. Note that all trackers, tracker commitments, and their associations with candidates can be shared with the coercer right after ballot casting, as they do not reveal the voter's choice.

Our protocol provides receipt-freeness for the verification mechanism, but not full receipt-freeness for ballot casting. In particular, we assume that the coercer neither provides the voter with a precomputed ballot nor learns the randomness used to generate the ballot. Otherwise, the coerced voter may be unable to equivocate, and the coercer could verify the cast vote on the bulletin board. One possible mitigation is to require the server to challenge the voter to re-randomize the ballot before final casting. A potential direction is to use interactive proofs and malleable signatures, as in BeleniosRF [5]. However, the malleable signature construction used in BeleniosRF cannot be directly applied to our protocol, because it does not support conjunctions of zero-knowledge proofs of knowledge, which our construction relies on. Achieving full receipt-freeness would significantly increase protocol and voter-interaction complexity.

Tallying. After voting closes, the tally servers collect the first vote ciphertexts $\{[\![v_{oi}]\!]\}_{i=1}^{M}$ from the voter rows. These M ciphertexts are processed by a further re-encryption and decryption mix, yielding the multiset of plaintext votes $\{v_{oi}\}_{i=1}^{M}$ in shuffled order. The election result can be computed and verified universally from this list of plaintext votes. No additional private channels, dual keys, or post-tallying delivery steps are required.

3.2 Cryptographic Primitives

Our construction uses standard building blocks. A public-key encryption scheme $(\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is used for candidate and tracker ciphertexts. A digital signature scheme $(\mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{SVerify})$ authenticates the ballots submitted by voters. Trackers are generated by the randomized algorithm $\mathbf{x} \xleftarrow{\$} \mathsf{TKeyGen}(pp)$, where $\mathbf{x} = (x_0, \ldots, x_{N-1})$. We assume a distributed (t, n)-threshold decryption setting with no trusted authority: any set of at least t tallying parties can decrypt, and the resulting decryption is publicly verifiable.

Privacy-preserving shuffle of ciphertexts is performed using verifiable re-encryption and decryption mixnets. Integrity of the mixnet operations is ensured by the corresponding proofs of correct re-encryption and decryption.

Finally, verifiability is guaranteed through non-interactive zero-knowledge proofs of knowledge (NIZKPoK). For each relation R used by the protocol we write $(\mathsf{Proof}_R, \mathsf{Verify}_R)$ for the prover and verifier algorithms. These relations cover the statements needed later in the protocol description, such as well-formed encryptions, correct re-encryption/shuffling, and correct threshold decryption.

3.3 Formal Description

The algorithms that define the protocol are as follows. For each voter identity $id \in \mathbb{I}$, let $\mathcal{BB}_{id}$ denote the row of voter id in the bulletin board $\mathcal{BB}$.

- $\mathsf{EASetup}(1^\lambda, \mathbb{I}, \mathbb{V}) \rightarrow (pk_{EA}, sk_{EA})$: on input the security parameter 1^λ, the electoral roll $\mathbb{I}$, and the candidate list $\mathbb{V}$, generates public parameters pp and computes $(pk_{EA}, sk_{EA}) \xleftarrow{\$} \mathsf{EKeyGen}(pp)$.
- $\mathsf{Setup}(id, pk_{EA}) \rightarrow (id, (\mathbf{x}, \mathbf{g^x}, sk), (\mathbf{ctr}, \pi_{\mathbf{ctr}}, pk))$: on input of the voter identity id and pk_{EA}, which contains pp, computes $(sk, pk) \xleftarrow{\$} \mathsf{SKeyGen}(pp)$, where pk contains a proof of knowledge π_{sk}, and generates random trackers $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1})$ and tracker commitments $\mathbf{g^x} = (g^{x_0}, g^{x_1}, \ldots, g^{x_{N-1}})$ by running $(\mathbf{x}, \mathbf{g^x}) \xleftarrow{\$} \mathsf{TKeyGen}(pp)$. Then, it computes $ctr_i \xleftarrow{\$} \mathsf{Enc}(pk_{EA}, g^{x_i}; r_{x_i})$ for $i \in [0, N-1]$, and sets $\mathbf{ctr} = (ctr_0, ctr_1, \ldots, ctr_{N-1})$. It runs $\pi_{\mathbf{ctr}} \xleftarrow{\$} \mathsf{Proof}_{ctr}(\chi, \omega)$, where $\chi = (\mathbf{ctr}, pk_{EA})$, $\omega = (\mathbf{x}, \mathbf{r_x})$ s.t. $(\chi, \omega) \in R^{\text{ctr}}$ iff

$$ctr_0 = \mathsf{Enc}(pk_{EA}, g^{x_0}; r_{x_0}) \wedge \ldots \wedge ctr_{N-1} = \mathsf{Enc}(pk_{EA}, g^{x_{N-1}}; r_{x_{N-1}}).$$

  Note that $\pi_{\mathbf{ctr}}$ retains the order of the tracker commitments and it is bound to the voter's identity id.
- $\mathsf{Vote}(id, sk, pk_{EA}, (v_0, \mathbf{ctr}, \pi_{\mathbf{ctr}})) \rightarrow \beta$: on input voter identity $id \in \mathbb{I}$, voter signing key sk, tallying servers' public key pk_{EA}, vote option $v_0 \in \mathbb{V}$, encryptions $\mathbf{ctr}$, proof $\pi_{\mathbf{ctr}}$, and implicit input $(\mathbb{V}, \mathbb{I})$ where $N = |\mathbb{V}|$,
  1. Compute $ct_0 \xleftarrow{\$} \mathsf{Enc}(pk_{EA}, v_0; r_{v_0})$
  2. Set $\mathbb{W} = \mathbb{V} \setminus \{v_0\}$

3. For $i \in [1, N-1]$, do:
$$v_i \xleftarrow{\$} \mathbb{W}$$
$$ct_i \xleftarrow{\$} \mathsf{Enc}(pk_{EA}, v_i; r_{v_i})$$
$$\mathbb{W} = \mathbb{W} \setminus \{v_i\}$$
4. Set $\mathbf{ct} = (ct_0, ct_1, \ldots, ct_{N-1})$, $\mathbf{v} = (v_0, v_1, \ldots, v_{N-1})$, and randomness $\mathbf{r_v} = (\mathbf{r_{v_0}}, \mathbf{r_{v_1}}, \ldots, \mathbf{r_{v_{N-1}}})$.
5. Run $\pi_v \xleftarrow{\$} \mathsf{Proof}_{vote}(\chi, \omega)$, where $\chi = (\mathbf{ct}, pk_{EA}, \mathbb{V})$, $\omega = (\mathbf{v}, \mathbf{r_v})$ s.t. $(\chi, \omega) \in R^{\text{vote}}$ iff

$$ct_0 = \mathsf{Enc}(pk_{EA}, v_0; r_{v_0}) \wedge \ldots \wedge ct_{N-1} = \mathsf{Enc}(pk_{EA}, v_{N-1}; r_{v_{N-1}}) \wedge$$
$$v_0 \in \mathbb{V} \wedge \ldots \wedge v_{N-1} \in \mathbb{V} \wedge \mathbf{v} \equiv \mathbb{V}$$

6. Set $\pi = (\pi_v, \pi_{\mathbf{ctr}})$ and compute $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, (id, \mathbf{ct}, \mathbf{ctr}, \pi))$, and return $\beta = (id, pk, \mathbf{ct}, \mathbf{ctr}, \pi, \sigma)$.

- $\mathsf{Validate}(\mathcal{BB}, \beta) \rightarrow \top/\bot$: on input ballot $\beta = (id, pk, \mathbf{ct}, \mathbf{ctr}, \pi, \sigma)$ and implicit input $(pk_{EA}, \mathbb{I}, \mathbb{V})$ checks that (i) $id \in \mathbb{I}$, (ii) no terms in β already appear in $\mathcal{BB}$, (iii) $\top \leftarrow \mathsf{SVerify}(pk, (id, \mathbf{ct}, \mathbf{ctr}, \pi), \sigma)$, and (iv) $\top \leftarrow \mathsf{Verify}(\chi, \pi)$, where $\chi = (pk_{EA}, \mathbf{ct}, \mathbf{ctr}, \mathbb{V})$ and $\pi = (\pi_v, \pi_{\mathbf{ctr}})$. If any of the checks fail, it returns $\bot$; otherwise it returns $\top$.
- $\mathsf{Append}(\mathcal{BB}, \beta) \rightarrow \mathcal{BB}$: on input a ballot $\beta = (id, pk, \mathbf{ct}, \mathbf{ctr}, \pi, \sigma)$, checks that $\mathsf{Validate}(\mathcal{BB}, \beta) = \top$. If so, it appends β to $\mathcal{BB}_{id}$.
- $\mathsf{ImmediateMix}(\mathcal{BB}, sk_{EA}, id) \rightarrow \mathcal{BB}$: on input $\mathcal{BB}$, sk_{EA}, and id, obtains the ballot $\beta = (id, pk, \mathbf{ct}, \mathbf{ctr}, \pi, \sigma)$ from $\mathcal{BB}_{id}$ and then does the following:
  1. On input $(ct_0, ctr_0), (ct_1, ctr_1), \ldots, (ct_{N-1}, ctr_{N-1})$, run a re-encryption mix to shuffle in parallel the vote ciphertexts and the tracker ciphertexts producing $\{(ct'_j, ctr'_j)\}_{j \in [N]}$ and proof $\pi_{\mathsf{r\text{-}enc}}$.
  2. Run a decryption mix on $\{(ct'_j, ctr'_j)\}_{j \in [N]}$, which produces the shuffled $\{(v_j, g^{x_j})\}_{j \in [N]}$ and proof π_d.
  3. Append $(\{(v_j, g^{x_j})\}_{j \in [N]}, \pi_{\mathsf{r\text{-}enc}}, \pi_d)$ to $\mathcal{BB}_{id}$.
- $\mathsf{VerifyVote}(\mathcal{BB}, id, v, x_v) \rightarrow \bot/\top$: on input $\mathcal{BB}$, voter identity id, vote option v and tracker x_v, checks that (i) there is a ballot β in $\mathcal{BB}_{id}$ and that $\mathsf{Validate}(\mathcal{BB}, \beta) = \top$; (ii) $\pi_{\mathsf{r\text{-}enc}}$ and π_d in $\mathcal{BB}_{id}$ are valid; (iii) compute g^{x_v} to recover (v_i, g^{x_v}) from $\mathcal{BB}_{id}$ and check that $v_i = v$. If any of the checks fail, it returns $\bot$; otherwise it returns $\top$.
- $\mathsf{Tally}(\mathcal{BB}, sk_{EA}) \rightarrow (R, \Pi)$: on input $\mathcal{BB}$, let $\{ct_{0i}\}_{i=1}^{M}$ be the first vote ciphertexts extracted from all valid non-empty voter rows of $\{\mathcal{BB}_{id}\}_{id \in \mathbb{I}}$:
  1. Run a re-encryption mix on $\{ct_{0i}\}_{i=1}^{M}$, producing shuffled $\{ct_{0i}'\}_{i=1}^{M}$ and proof $\pi_{\mathsf{r\text{-}enc\text{-}tally}}$.
  2. Run a decryption mix on $\{ct_{0i}'\}_{i=1}^{M}$, producing shuffled $\{v_i\}_{i=1}^{M}$ and proof $\pi_{\mathsf{d\text{-}tally}}$.
  3. Set $\Pi = (\pi_{\mathsf{r\text{-}enc\text{-}tally}}, \pi_{\mathsf{d\text{-}tally}})$ and compute the result $R = (R_v)_{v \in \mathbb{V}}$ by counting occurrences of each candidate in $\{v_i\}_{i=1}^{M}$.
- $\mathsf{VerifyTally}(\mathcal{BB}, (R, \Pi)) \rightarrow \bot/\top$: on input $\mathcal{BB}$, the tally result and corresponding proofs (R, Π), verifies all proofs on $\mathcal{BB}$ and the correctness of (R, Π). If any of the checks fail, it returns $\bot$; otherwise it returns $\top$.

The protocol is organised into three phases.

Setup Phase: The tallying servers run $\mathsf{EASetup}(1^\lambda, \mathbb{I}, \mathbb{V})$ to generate their key pair (pk_{EA}, sk_{EA}). Each voter then runs $\mathsf{Setup}(id, pk_{EA})$ to register and obtain a signing key pair (sk, pk), together with trackers $\mathbf{x}$, the tracker commitments $\mathbf{g^x}$, the tracker commitment encryptions $\mathbf{ctr}$, and the corresponding proof $\pi_{\mathbf{ctr}}$. If two devices are used (one for tracker generation and one for ballot casting), the casting device generates the signing key pair, while the tracker device generates the tracker-related material. This split supports cast-as-intended verifiability and, consequently, end-to-end verifiability (see Sect. 3.4). The bulletin board $\mathcal{BB}$ is initialised with the voter's id, pk, and π_{sk}.

Voting Phase: The voter chooses a candidate $v_0 \in \mathbb{V}$ and creates a ballot β using $\mathsf{Vote}(id, sk, pk_{EA}, (v_0, \mathbf{ctr}, \pi_{\mathbf{ctr}}))$. The ballot is submitted to $\mathcal{BB}$ via $\mathsf{Append}(\mathcal{BB}, \beta)$. The tallying servers then execute $\mathsf{ImmediateMix}(\mathcal{BB}, sk_{EA}, id)$, which mixes and decrypts the candidate-tracker commitment pairs and appends the resulting plaintext pairs to the voter's row $\mathcal{BB}_{id}$. The voter checks inclusion and consistency by running $\perp/\top \leftarrow \mathsf{VerifyVote}(\mathcal{BB}, id, v, x_v)$.

Tallying Phase: After voting closes, the tallying servers run the tallying algorithm $(R, \Pi) \leftarrow \mathsf{Tally}(\mathcal{BB}, sk_{EA})$ to compute the final tally R. Any observer can verify its correctness by running $\perp/\top \leftarrow \mathsf{VerifyTally}(\mathcal{BB}, (R, \Pi))$.

3.4 End-to-End Verifiability

We discuss informally here how our protocol achieves end-to-end verifiability. We provide a formal proof in Sect. 5.

End-to-end verifiability ensures that the system provides cast-as-intended, recorded-as-cast, and tallied-as-recorded assurances to voters and observers. In our protocol, the voter can directly verify cast-as-intended and recorded-as-cast assurances after ballot casting. Tallied-as-recorded assurance is guaranteed by the verifiable mixnet operations, which are publicly verifiable.

Cast-as-intended verifiability concerns auditing the ballot-creation process. This is typically achieved either by using *return codes* [6] or by giving the voter two devices, with at least one of them trusted. As in other tracker-based voting schemes, our protocol can achieve cast-as-intended verifiability with two devices. Specifically, if the voter has two separate devices (e.g., one for tracker generation and one for ballot casting), Setup can be split as follows: the tracker device runs $\mathsf{TKeyGen}$ to generate trackers and tracker commitments, and the casting device runs $\mathsf{SKeyGen}$ to generate the signing key pair. The first generated tracker, x_0 (the one associated with the voter's choice), is shown to the voter, and the encrypted commitments $\mathbf{ctr}$ with the corresponding proof $\pi_{\mathbf{ctr}}$ are transferred to the casting device (e.g., via QR code) before the ballot is cast. The voter inputs their vote choice to the casting device which generates a list of the encryptions of the vote options with the voter's choice first in the list. The proof $\pi_{\mathbf{ctr}}$ on encrypted commitments ensures that anyone can detect whether a malicious

casting device has altered their order. Therefore, the voter does not need to verify that the first ballot term is the encryption of the chosen candidate. As we show later formally, it is sufficient for cast-as-intended that either the tracker device or the casting device is trusted.

Recorded-as-cast verifiability is achieved by the immediate mixnet operations that reveal the vote-tracker commitment associations on the bulletin board right after ballot casting. The mixnets provide publicly verifiable proofs of their operations. However, the voter can directly check that their vote is correctly recorded by verifying that tracker x_0 corresponds to the tracker commitment g^{x_0} associated with the chosen candidate. This also provides immediate assurance to voters, as they do not need to wait until tallying to verify that their vote is recorded as intended.

Tallied-as-recorded verifiability is guaranteed by the verifiable mixnet operations that are publicly verifiable. The tallying mixnet processes the first vote ciphertexts, and the corresponding proofs ensure that the votes are correctly included in the tally and that the decryption is correct. Therefore, any observer can verify that the final tally corresponds to the recorded votes on the bulletin board.

3.5 Performance

The protocol relies on standard cryptographic primitives that are already present in state-of-the-art voting systems. In fact, it uses fewer components than prior tracker-based systems, as it does not require exponentiation mixnets or dual keys. The most expensive operations are the mixnet operations, which are performed once per voter for the immediate mix and once for the final tally. This has two advantages over prior tracker-based systems: (i) immediate mixing enables faster verification by voters, who do not need to scan the entire bulletin board to find their vote-tracker association; and (ii) the tallying phase depends only on the number of voters, not also on the number of candidates as in Surtr, because only the first ciphertexts of the ballots are included in the tallying mixnet.

4 Ballot Privacy

In this section, we prove that our protocol defined by $\mathcal{VS} = (\mathsf{EASetup}, \mathsf{Setup}, \mathsf{Vote}, \mathsf{Validate}, \mathsf{Append}, \mathsf{ImmediateMix}, \mathsf{VerifyVote}, \mathsf{Tally}, \mathsf{VerifyTally})$ satisfies ballot privacy using the ballot privacy game $\mathsf{Ballot\text{-}Priv}^b$ described in Fig. 1. We adapt the game-based security definition of ballot privacy from [11]. In Hyperion, there is a post-tally verification that relies on the secret value sent to the voter through a private channel. However, since our scheme is not based on post-tally verification and does not require a secure channel, we modify the oracle $\mathsf{VerifyVote}$ in the definition by eliminating the $\mathsf{GetSecret}$ function. Furthermore, we modify the oracle Board by adding the function $\mathsf{ProcessBallot}$, which appends the output of the decryption mixnet for each voter id to the corresponding row of the bulletin board. A voter ballot together with the output of the decryption

mixnet is referred to as Processed. In our scheme, the oracle VerifyVote calls the oracle Board to verify the voter ballot. For ballot privacy, we assume that the bulletin board and the tallying server are trusted, each voter is registered with a unique id, and that the voter devices are trusted, that is, these devices do not leak sensitive information, such as the secret signing key, trackers, or encryption randomness. In the Ballot-Priv^b game, the PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ interacts with the challenger, which executes the algorithms of the voting system through the various oracles. SK and PK are used to store the voter's secret key sk and public key pk. The left and right vote inputs to the oracle LoR, are stored in V_0 and V_1, respectively, while ST stores the state st associated with the voters' ballots. The adversary $\mathcal{A}_1$ can register honest voters HV using the oracle HSetup or register dishonest voters DV with chosen public keys using the oracle DSetup. ValidCred outputs $\top$ if π_{sk} for voter id is valid, and $\perp$ otherwise. The adversary $\mathcal{A}_2$ has access to the following oracles:

- LoR(id, v_0, v_1), which allows the adversary to have an honest voter id cast one of two chosen votes (v_0 or v_1) based on a hidden bit b.
- Tally(), which initiates the tallying process but only if the result of votes cast in both scenarios (when $b = 0$ and $b = 1$) is identical.
- VerifyVote(id), which models a voter verifying their vote after casting.
- Board(), which allows the adversary $\mathcal{A}_2$ to obtain a published version of the Bulletin Board $\mathcal{BB}$.
- Cast(id, β), which enables $\mathcal{A}_2$ to directly submit a ballot β to $\mathcal{BB}$ for a voter id.
- Verify(), which allows $\mathcal{A}_2$ to check the validity of all ballots and the tally on public $\mathcal{BB}$.

The adversary $\mathcal{A}$ wins the game if it can determine the hidden value of b where b means that the challenger is processing v_b for an honest voter id. We define the advantage of the adversary $\mathcal{A}$ in the ballot privacy experiment as follows:

Theorem 1 (Ballot Privacy). *Assume that the encryption scheme with zero-knowledge proof of knowledge is IND-1-CCA secure, the signature scheme is EUF-CMA secure, the zero-knowledge proof systems are zero-knowledge, and the re-encryption mix is permutation hiding. Then the voting scheme $\mathcal{VS}$ achieves ballot privacy. In particular, for every PPT adversary $\mathcal{A}$,*

$$\left|\Pr[\text{Ballot-Priv}^1_{\mathcal{A},\mathcal{VS}}(\lambda) = 1] - \Pr[\text{Ballot-Priv}^0_{\mathcal{A},\mathcal{VS}}(\lambda) = 1]\right| \leq Adv^{\mathsf{IND-1-CCA}}_{\mathcal{B}} + \epsilon,$$

where ϵ is a negligible function in security parameter λ and $Adv^{\mathsf{IND-1-CCA}}_{\mathcal{B}}$ denotes the advantage of an adversary $\mathcal{B}$ against IND-1-CCA security.

Proof. **Game G_0.** Let G_0 be the ballot privacy game Ballot-Priv$^b_{\mathcal{A},\mathcal{VS}}$ in Fig. 1. We denote the adversary advantage in game G_0 by $\mathsf{Adv}^{G_0}_{\mathcal{A}}$. In G_0, upon a query LoR(id, v_0, v_1), the challenger samples two random trackers and computes their commitments (x_0, g^{x_0}) and (x_1, g^{x_1}) corresponding to the vote options v_0, v_1, respectively. It then defines two possible ballots for the two inputs of the

Game $\mathsf{Ballot}-\mathsf{Priv}^{b}_{\mathcal{A},\mathcal{VS}}$
$\mathsf{SK},\mathsf{PK},\mathsf{ST},\mathsf{V}_0,\mathsf{V}_1 \leftarrow []_{\perp}$
$HV, DV \leftarrow \emptyset$
$(\mathsf{pk}_{EA},\mathsf{sk}_{EA}) \leftarrow \mathsf{EASetup}()$
$\mathsf{st}_{\mathcal{A}} \leftarrow \mathcal{A}_1(\mathsf{pk}_{EA},\mathsf{PK})$
$b' \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}})$
return $(b'=b)$

Oracle $\mathsf{LoR}(id, v_0, v_1)$
if $id \notin HV$ **return** $\perp$
$sk \leftarrow \mathsf{SK}[id]$
$pk \leftarrow \mathsf{PK}[id]$
$(\beta,\mathsf{st}) \leftarrow \mathsf{Vote}(id, sk, \mathsf{pk}_{EA}, v_b)$
if $\mathsf{Validate}(\mathcal{BB},\beta)=\perp$
  return $\perp$
$\mathbf{V}_0[id] \leftarrow v_0, \mathbf{V}_1[id] \leftarrow v_1,$
$\mathsf{ST}[id] \leftarrow \mathsf{st}, \mathcal{BB}[id] \leftarrow \beta$

Oracle $\mathsf{Board}()$
for $id \in HV \cup DV$
  if $\mathcal{BB}[id] \neq \emptyset$ **and** $\mathsf{Processed}[id]=0$
    $\mathcal{BB}[id] \leftarrow \mathsf{ProcessBallot}(\mathcal{BB}[id],\mathsf{sk}_{EA})$
    $\mathsf{Processed}[id]=1$
return $\mathcal{BB}$

Oracle $\mathsf{VerifyVote}(id)$
if $id \notin HV \cup DV$**return** $\perp$
$\mathcal{BB} \leftarrow \mathsf{Board}()$
$pk \leftarrow \mathsf{PK}[id], sk \leftarrow \mathsf{SK}[id], \mathsf{st} \leftarrow \mathsf{ST}[id]$
return $\mathsf{VerifyVote}(\mathcal{BB},(id,pk),\mathsf{st})$

Oracle $\mathsf{HSetup}(id)$
$(sk,pk) \leftarrow \mathsf{Setup}(id,\mathsf{pk}_{EA})$
if $\mathsf{ValidCred}(id,pk,\mathsf{pk}_{EA})=\perp$
  return $\perp$
$\mathsf{SK}[id] \leftarrow sk,\ \mathsf{PK}[id] \leftarrow pk$
$HV \leftarrow HV \cup \{id\}$

Oracle $\mathsf{DSetup}(id,pk)$
if $id \in HV$ **or**
$\mathsf{ValidCred}(id,pk,\mathsf{pk}_{EA})=\perp$
  return $\perp$
$\mathsf{PK}[id] \leftarrow pk$
$DV \leftarrow DV \cup \{id\}$

Oracle $\mathsf{Cast}(id,\beta)$
if $id \notin HV \cup DV$**or**
$\mathsf{Validate}(\mathcal{BB},\beta)=\perp$
  return $\perp$
$\mathcal{BB}[id] \leftarrow \beta$

Oracle $\mathsf{Tally}()$
if $\rho(\mathbf{V}_0)=\rho(\mathbf{V}_1)$
return $\mathsf{Tally}(\mathcal{BB},\mathsf{sk}_{EA})$

Oracle $\mathsf{Verify}()$
return $\mathsf{Verify}(\mathcal{BB},(R,\Pi))$

Fig. 1. Ballot privacy game. The adversary $\mathcal{A}$ wins if $b'=b$.

oracle $\mathsf{LoR}(id,v_0,v_1)$ as follows: $\beta_0 = (id,pk,(ct_0,ctr_0),(ct_1,ctr_1),\pi_0,\sigma_0)$ and $\beta_1 = (id,pk,(ct_1,ctr_1),(ct_0,ctr_0),\pi_1,\sigma_1)$. When the board is published, β_b is processed by the decryption mixnet. The mixnet samples a random permutation τ and publishes the shuffled plaintext pairs $(v_{\tau(b)}, g^{x_{\tau(b)}}),(v_{\tau(1-b)}, g^{x_{\tau(1-b)}})$.

Game G_1**.** In game G_1, all proofs (vote, tracker, re-encryption, decryption, and mix proofs) are simulated. By the zero-knowledge property, the adversary $\mathcal{A}$ cannot distinguish simulated proofs from real proofs. Therefore, we have $\mathsf{Adv}^{G_0}_{\mathcal{A}} \leq \mathsf{Adv}^{G_1}_{\mathcal{A}} + \epsilon_{zk}$, where ϵ_{zk} denotes the advantage of an adversary in a zero-knowledge proof system in distinguishing simulated proofs from real proofs.

Game G_2**.** In G_2, we modify the oracle $\mathsf{Cast}(id,\beta)$ in game G_1 so that the adversary $\mathcal{A}$ can only cast a ballot β on behalf of a dishonest voter $id \in DV$. This is because we assume that the signature scheme is existentially unforgeable. Hence,

we have $\mathsf{Adv}_{\mathcal{A}}^{G_1} \leq \mathsf{Adv}_{\mathcal{A}}^{G_2} + \epsilon_{\mathsf{EUF-CMA}}$, where $\epsilon_{\mathsf{EUF-CMA}}$ denotes the advantage of an adversary against the EUF-CMA property of the signature scheme.

Game G_3**.** In G_3 we modify the oracle VerifyVote in game G_2 so that it always returns $\top$. By the correctness of the scheme, honestly generated ballots are accepted by VerifyVote. Moreover, the adversary cannot modify or cast on behalf of honest voters due to the unforgeability of the signature scheme. Therefore, $\mathsf{Adv}_{\mathcal{A}}^{G_2} = \mathsf{Adv}_{\mathcal{A}}^{G_3}$.

Game G_4**.** This game is identical to the previous game G_3 except that, in the mixnet, instead of computing re-encryptions of the input ciphertext pairs, the challenger computes fresh encryptions of the same plaintext pairs. That is, for each input pair encrypting (v_j, x_j), the challenger samples fresh randomness r'_j, s'_j and outputs $(ct'_j, ctr'_j) = (\mathsf{Enc}(pk_{EA}, v_j; r'_j), \mathsf{Enc}(pk_{EA}, g^{x_j}; s'_j))$. For re-randomizable encryption, this distribution is identically distributed, since $\mathsf{ReEnc}(\mathsf{Enc}(m; r); r') = \mathsf{Enc}(m; r + r')$, and $r + r'$ is uniformly distributed whenever r' is sampled uniformly at random. Hence we have: $\mathsf{Adv}_{\mathcal{A}}^{G_4} = \mathsf{Adv}_{\mathcal{A}}^{G_3}$.

Game G_5**.** Game G_5 differs from G_4 in that in oracle Board, we replace the real re-encryption/decryption mixnet execution by an ideal execution that first samples a uniform random permutation $\tau \leftarrow S_2$, applies it to $((v_b, g^{x_b}), (v_{1-b}, g^{x_{1-b}}))$ corresponding to the output ciphertext pairs of the oracle $\mathsf{LoR}(id, v_0, v_1)$ namely $((ct_b, ctr_b), (ct_{1-b}, ctr_{1-b}))$ and outputs $((v_{\tau(b)}, g^{x_{\tau(b)}}), (v_{\tau(1-b)}, g^{x_{\tau(1-b)}}))$. By the permutation-hiding property of the mixnet, the adversary $\mathcal{A}$ cannot distinguish the real mixnet execution from this ideal permuted execution except with negligible probability ϵ_{mix}. Hence, $\mathsf{Adv}_{\mathcal{A}}^{G_5} \leq \mathsf{Adv}_{\mathcal{A}}^{G_4} + \epsilon_{mix}$.

Game G_6**.** In G_6, analogously to game G_5, we modify the oracle Tally(), where the ideal mixnet computes the result R based on the output oracles LoR and Cast provided that $\rho(\mathbf{V}_0) = \rho(\mathbf{V}_1)$. In this case the result R is the mix of $\rho(\mathbf{V}_b)$ and the decryption of the all valid ballots cast by the adversary $\mathcal{A}$. By the correctness of decryption and indistinguishability of the real and ideal executions of the mixnet, we have that $\mathsf{Adv}_{\mathcal{A}}^{G_6} = \mathsf{Adv}_{\mathcal{A}}^{G_5}$ (Fig. 2).

In our scheme, the vote encryption scheme with zero-knowledge proof of knowledge with label *id* for each voter satisfies IND-1-CCA security. Finally, we argue that the advantage of the adversary in game G_6, namely $\mathsf{Adv}_{\mathcal{A}}^{G_6}$, is equal to the advantage of the adversary $\mathcal{B}$ in the IND-1-CCA security game, namely $\mathsf{Adv}_{\mathcal{A}}^{G_6} = \mathsf{Adv}_{\mathcal{B}}^{\mathsf{IND-1-CCA}}$. By combining the games, we obtain

$$\mathsf{Adv}_{\mathcal{A}}^{Ballot\text{-}Priv^b} \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{IND-1-CCA}} + \epsilon_{zk} + \epsilon_{\mathsf{EUF-CMA}} + \epsilon_{mix},$$

where $\epsilon = \epsilon_{zk} + \epsilon_{\mathsf{EUF-CMA}} + \epsilon_{mix}$ and $\mathsf{Adv}_{\mathcal{A}}^{Ballot\text{-}Priv^b}$ is defined as

$$\left|\Pr[\mathsf{Ballot\text{-}Priv}^1_{\mathcal{A},\mathcal{VS}}(\lambda) = 1] - \Pr[\mathsf{Ballot\text{-}Priv}^0_{\mathcal{A},\mathcal{VS}}(\lambda) = 1]\right|.$$

Game $\mathsf{Verif}(\mathcal{A})$
$\mathsf{SK}_c, \mathsf{PK}_c \leftarrow [\,]_\perp$
$\mathsf{SK}_v, \mathsf{PK}_v, \mathsf{ST}, V \leftarrow [\,]_\perp$
$H_c, D_c, H_v, D_v, V_i \leftarrow \emptyset$
$(\mathsf{pk}_{EA}, \mathsf{sk}_{EA}) \leftarrow \mathsf{EASetup}()$
$(\mathcal{BB}, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}_1(\mathsf{pk}_{EA}, \mathsf{PK})$
for $id \in V_{\mathsf{Chkd}} \cap V_i$ **do**
if $\mathsf{VerifyVoted}(id, \mathcal{BB}) \neq \top$
return $\perp$
if $id \in H_v$ **then**
$sk \leftarrow \mathsf{SK}_v[id],\ v \leftarrow V[id]$
if $\mathsf{VerifyVote}(id, sk, v, \mathcal{BB}) \neq \top$
return $\perp$
$(r, \pi, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}_2(\mathsf{st}_\mathcal{A}, \mathcal{BB})$
if $\mathsf{Verify}(\mathcal{BB}, r, \pi) \neq \top$**or**
$r = \perp$ **return** $\perp$
$M \leftarrow V_{\mathsf{Chkd}} \cap (H_c \cup H_v)$
$a \leftarrow |V_{\mathsf{Chkd}} \cap D_v|$
$b \leftarrow |D_c \setminus (V_{\mathsf{Chkd}} \cap H_v)|$
if $\exists n \in \{a, \ldots, b\}$,
$\exists v_1, \ldots, v_n \in \mathcal{V}$,
$\exists S \subseteq (V_i \setminus V_{\mathsf{Chkd}}) \cap H_c$ s.t.
$r = \rho([V[j]]_{j \in S}) \star \rho([v_j]_{j=1}^n) \star \rho([V[j]]_{j \in M})$
return $\perp$

Oracle $\mathsf{HCastSetup}(id)$
for $id \notin D_c$
$(sk, pk) \leftarrow \mathsf{Setup}_c(id, \mathsf{pk}_{EA})$
if $\mathsf{ValidCred}_c(id, pk, \mathsf{pk}_{EA}) = \top$
then $\mathsf{SK}_c[id] \leftarrow sk;\ \mathsf{PK}_c[id] \leftarrow pk,$
$H_c \leftarrow H_c \cup \{id\}$

Oracle $\mathsf{DCastSetup}(id, pk)$
for $id \notin H_c$
if $\mathbf{ValidCred}_c(id, pk, \mathsf{pk}_{EA}) = \top$
then $\mathsf{PK}_c[id] \leftarrow pk,$
$D_c \leftarrow D_c \cup \{id\}$

Oracle $\mathsf{HVerSetup}(id)$
if $id \notin D_v$
then return $\perp$
$(sk, pk) \leftarrow \mathsf{Setup}_v(id, \mathsf{pk}_{EA})$
if $\mathsf{ValidCred}_v(id, pk, \mathsf{pk}_{EA}) = \top$
$\mathsf{SK}_v[id] \leftarrow sk;\ \mathsf{PK}_v[id] \leftarrow pk$
$H_v \leftarrow H_v \cup \{id\}$

Oracle $\mathsf{DVerSetup}(id, pk)$
if $id \in H_v$**return** $\perp$
if $\mathsf{ValidCred}_v(id, pk, \mathsf{pk}_{EA}) = \perp$
return $\perp$
$\mathsf{PK}_v[id] \leftarrow pk, \quad D_v \leftarrow D_v \cup \{id\}$

Oracle $\mathsf{Vote}(id, v)$
if $id \notin H_c \cup D_c$, Return $\perp$
$V_i \leftarrow V_i \cup \{id\};\ V[id] \leftarrow v$
if $id \in H_c$ **then**
$sk \leftarrow \mathsf{SK}_c[id];\ pk \leftarrow \mathsf{PK}_c[id]$
$(\beta, st) \leftarrow \mathsf{Vote}(\mathsf{pk}_{EA}, id, sk, pk, v)$
$\mathsf{ST}[id] \leftarrow st, \quad$ Return β

Fig. 2. Game-based definition of verifiability against a malicious voting board and a compromised device. The indices v and c denote verification and vote casting, respectively. The index i in V_i denotes voters with intended votes.

5 Verifiability

In this section, we prove that the proposed protocol satisfies verifiability against malicious tallying authorities and bulletin board[1], even in the presence of compromised voting or tracker generator devices. In particular, we consider the case

[1] If individual ballot verification is performed once and before tallying, then the bulletin board is assumed to be append-only. This is necessary since otherwise the bulletin board could remove the ballot of the voter immediately after the voter verifies their ballot. If verification is instead performed (also) after tallying, this assumption is not required.

where either the secret signing key or the secret tracker is leaked through a compromised device. We adopt the verifiability definition of Hyperion [11] to capture the cast-as-intended and recorded-as-cast guarantees of the voting scheme. We assume that each voter is uniquely registered with credentials (id, pk) and that the bulletin board provides a consistent public view during the entire election. Each voter uses two separate devices: a vote-casting device holding the signing key sk, and a tracker generator device holding the secret trackers $\mathbf{x}$. The definition models leakage from compromised voting devices by distinguishing between honest and corrupted casting and tracker generator devices. A corrupted casting device D_c leaks the voter's signing key sk, while a corrupted tracker generator device D_v leaks the secret trackers $\mathbf{x}$. We denote by V_{Chkd} the set of voters who successfully complete the verification procedure after casting their vote. This procedure consists of checking that a valid ballot associated with their identity id appears on the bulletin board $\mathcal{BB}$, and executing $\mathsf{VerifyVote}$. We use $\mathsf{VerifyVoted}$ as shorthand for this combined verification procedure. Voters whose casting and tracker generator devices are both corrupted are considered fully corrupted voters. The algorithms $\mathsf{ValidCred}_c$ and $\mathsf{ValidCred}_v$ are used for the signing key and the verification key, respectively. We apply the same distinction to the Setup algorithm.

The definition captures the adversary's ability to output a valid bulletin board and tally result while achieving at least one of the following malicious goals:

(i) changing the vote of a voter who possesses at least one honest device and successfully verifies their vote, namely voters in $V_{\mathsf{Chkd}} \cap H_v$ or $V_{\mathsf{Chkd}} \cap H_c$;
(ii) for voters equipped with honest casting devices, forging, stuffing, or modifying ballots in any manner other than simply dropping the ballot, namely voters in $V_i \setminus (V_{\mathsf{Chkd}} \cap H_c)$;
(iii) for the remaining eligible voters, namely voters in $V_{\mathsf{Chkd}} \cap D_v$ and $D_c \setminus (V_{\mathsf{Chkd}} \cap H_v)$, cast more than one vote per voter.

Correctness. We first prove the correctness of the proposed scheme. Assume that all parties execute the protocol honestly and that the underlying cryptographic primitives satisfy their correctness properties, including the encryption scheme, signature scheme, mixnet, and zero-knowledge proof systems. Under these assumptions, for every honest voter the generated public key and the ballot generated by the voting algorithm are accepted by the bulletin board and verification procedures, the recorded ballot encodes the voter's intended vote, and the tallying algorithm correctly computes the election result from the set of recorded honest ballots. Hence, the published tally corresponds to the voters' intended votes.

Theorem 2 (Verifiability). *Assume the signature scheme is EUF-CMA secure, the NIZK proof systems satisfy simulation-sound extractability, the soundness of the mixnet, and hardness of the discrete logarithm problem. Then the voting scheme $\mathcal{VS}$ achieves verifiability.*

Proof. We first prove that for voters with honest vote-casting devices who have verified their ballot, $id \in (V_{\mathsf{Chkd}} \cap H_c)$, the adversary cannot change the voter's ballot on $\mathcal{BB}$. We also show that for $id \in H_c$, the adversary can at most drop ballots, which proves that $\exists S \subseteq (V_i \setminus V_{\mathsf{Chkd}}) \cap H_c$. Let $id \in H_c$, where the honest casting device generates a ballot containing a valid signature. Since the adversary $\mathcal{A}$ does not know the secret signing keys of honest voters, the EUF-CMA security of the signature scheme guarantees that $\mathcal{A}$ cannot forge a valid ballot, alter an honestly cast ballot, or cast ballots on behalf of any voter $id \in H_c$. Therefore, for all $id \in H_c$, the adversary can only drop the cast ballot. However, for all voters $id \in (V_{\mathsf{Chkd}} \cap H_c)$, the ballots appear unchanged on $\mathcal{BB}$.

Now we consider all voters with $id \in (V_{\mathsf{Chkd}} \cap H_v)$, namely voters with an honest tracker generator device who check their vote. Let $\{v_0, v_1\}$ be the candidate set and let β be a ballot of voter id with intended vote v_0. Then we have $\beta = (id, pk, ((ct_0, ctr_0), (ct_1, ctr_1)), \pi, \sigma)$ as a ballot of voter id, where $(ct_i, ctr_i) = (\mathsf{Enc}(pk_{EA}, v_i; r_{v_i}), \mathsf{Enc}(pk_{EA}, g^{x_i}; r_{x_i}))$ for $i = 0, 1$. The adversary $\mathcal{A}$ (tally server) may have access to (g^{x_0}, g^{x_1}) by decrypting ctr_i and to the voter's secret signing key sk, assuming a dishonest vote-casting device. If the adversary $\mathcal{A}$ generates a modified ballot by altering the vote ciphertexts $\{ct_i\}$ or their ordering, this modification results in $\{(v_0, g^{x_1}), (v_1, g^{x_0})\}$, which does not pass verification. Assume that the adversary $\mathcal{A}$ modifies the ballot β to β' with intended vote v_1 such that voter verification $\mathsf{VerifyVote}(BB, (id, pk), \mathsf{st}) = \top$, which means that the ballot β' on $\mathcal{BB}$ is a valid ballot for the voter with (id, pk) and the decryption mixnet returns a mix of $\{(v_0, g^{x_0}), (v_1, g^{x_1})\}$. Since the honest tracker generator device generates a proof π_{ctr} proving knowledge of witnesses (x_0, x_1, r_0, r_1) such that $ctr_i = \mathsf{Enc}(pk_{EA}, g^{x_i}; r_i)$ for $i \in \{0, 1\}$, any adversary producing a fresh accepting proof for a modified ordered tuple must know valid witnesses corresponding to the new statement. Since the adversary $\mathcal{A}$ outputs a modified ballot $\beta' = (id, pk, ((ct'_1, ctr'_1), (ct'_0, ctr'_0)), \pi', \sigma')$ with valid proof $\pi' = (\pi'_v, \pi'_{\mathbf{ctr}})$, where $(ctr'_1, ctr'_0, \pi'_{\mathbf{ctr}}) \neq (ctr_0, ctr_1, \pi_{\mathbf{ctr}})$. By the knowledge soundness of the proof system, there exists an extractor $\mathcal{E}$ that outputs witnesses (x'_1, x'_0, r'_0, r'_1) for the modified statement (ctr'_1, ctr'_0), where $x'_1 = x_1$ and $x'_0 = x_0$. Therefore, the existence of $\mathcal{A}$ contradicts the discrete logarithm assumption underlying the binding and knowledge soundness properties of the commitments.

We now determine the bound on the number of malicious ballots that may be altered by the adversary. Consider voters with $id \in D_c$, namely voters with a dishonest vote-casting device. For voters satisfying $id \in D_c$ and $id \notin V_{\mathsf{Chkd}}$, the adversary $\mathcal{A}$ may arbitrarily modify the vote or drop the ballot entirely, since no successful verification is performed. Therefore, the upper bound is $b = |D_c \setminus (V_{Chkd} \cap H_v)|$. Next, we consider voters such that $id \in D_c$ and $id \in V_{\mathsf{Chkd}} \cap D_v$. For these voters, $\mathsf{VerifyVoted}(id, \mathcal{BB}) = \top$, which guarantees that a valid ballot has been recorded on the bulletin board. However, since the tracker generator device is also dishonest, successful verification does not guarantee that the recorded vote corresponds to the voter's intended vote. Therefore, the minimum number

of ballots whose plaintext votes may be maliciously chosen by the adversary is lower-bounded by $a = |V_{\mathsf{Chkd}} \cap D_v|$.

Let $M = V_{Chkd} \cap (H_v \cup H_c)$, $S \subseteq V_i \setminus (V_{Chkd} \cap H_c)$, and $n \in [a, b]$, which denotes the number of votes defined by $\mathcal{A}$. Therefore we have

$$R = \rho\left([V[j]]_{j \in \mathcal{S}}\right) \star \rho\left([v_j]_{j=1}^{n}\right) \star \rho\left([V[j]]_{j \in \mathcal{M}}\right),$$

where $V[j]$ denotes the intended vote of voter j and $[v_j]_{j=1}^{n}$ are the votes chosen by the adversary $\mathcal{A}$.

6 Conclusion

We have presented a tracker-based voting protocol that simultaneously achieves receipt-free and transparent verification, and immediate ballot-casting assurance, while remaining simpler than prior related schemes. The key insight is for the tally servers to immediately mix and decrypt all candidate-tracker commitment pairs upon ballot receipt, then publish the result in the voter's bulletin-board row. This step provides clear vote-tracker confirmation right after casting, without requiring a private notification channel, post-tally interaction, or memorization of any coercer-chosen tracker until tallying.

From a cryptographic standpoint, the scheme relies only on standard building blocks: threshold public-key encryption, digital signatures, and verifiable re-encryption and decryption mixnets, all of which are already present in state-of-the-art tracker-based voting systems. In particular, it uses a strict subset of the primitives required by Selene, Hyperion, and Surtr: there are no exponentiation mixnets, no dual keys, and no verified tracker generation. We formally prove that the scheme satisfies ballot privacy under IND-1-CCA encryption, EUF-CMA signatures, and a permutation-hiding mixnet, and we give a game-based verifiability definition that covers malicious boards and device malware.

An interesting direction for future work is to investigate everlasting privacy variants of the scheme, following the approach of [17] for Hyperion. It would also be worth studying whether the two-device assumption for cast-as-intended can be relaxed, perhaps achieving universally cast-as-intended verifiability [18] and simplifying the voter experience even further.

References

1. Adida, B., Neff, C.A.: Ballot casting assurance. In: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop. EVT 2006, USENIX Association (2006)
2. Araújo, R., Foulle, S., Traoré, J.: A practical and secure coercion-resistant scheme for remote elections. In: Dagstuhl Seminar Proceedings (2008)
3. Arnaud, M., Cortier, V., Wiedling, C.: Analysis of an electronic boardroom voting system. In: Heather, J., Schneider, S., Teague, V. (eds.) International Conference on E-Voting and Identity, Springer (2013). https://doi.org/10.1007/978-3-642-39185-9_7

4. Boyen, X., Haines, T., Müller, J.: Epoque: practical end-to-end verifiable post-quantum-secure e-voting. In: 2021 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 272–291. IEEE (2021)
5. Chaidos, P., Cortier, V., Fuchsbauer, G., Galindo, D.: BELENIOSRF: a non-interactive receipt-free electronic voting scheme. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1614–1625 (2016)
6. Chaum, D.: Surevote. Int. Patent WO **1**(55940), A1 (2001)
7. Chaum, D., et al.: Scantegrity: end-to-end voter-verifiable optical-scan voting. IEEE Secur. Priv. **6**(3), 40–46 (2008)
8. Clark, J., Hengartner, U.: Selections: internet voting with over-the-shoulder coercion-resistance. In: Danezis, G. (eds.) International Conference on Financial Cryptography and Data Security, pp. 47–61. Springer (2011). https://doi.org/10.1007/978-3-642-27576-0_4
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). https://doi.org/10.1007/978-3-642-27576-0_4
10. Cortier, V., Gaudry, P., Yang, Q.: Is the JCJ voting system really coercion-resistant? Cryptology ePrint Archive (2022)
11. Damodaran, A., Rastikian, S., Rønne, P.B., Ryan, P.Y.A.: Hyperion: transparent end-to-end verifiable voting with coercion mitigation. In: Nicomette, V., Benzekri, A., Boulahia-Cuppens, N., Vaidya, J. (eds.) European Symposium on Research in Computer Security, pp. 124–143. Springer (2025). https://doi.org/10.1007/978-3-032-07891-9_7
12. Giustolisi, R., Garjan, M.S., Schuermann, C.: Thwarting last-minute voter coercion. In: 2024 IEEE Symposium on Security and Privacy (SP) (2024)
13. Giustolisi, R., Garjan, M.S., Rønne, P.B.: Surtr: transparent verification with simple yet strong coercion mitigation. In: E-VOTE-ID (2025)
14. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Chaum, D., et al. (eds.) Towards Trustworthy Elections, pp. 37–63. Springer (2010). https://doi.org/10.1007/978-3-642-12980-3_2
15. Küsters, R., Müller, J., Scapin, E., Truderung, T.: sElect: a lightweight verifiable remote voting system. In: CSF, pp. 341–354. IEEE (2016)
16. Locher, P., Haenni, R., Koenig, R.E.: Coercion-resistant internet voting with everlasting privacy. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 161–175. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_11
17. Mosaheb, R., Rønne, P.B., Ryan, P.Y., Sarfaraz, S.: Direct and transparent voter verification with everlasting receipt-freeness (2024)
18. Ryan, P.Y.A., Rønne, P.B., Arriaga, A., Pereira, O.: End-to-end verifiable elections with casting of publicly audited (2025). cleartext ballots. E-VOTE-ID
19. Ryan, P.Y.A., Rønne, P.B., Iovino, V.: Selene: voting with transparent verifiability and coercion-mitigation. n: Clark, J., Meiklejohn, S., Ryan, P., Wallach, D., Brenner, M., Rohloff, K. (eds.) International Conference on Financial Cryptography and Data Security, pp. 176–192. Springer (2016). https://doi.org/10.1007/978-3-662-53357-4_12
20. Ryan, P.Y.A., Teague, V.: Pretty good democracy. In: Christianson, B., Malcolm, J.A., Matyáš, V., Roe, M. (eds.) International Workshop on Security Protocols, pp. 111–122. Springer (2009). https://doi.org/10.1007/978-3-642-36213-2_15

21. Spycher, O., Koenig, R., Haenni, R., Schläpfer, M.: A new approach towards coercion-resistant remote e-voting in linear time. In: Danezis, G. (eds.) Financial Cryptography and Data Security, pp. 182–189. Springer (2012). https://doi.org/10.1007/978-3-642-27576-0_15
22. Zagórski, F., Carback, R., Chaum, D., Clark, J., Essex, A., Vora, P.L.: Remotegrity: design and use of an end-to-end verifiable remote voting system. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) International Conference on Applied Cryptography and Network Security, pp. 441–457. Springer (2013). https://doi.org/10.1007/978-3-642-38980-1_28

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



SoK: Recommendations for Universally Verifiable Electronic Elections and Independent Audits

Audhild Høgåsen[1(✉)], Tjerand Silde[2], Chiara Spadafora[1], and Amund Fredrik Strømsnes[2]

[1] Swiss Post, Bern, Switzerland
{audhild.hoegaasen,chiara.spadafora.1}@post.ch
[2] Norwegian University of Science and Technology, Trondheim, Norway
tjerand.silde@ntnu.no, amundfs@alumni.ntnu.no

Abstract. Universally verifiable electronic voting systems enable any party to confirm that election results match recorded votes. In practice, however, meaningful verifiability extends beyond protocol design and requires well-defined audit procedures, comprehensive documentation, and verifier software that can be independently implemented and validated.

Existing research proposes audit frameworks and verification mechanisms, but provides limited guidance on how to design and operate systems to support independent third-party verification in real-world deployments.

In this work, we conduct a systematic literature review of auditing and verification in e-voting systems to identify requirements for practical auditability. We analyse the literature across six dimensions: verification requirements, available audit data, verification procedures, trust assumptions, documentation, and implementability, and synthesise the results into a unified framework. From this analysis, we derive twenty-four recommendations for system designers and election authorities.

Our findings show that practical auditability depends on aligning evidence generation, verification procedures, and trust assumptions, as well as reducing barriers to independent implementation of verifier software. We also identify key challenges, including system complexity, lack of standardisation, and incomplete or inconsistent documentation, which hinder independent verification. Together, these findings provide guidance for the design of practically auditable e-voting systems.

Keywords: Electronic voting · Universal verifiability · Audits · Verifier

This paper is a compressed version of the master's thesis of Amund Fredrik Strømsnes at the Norwegian University of Science and Technology (NTNU), titled "Enabling independent audits through verifier specifications – A development framework for verifier specifications in universally verifiable e-voting systems".

© The Author(s) 2027
D. Duenas-Cid et al. (Eds.): E-Vote-ID 2026, LNCS 17083, pp. 87–107, 2027.
https://doi.org/10.1007/978-3-032-39512-2_6

1 Introduction

In most democratic countries, elections are conducted using paper ballots, where procedural safeguards, such as private voting booths and public observation of the counting process, provide voters with confidence that votes remain secret and are counted correctly. However, these assurances rely on physical transparency and human oversight, which do not naturally translate to digital environments. As a result, electronic voting (e-voting) systems must provide equivalent guarantees through technical means, motivating the development of verifiable systems.

Kremer et al. [35] define a verifiable e-voting system as one that provides *individual verifiability* (voters can confirm that their intended vote has been correctly recorded by the system), *universal verifiability* (any party can verify that the announced election outcome corresponds to the set of recorded votes), and *eligibility verifiability* (any party can verify that all recorded votes originate from eligible voters and that no voter has voted more than once).

In this work, we focus primarily on universal (and, to some extent, eligibility) verifiability, as these properties depend on the availability and correctness of audit data and verification procedures. In contrast, individual verifiability relies on voter interaction and is therefore only indirectly related to the audit process considered in this paper.

Universal and eligibility verifiability are realised through the auditing of election artefacts using verifier software [21,27,58]. These audits validate both the correctness of the protocol execution and the resulting election outcome, based on the evidence produced during the election process. This shifts the focus of trust from the voting system alone to the verification process. In particular, the correctness and soundness of the verifier software become critical, as errors in the verification process may lead to incorrect conclusions about the election result.

This concern is closely related to the notion of software independence, defined by Rivest [47] as the property that system errors cannot cause undetectable changes to the election outcome. Subsequent work has argued that this property depends not only on the voting system, but also on the correctness of the verifier software and audit procedures [27]. Therefore, trust in an election outcome depends not only on the security of the voting protocol, but also on the integrity and correctness of the entire verification process.

If audits are performed solely by the system vendor, trust in the audit depends on trusting the vendor [10,61]. Independent audits reduce the necessary trust assumptions placed on the vendor. One way to facilitate independent audits is for the vendor to provide verifier software that can be operated by independent auditors [3]. However, an audit is not entirely independent if vendor-provided verifier software is used [7,58]. Therefore, to further reduce the necessary trust assumptions placed on the vendor, election authorities and system vendors should facilitate third-party implementations of verifier software.

Although the auditability of universally verifiable e-voting systems has been studied before, existing work primarily focuses on audit frameworks and verification mechanisms in isolation. In particular, it provides limited guidance on how

systems should be designed, documented, and operated to enable independent audits and third-party implementation of verifier software in practice.

As a result, there is a gap between theoretical verifiability and the ability of independent parties to meaningfully verify election outcomes. This work addresses that gap by conducting a systematic literature review to identify, consolidate, and structure recommendations that support practical auditability and independent verification.

1.1 Related Works

Treier [58] and Haenni et al. [21] propose audit frameworks that structure the verification process for universally verifiable elections. However, these works primarily focus on the methodology and organisation of audits, and provide limited guidance on how systems should be designed and documented to support independent audits and third-party implementation of verifier software in practice.

Several works address related aspects of auditability from different perspectives. Moser et al. [42] present recommendations for third parties implementing audit tools for individual verifiability, while Haines and Rønne [29] define requirements for making e-voting systems auditable from the viewpoint of source code examination. Haenni et al. [22] provide practical recommendations based on their experience developing an e-voting system. These contributions address specific aspects of auditability, including tooling, implementation, and system design.

However, existing work remains fragmented across these perspectives and does not provide a unified set of recommendations that connects protocol design, audit processes, documentation, trust assumptions, and implementability. In particular, limited attention has been given to how these aspects interact to enable independent and practically feasible verification by third parties.

This work addresses this gap by consolidating and structuring recommendations from the literature into a unified framework that supports the design of e-voting systems verifiable not only in theory, but also auditable in practice.

1.2 Contributions

We identify and consolidate recommendations from the literature to support the design, documentation, and execution of election audits enabling independent verification and third-party verifier software. We provide a unified view that connects audit design, documentation, and verifier implementability, bridging the gap between theoretical definitions of verifiability and the practical requirements to enable third-party audits.

Based on a systematic literature review, we derive actionable recommendations structured across six dimensions: verification requirements, available audit data, verification processes, trust assumptions, documentation, and implementability. Together, these dimensions form a unified framework that captures the end-to-end requirements to achieve practical and independently verifiable election systems.

The remainder of the paper is structured as follows. Section 2 describes the methodology used in the literature review, including database selection and query construction. Section 3 presents the findings, organised by category. Each subsection introduces the relevant findings, discusses tensions and trade-offs identified in the literature, and concludes with a set of consolidated recommendations. Finally, the paper concludes with a discussion of the overall findings, their implications, and directions for future work in Sect. 4.

2 Methodology

The literature review conducted in this work follows systematic principles to ensure transparency, reproducibility, and comprehensive coverage of relevant research. The approach consists of a structured search over a predefined set of databases using carefully constructed queries to obtain an initial corpus of records. This corpus is subsequently subjected to a multi-stage screening process, after which the remaining works are selected for detailed evaluation.

We queried three databases chosen to provide broad coverage of both e-voting and cryptographic research: the IACR Cryptology ePrint Archive[1], which hosts cryptographic preprints, and Web of Science[2] and Scopus[3], which index peer-reviewed publications across a wide range of disciplines.

Each database was queried using the same search string. Searches were restricted to the title, abstract, and keyword fields, and were constructed from keywords grouped into three thematic categories, as shown in Table 1. Within each category, keywords were combined using the Boolean operator OR, while the categories themselves were combined using AND.

To ensure precise matching, keywords were enclosed in quotation marks, except where wildcard characters were used. The wildcard symbol (*) was employed to capture multiple word forms sharing a common root; for example, *verifi** matches *verifier*, *verifiable*, and *verification*. The three keyword categories are defined as follows:

E-voting systems This category targets literature on electronic voting systems. As the field employs varied terminology, multiple synonyms, spellings, and related terms were included to maximise coverage.

Verifiability and audits This category ensures that retrieved records address verifiability or auditing. Some overlap may occur with terms such as *verifiable election* or *verifiable voting* from the first category; however, such overlap is retained to avoid excluding relevant works.

Implementation and practice This category captures literature related to implementation aspects, including verification processes, standardisation, and known implementation issues. It also includes terms such as *best practice* and *lessons learned*, which aim to identify experience-based and practical guidance.

[1] https://eprint.iacr.org.
[2] https://www.webofscience.com.
[3] https://www.scopus.com.

Table 1. Literature review search query components.

Category	Keywords
E-voting systems	electronic voting, e-voting, evoting, internet voting, i-voting, ivoting, online voting, cryptographic voting, remote voting, digital voting, verifiable voting, verifiable election*, voting system
Verifiability and audits	verifi*, audit*
Implementation and practice	best practice, lessons learned, implementation security, implementation recommendation, mistaken implementation*, reference implementation*, compliance, regulatory framework, assurance*, source code examination, technical standard, verification capabilit*, performance requirement, formal verification, vulnerabilit*, verification process, verification software

After obtaining the initial corpus of records, we conducted a multi-stage screening process to identify relevant works. This process consisted of three stages: removal of duplicate entries, filtering based on title and abstract, and a full-text review of the remaining records. In cases where multiple versions of the same work were retrieved, only the most recent version was included.

Records focusing on individual verifiability were retained when their findings on audit data, tooling, or documentation transfer to the audit process considered here. In total, 355 records were identified, of which 106 were duplicates. Of the remaining records, 98 were deemed irrelevant based on title and abstract screening, and a further 100 were excluded following full-text review.

The relatively high number of excluded records indicates that the search query may be broad. However, a broader query was intentionally chosen to maximise coverage and reduce the risk of omitting relevant literature, favouring recall over precision in the initial search phase.

To complement the resulting set of records and ensure broad coverage, two additional sources were incorporated after the screening process. First, all published long- and short-form papers from the E-Vote-ID conference[4] were evaluated for relevance and included in the final dataset when applicable. These papers were considered separately due to the venue's high relevance to e-voting research. This step resulted in the inclusion of two additional records.

Second, we performed backward and forward reference searches based on two works identified as particularly influential within the corpus: *How not to prove your election outcome* [25], which is highly cited by the retrieved records, and *Machine-checking the universal verifiability of ElectionGuard* [24], the only retrieved work constructing verifier software via formal verification tools. This process yielded six additional relevant records. The same search applied to the

[4] https://e-vote-id.org.

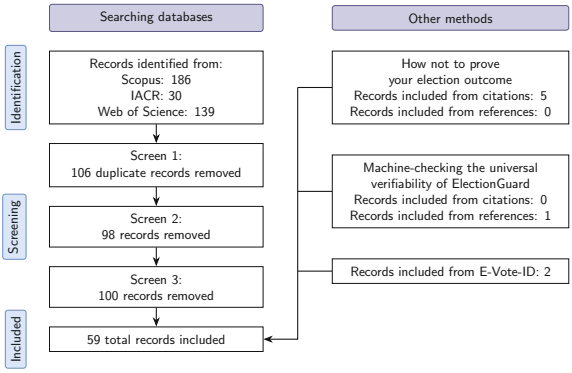


Fig. 1. Overview of the results from the literature review.

two other retrieved frameworks for verifier development, those of Treier [58] and Haenni et al. [21], led to no further inclusions, indicating that the reference search had saturated and that the outcome is not sensitive to the choice of seed papers.

An overview of the complete literature selection process is presented in Fig. 1. The findings are organised into categories that reflect key aspects of universally verifiable e-voting systems and the associated audit process. Table 2 summarises these categories, together with the number of records contributing to each.

Individual records may be assigned to multiple categories where they address more than one aspect of the audit process. Within certain categories, findings are further grouped to improve clarity and presentation.

Each category concludes with a discussion that synthesises the results, resolves differences in perspective across the literature, and highlights tensions arising from competing goals supported by conflicting methods or assumptions.

Table 2. Overview of categories and included records per category.

Category	Focus	No. of records
Verification requirements	What must be verified	39
Available audit data	What information auditors receive	33
Verification process	Audit methodology and procedure	31
Trust assumptions	Trust model of audits	9
Documentation	Structure of system/audit documentation	19
Implementability	Facilitating third-party audits	22

For each included work, we recorded the audit-related claims, requirements, limitations, and design practices relevant to the six categories of Table 2. We then

compared these items across papers and merged those that addressed the same auditability challenge. The resulting recommendations synthesise these findings.

Responsibility for applying the recommendations is shared among the parties involved in the election process: election authorities define the audit goals and trust model, vendors provide the necessary artefacts and documentation, auditors perform and report the verification, and independent implementers support the development of third-party verifiers.

3 Recommendations for Universally Verifiable E-Voting

Based on the literature, we identify practices and guidelines for conducting election audits and supporting independent verification. This section presents the findings organised into six categories, each capturing a distinct aspect of the audit process.

3.1 Verification Requirements

We organise the identified requirements into four groups, reflecting different aspects of the e-voting system and audit process.

System and Audit Correctness. This group concerns requirements related to the correctness and soundness of the e-voting system, the audit process, and the verifier software:

1. The system specification and implementation should be audited to demonstrate that the system provides its claimed security properties [2,16,40,53,56].
2. The audit process itself should be assessed to establish its correctness and soundness [2,18].
3. The verifier software should be validated to ensure its correctness and soundness, as errors may lead to incorrect conclusions about the election result [18,24,26,27,54,58].

Protocol Execution and Evidence. This group concerns requirements related to the evidence generated during protocol execution:

4. The audit should confirm that the protocol was executed correctly [12,18,21,49,50,54,58].
5. Post-election audits should confirm that no unresolved voter complaints were raised through individual verifiability mechanisms [16,38].
6. The audit of the tally should ensure that no invalid votes are included in the final result [3,12,16,17,31,40,45,46,51,53,61].
7. The correctness of the protocol execution, tally, and claimed security properties should be established by evaluating the evidence and proof transcripts[5] generated during execution [12,18,21,23,24,26,31,38,39,51,60,61].

[5] Proof transcripts may include, for example, proofs of correct decryption, mixing, or other cryptographic operations, depending on the protocol design.

Audit Data. The third group concerns the integrity and quality of the audit data:

8. The integrity of the audit data should be checked to ensure that it has not been modified or tampered with [3,9,11,13,21,45,50,55,57].
9. The authenticity of the audit data should be established to ensure that it originates from the expected system components [21,57,58].
10. The completeness and consistency of the audit data should be ensured, confirming that all required elements are present and mutually consistent, including public keys and other system parameters [21,28,30,31,40,49,57, 58].

Limitations and Pitfalls. The final group highlights issues that may prevent verification procedures from achieving their goals:

11. The evidence model of the e-voting protocol may be insufficient to support its claimed security properties [8,25,30,56,57].
12. Errors in the system implementation may invalidate security guarantees regardless of the evidence produced [20,25,38,41].
13. The verification process may be incomplete, either by considering only part of the available evidence or by failing to cover all relevant components of the system [15,22,25].

Discussion. If evidence-generation mechanisms and audit procedures are properly designed and aligned, evaluating the resulting evidence can provide strong assurance of correct protocol execution and tally. However, this assurance depends critically on the quality of the audit data. In particular, its integrity, authenticity, completeness, and consistency must be established to ensure that the evidence is trustworthy and relevant to the election.

These guarantees hold only within the assumptions of the underlying threat model. Implementation errors, incomplete evidence models, or flaws in the verification process may undermine the validity of the audit, emphasising the need for careful system and audit design.

Recommendations. We derive the following requirements for election audits:

1A Establish the correctness and soundness of the e-voting protocol, the audit process, and the verifier software.

1B Ensure the validity of the audit data to support correct interpretation of the protocol execution:

(a) Assess the evidence produced during protocol execution.
(b) Ensure completeness of the audit data.
(c) Ensure integrity of the audit data.
(d) Establish authenticity of the audit data.
(e) Ensure consistency of the audit data.

3.2 Available Audit Data

For auditors to conduct meaningful and effective audits, they must be provided with sufficient information about both the election outcome and the underlying system. The literature identifies several categories of information that should be made available:

1. Access to the source code of the e-voting system and the associated verifier software is widely considered essential for enabling thorough and independent audits [22,29,33,40,46,61].
2. Auditors should be provided with comprehensive system documentation to enable understanding of system behaviour, design decisions, and security properties [16,30,56].
3. The final tally and the corresponding ballot box must be accessible to auditors to enable verification of tally correctness [5,6,11,12,21,28,40,43,46,53,57,60,61]. This should be complemented by comprehensive audit logs from monitoring systems, individual components, and intermediate processing steps, allowing the audit to reconstruct and analyse the execution of the election process [18,37,40,42,49,51,59].
4. Audit data should include proof transcripts and other cryptographic artefacts generated during protocol execution, enabling verification of the correctness of the protocol and its underlying cryptographic operations [13,15,24,40,45,46,48–51,57].
5. Auditors should be provided with the inputs and outputs of the election process, together with sufficient metadata[6] to allow correct interpretation and validation of the audit data [2,13,21,34,40,48,57].

Discussion. The literature shows a high degree of agreement regarding the types of data that should be made available to auditors. Beyond protocol evidence, effective auditing requires comprehensive supporting information, including system documentation, audit procedures, and descriptions of audit artefacts, to enable correct interpretation and validation of the data.

These findings highlight that the availability of audit data alone is insufficient: the data must also be complete, well-structured, and accompanied by adequate contextual information. Without this, auditors may be unable to correctly interpret the evidence or verify the claimed properties of the system, limiting the practical effectiveness of the audit process.

Recommendations We derive the following recommendations regarding the information that should be made available to auditors:

[6] This includes, for example, randomisation seeds, authentication data such as cryptographic keys, and contextual information about the election, including ballots and candidates.

2A Provide complete audit data, including the final tally, intermediate results, and proof transcripts, together with sufficient metadata (e.g., checksums, digital signatures, and election context information) to ensure the data can be correctly interpreted and validated.
2B Provide comprehensive system documentation and access to the source code of both the e-voting system and the verifier software to enable thorough analysis and independent verification.

3.3 Verification Process

The findings related to the verification process cover both the methodology and tooling used to conduct audits. These requirements can be grouped into three main aspects: audit timing, audit execution, and audit reliability.

Audit Timing and Phases. The literature consistently recommends that audits should be performed at multiple stages of the election process:

1. An initial audit should be conducted after the setup phase, before voting begins, to verify that the system has been correctly initialised before any votes are cast [3,7,12,14,15,18,23,33,40,48,49,51].
2. After the tally phase, the correctness of the tally and the execution of the protocol should be verified [6,16,18,20,23,33,45,46,50,53,55,56].

When verification is performed after the tally phase, strict procedural ordering of tests is not required, enabling optimisations such as batch verification [33] and parallel execution [2]. Also, probabilistic techniques, such as random sampling, may be used to verify tally correctness in certain settings [3,18,19,51].

Audit Execution and Reporting. The following requirements concern how audits are conducted and reported:

3. Audits should be performed using dedicated verifier software to ensure systematic and reproducible verification [18–20,24,26,56–58].
4. The audit process should produce a verification report documenting the results [21,57].
5. To support accountability, the report should, where feasible, describe identified failures and quantify their extent [21].
6. Audits should be conducted by independent entities and involve relevant stakeholders to enhance transparency and reduce reliance on individual parties [16,32,58].
7. Auditors should possess sufficient expertise in cryptography and the underlying system to correctly interpret the verification process [31,58].

Limitations and Reliability of the Verification Process. The literature also highlights several factors that may compromise the soundness of the verification process:

8. The verification process may fail if not all planned audit steps are carried out [57].
9. Errors in the development of verifier software may introduce faults that remain undetected during audits [24].
10. Even when formally verified, the verification process itself may rely on assumptions or tools whose correctness is not guaranteed [44].

Discussion. The literature presents a largely consistent view of the verification process, emphasising structured audit phases, the use of dedicated verifier software, and the importance of comprehensive reporting. However, a key challenge lies in the soundness of the verification process itself.

In particular, the correctness of an audit depends not only on the availability of evidence, but also on the reliability of the tools and procedures used to verify it. This introduces a recursive dependency: the verifier software and audit methodology must themselves be trusted or independently validated. As a result, the verification process forms part of a broader chain of trust, in which weaknesses at any stage may undermine the overall assurance provided by the audit.

These findings highlight that achieving meaningful verification requires careful design of both the audit process and the supporting tooling, as well as explicit consideration of the assumptions underlying their correctness.

Recommendations. We derive the following recommendations for structuring and executing the verification process:

3A Perform an audit after the setup phase, prior to the start of voting, to ensure that the system has been correctly initialised. Voting should only proceed if no errors are detected.

3B Perform an audit after the tally phase to verify the correctness of the protocol execution and the final election outcome.

3C Validate the verifier software before using it in an audit. Test it against documented test cases, edge cases, and fault-seeded inputs, and record which verification claims are covered by this validation.

3D Ensure that the verification process produces a comprehensive audit report. In the event of verification failure, the report should clearly describe the nature and extent of the failure.

3E Ensure that both developers of verifier software and auditors using it possess sufficient expertise in cryptography and the underlying e-voting system.

3F Involve independent entities and relevant stakeholders in the audit process to enhance transparency and reduce reliance on any single party.

3.4 Trust Assumptions

All systems operate within a trust model, composed of trust assumptions that affect the e-voting system, verifier software, auditors, and audit infrastructure. The findings below concern the audit trust model and how trust assumptions interact with other security properties:

1. Trust assumptions on the auditors, the verifier software, and the method that validates it should be explicitly defined rather than implicitly assumed [49,58].
2. Trust assumptions could limit the verifiability of the system, but may be introduced to preserve other security properties, such as voter privacy, when direct verification would compromise them [52,58].
3. Laws, regulations, and election-specific operational constraints may affect how the audit trust model is constructed [16,23].
4. Independent audits should be performed to reduce the trust placed in the system vendor [10,61].
5. A verification test does not provide any guarantees when it does not align with the trust model [10,41,57].

Discussion. One identified tension concerns audit independence and reliance on trusted parties. Trusting auditors, verifier software, or vendor-controlled infrastructure can simplify the audit process and reduce complexity. However, concentrating trust in a limited set of actors weakens the audit's practical independence and increases the consequences of compromise or dishonesty.

A similar identified tension concerns transparency and audit independence. Vendor-provided verifier software and audit tooling facilitate audits, but reliance on vendor-controlled tools weakens the independence of the verification process. The vendor should therefore support independently developed verifier implementations to reduce this concentration of trust.

Finally, the literature distinguishes different forms of assurance provided by audit procedures: some verification steps are deterministic once the protocol assumptions and audit data are fixed, while others, such as sampling-based checks, provide probabilistic assurance. Cryptographic hardness assumptions belong to the protocol and verifier trust model and should be stated separately from the assurance provided by a particular audit step.

Recommendations. We derive the following recommendations regarding trust assumptions:

4A Clearly define the trust assumptions for the e-voting system, audit process, auditors, verifier software, and supporting infrastructure.

4B Treat vendor-provided verifier software as a reference tool, not as the trust anchor of the audit. Verification results should be reproducible with an independently developed verifier or with a verifier that has been independently validated.

4C For each verification procedure, state what is checked deterministically, what relies on probabilistic methods, and which protocol or verifier assumptions are required.

3.5 Documentation

System documentation provides auditors with an understanding of the e-voting system, while audit documentation defines how verification should be conducted. Together, they enable auditors to correctly interpret audit data and ensure that the verification process is sound. The literature identifies several requirements for documentation, which we group into four categories.

Documentation of Security Properties and Models. The first group concerns how system security properties and assumptions are specified:

1. Documentation should clearly state the security properties claimed by the system [29,42,56], and define them using precise and appropriate formalism or terminology [6,9,31].
2. Documentation should explain how these security properties are achieved, including the mechanisms and assumptions underlying them [6,28,31].

Documentation of System Behaviour. The second group concerns the description of system behaviour and structure:

3. Documentation should include both high-level and low-level descriptions of the protocol, system components, and overall election process [21,22,42,54,56].
4. All system algorithms should be specified in detail, preferably using pseudocode [16,18,22,29,31,33,42].

Algorithm specifications should define inputs, outputs, and execution context [42], decompose complex procedures into smaller components where appropriate [42], and clearly specify data formats and serialisation methods [16,18,22]. Relevant cryptographic parameters should also be explicitly defined [22].

Documentation of the Verification Process. The third group concerns how the audit process itself is specified:

5. Documentation should define the audit process and its outputs, with a focus on the verification tasks performed by the verifier [21].
6. The audit should be formalised as a structured set of verification tests, organised in a test catalogue, where each test corresponds to a specific verification requirement [21,58].
7. The roles, responsibilities, and interactions of all parties involved in the audit process should be clearly specified [12,18,23,30,31,42,57].
8. Where applicable, documentation should specify the possible degrees of failure for each verification test [21].

Documentation Maintenance and Consistency. The final group concerns ensuring that documentation remains reliable and aligned with the system:

9. The documentation, source code, and executable implementations should remain consistent with one another [21,22,24,29,30,42,56].
10. Documentation and implementation should be continuously maintained and updated as flaws are identified and corrected [42].

Discussion. The findings indicate that comprehensive documentation of the e-voting system, verifier software, and audit process is essential for enabling effective auditing. However, a fundamental trade-off arises between the level of detail and the maintainability of the documentation. While detailed and precise documentation improves clarity and supports correct verification, it also increases the effort required to ensure consistency with the evolving system.

Maintaining alignment between documentation, source code, and system behaviour becomes increasingly challenging as systems evolve, potentially introducing inconsistencies that undermine the reliability of the audit process. Although the literature proposes methods to improve readability and structure, such as modular descriptions and the use of pseudocode, these approaches primarily address presentation rather than reducing the underlying maintenance burden.

These findings highlight that documentation quality is not solely a matter of completeness, but also of evolvability. Ensuring that documentation remains accurate, consistent, and up to date over time is a critical requirement for supporting reliable and independent verification.

Recommendations. We derive the following recommendations for system and audit process documentation:

5A Document the system's claimed security properties, threat model, trust assumptions, and the mechanisms by which these guarantees are achieved.
5B Document the complete protocol structure, including algorithms, cryptographic mechanisms, data structures, component interactions, and the overall election process.
5C Specify the audit process in detail, including verifier responsibilities, verification procedures, and how individual tests collectively establish a complete verification chain.
5D Ensure that documentation remains accurate and reliable over time. Organise documentation so that auditors can always identify the authoritative source. Avoid duplicating definitions across documents, and state how inconsistencies between documents are resolved.
5E Improve clarity and reduce ambiguity by decomposing complex algorithms, using readable pseudocode, and defining explicit data formats and interfaces.

3.6 Implementability

Achieving independent verification in practice requires more than theoretical auditability: election authorities and system vendors must actively support third-party auditors and verifier implementers. The findings highlight several requirements for enabling practical and meaningful independent audits.

Support for Independent Auditors. The first group of findings concerns how election authorities and system vendors can facilitate independent audits:

1. Independent auditors should be permitted to audit the election and be provided with access to all non-confidential artefacts and relevant outputs [40,58].
2. Auditors should be given sufficient time to perform their work, while allowing adequate time for system vendors to address any identified issues [16,30,58].
3. Verifier software should be made available to support audit execution [3]. However, reliance solely on vendor-provided tools reduces audit independence, as it introduces a dependency on the same entity being evaluated [7,58].
4. Documentation of system algorithms and design should be publicly available to enable independent understanding and verification [1,8,15,16].
5. Data and documentation should be presented in a form that is accessible and understandable to potential auditors [56].

Verifier Development and Usability. The second group concerns requirements for enabling third-party development and validation of verifier software:

6. The barrier to entry for third-party implementers should be minimised by reducing the complexity of the verifier and verification process, for example by favouring simpler designs and widely understood tools [18,20,23,36,61].
7. Auditors should be provided with validation datasets, including fault-seeded inputs and edge cases, to enable testing and verification of audit tools [29,42,58].
8. Where feasible, verifier software should be derived from formally verified specifications to improve assurance in its correctness [24,26,27].
9. Third-party implementations should rely on transparent development processes, ensure long-term availability of verifier software, and account for operational constraints of the election context [42].

Incentives for Independent Auditing. The third group concerns mechanisms for encouraging independent participation:

10. A culture of independent auditing should be fostered throughout the development and deployment of the e-voting system [8].
11. Incentives should be provided to encourage third parties to perform audits [28,56]. For example, financial compensation or formal recognition mechanisms may be used to motivate participation [4].

Discussion. E-voting systems may fail to support meaningful independent audits if potential auditors cannot realistically implement, validate, or operate verifier software. This highlights a fundamental practical limitation: auditability depends not only on the availability of audit mechanisms, but on their accessibility and usability by independent parties.

A key tension arises between facilitating independent audits and protecting sensitive information. Enabling independent verification requires the disclosure of election artefacts and system details, which may be restricted by legal or regulatory constraints. When such constraints limit access to audit data, they reduce the pool of authorised auditors and thereby weaken the practical independence of the audit process.

As a result, trust becomes concentrated in a limited set of privileged actors who are permitted to access the necessary information. This introduces a trade-off between transparency and regulatory compliance, where stronger privacy or legal guarantees may come at the cost of reduced audit independence.

Recommendations. We derive the following recommendations for enabling independent audits and third-party verifier implementations:

6A Provide independent auditors with access to all non-confidential audit artefacts, technical specifications, and verifier tooling necessary to conduct meaningful verification.
6B Provide sufficient time before the election for independent auditors to implement, validate, and test verifier software.
6C Publish verifier software, technical specifications, and audit artefacts in publicly accessible and understandable formats to support independent implementation.
6D Reduce technical barriers to independent auditing by favouring simple verifier designs, readable specifications, and widely used, standardised algorithms where possible.
6E Provide complete validation datasets, including edge cases and fault-seeded inputs, to enable third parties to test and verify their implementations.
6F Encourage sustained third-party auditing through transparent development processes and appropriate incentives.

4 Concluding Remarks

This work shows that achieving universal verifiability in e-voting systems depends not only on protocol design but also on practical auditability factors, including documentation quality, the availability of audit data, and support for independent verification. Systems must enable independent parties to verify election outcomes in practice, thereby reducing reliance on system vendors and election authorities. In particular, achieving practical verifiability requires treating auditability and implementability as prioritised design goals.

A central finding of this study is that existing research provides complementary but fragmented guidance. While prior work proposes audit frameworks, verification methods, and implementation practices, few contributions address how these aspects must be combined to support end-to-end, independently verifiable elections. By consolidating these perspectives, we provide a unified framework of recommendations applicable across a broad range of system designs.

Our analysis highlights several fundamental tensions. There is a trade-off between transparency and regulatory or operational constraints, as independent audits require access to artefacts that may be restricted by law. Similarly, there is a tension between system complexity and auditability. Simplifying documentation can improve accessibility for auditors, but may introduce inaccuracies or omissions, while maintaining consistent documentation requires additional effort.

Overall, these recommendations should not be interpreted as a checklist whose items must always be maximised simultaneously; their practical application requires explicit prioritisation and careful balancing of the associated trade-offs within the specific election context.

Future work should focus on improving the usability of verifier software, reducing the expertise required for auditing, developing effective incentives for independent auditors, and balancing transparency requirements with legal and operational constraints.

Acknowledgments. We thank Céline Camacho, Matthieu Decotignie, Christopher Klahn, and Baptiste Sottas for their constructive feedback.

Disclosure of Interests. Audhild Høgåsen and Chiara Spadafora are employed by Swiss Post, which develops an e-voting system. They have no competing interests here.

References

1. Al-Shammari, A.F.N., Villafiorita, A., Weldemariam, K.: Understanding the development trends of electronic voting systems. In: ARES (2012). https://doi.org/10.1109/ARES.2012.76
2. Antonyan, T., et al.: State-wide elections, optical scan voting systems, and the pursuit of integrity. IEEE TIFS (2009). https://doi.org/10.1109/TIFS.2009.2033232
3. Arafat, S.M.: On the Estonian internet voting system, IVXV, SoK and suggestions. Cryptology ePrint Archive (2025). https://eprint.iacr.org/2025/506,
4. B, S., V, R.T., Krishna M P, N., J, B.R., Arvindh M, S., Alagappan, M.: Secured electronic voting system using the concepts of blockchain. In: IEMCON (2019). https://doi.org/10.1109/IEMCON.2019.8936310
5. Babenko, L., Pisarev, I.: E-voting system based on multiple ballot casting. In: ICICT (2020). https://doi.org/10.1007/978-981-32-9343-4_9
6. Basin, D., Radomirovic, S., Schmid, L.: Alethea: a provably secure random sample voting protocol. In: IEEE CSF (2018). https://doi.org/10.1109/CSF.2018.00028
7. Benaloh, J., Naehrig, M., Pereira, O.: REACTIVE: rethinking effective approaches concerning trustees in verifiable elections. In: E-Vote-ID (2026). https://doi.org/10.1007/978-3-032-05036-6_2

8. Buckland, R., Teague, V., Wen, R.: Towards best practice for e-election systems: lessons from trial and error in Australian elections. In: E-Voting and Identity (2012). https://doi.org/10.1007/978-3-642-32747-6_14
9. Cansell, D., Gibson, J., Méry, D.: Refinement: a constructive approach to formal software design for a secure e-voting interface. ENTCS (2007). https://doi.org/10.1016/j.entcs.2007.01.060
10. Chang-Fong, N., Essex, A.: The cloudier side of cryptographic end-to-end verifiable voting: a security analysis of Helios. In: ACSAC (2016). https://doi.org/10.1145/2991079.2991106
11. Chaum, D., Ryan, P., Schneider, S.: A practical voter-verifiable election scheme. In: ESORICS (2005). https://doi.org/10.1007/11555827_8
12. Chondros, N., et al.: D-DEMOS: a distributed, end-to-end verifiable, Internet Voting system. In: IEEE ICDCS (2016). https://doi.org/10.1109/ICDCS.2016.56
13. Clarkson, M., Chong, S., Myers, A.: Civitas: toward a secure voting system. In: IEEE S&P (2008). https://doi.org/10.1109/SP.2008.32
14. Cortier, V., Debant, A., Gaudry, P.: Breaking verifiability and vote privacy in CHVote. In: ESORICS (2026). https://doi.org/10.1007/978-3-032-07901-5_5
15. Cyrus, M., Tiwari, M.: Formally verified verifiable group generators. In: FSEN (2025). https://doi.org/10.1007/978-3-031-87054-5_12
16. Driza Maurer, A.: The Swiss Post/Scytl transparency exercise and its possible impact on internet voting regulation. In: E-Vote-ID (2019). https://doi.org/10.1007/978-3-030-30625-0_6
17. Erdmanis, J.: Unconditional individual verifiability with receipt freeness via post-cast isolation. Cryptology ePrint Archive (2025). https://eprint.iacr.org/2025/1186
18. Essex, A., Clark, J., Hengartner, U., Adams, C.: Eperio: mitigating technical complexity in cryptographic election verification. In: USENIX EVT/WOTE (2010). https://eprint.iacr.org/2012/178
19. Garera, S., Rubin, A.: An independent audit framework for software dependent voting systems. In: ACM CCS (2007). https://doi.org/10.1145/1315245.1315278
20. Gawel, D., Kosarzecki, M., Vora, P., Wu, H., Zagórski, F.: Apollo - end-to-end verifiable internet voting with recovery from vote manipulation. In: E-Vote-ID (2016). https://doi.org/10.1007/978-3-319-52240-1_8
21. Haenni, R., Dubuis, E., König, R.E., Locher, P.: Process models for universally verifiable elections. In: E-Vote-ID (2018). https://doi.org/10.1007/978-3-030-00419-4_6
22. Haenni, R., Dubuis, E., König, R.E., Locher, P.: CHVote: sixteen best practices and lessons learned. In: E-Vote-ID (2020). https://doi.org/10.1007/978-3-030-60347-2_7
23. Haenni, R., König, R.E., Locher, P., Dubuis, E.: CHVote protocol specification. Cryptology ePrint Archive (2017). https://eprint.iacr.org/2017/325
24. Haines, T., Goré, R., Stodart, J.: Machine-checking the universal verifiability of ElectionGuard. In: NordSec (2021). https://doi.org/10.1007/978-3-030-70852-8_4
25. Haines, T., Lewis, S., Pereira, O., Teague, V.: How not to prove your election outcome. In: IEEE S&P (2020). https://doi.org/10.1109/SP40000.2020.00048
26. Haines, T., Goré, R., Sharma, B.: Did you mix me? Formally verifying verifiable mix nets in electronic voting. In: IEEE S&P (2021). https://doi.org/10.1109/SP40001.2021.00033
27. Haines, T., Goré, R., Tiwari, M.: Verified verifiers for verifying elections. In: ACM CCS (2019). https://doi.org/10.1145/3319535.3354247
28. Haines, T., Rose, J.: Vestigial vulnerabilities in deployed verifiable E-voting systems. Cryptology ePrint Archive (2025). https://eprint.iacr.org/2025/2033

29. Haines, T., Rønne, P.: New standards for E-voting systems: reflections on source code examinations. In: FC 2021 International Workshops (2021). https://doi.org/10.1007/978-3-662-63958-0_24
30. Halderman, J., Teague, V.: The New South Wales iVote System: security failures and verification flaws in a live online election. In: VoteID (2015). https://doi.org/10.1007/978-3-319-22270-7_3
31. Heiberg, S., Martens, T., Vinkel, P., Willemson, J.: Improving the verifiability of the Estonian internet voting scheme. In: E-Vote-ID (2017). https://doi.org/10.1007/978-3-319-52240-1_6
32. Jafar, U., Ab Aziz, M., Shukur, Z.: Blockchain for electronic voting system: review and open research challenges. Sensors (2021). https://doi.org/10.3390/s21175874
33. Jivanyan, A., Feickert, A.: Aura: private voting with reduced trust on tallying authorities. Cryptology ePrint Archive (2022). https://eprint.iacr.org/2022/543
34. Kiayias, A., et al.: Tampering with special purpose trusted computing devices: a case study in optical scan e-voting. In: ACSAC (2007). https://doi.org/10.1109/ACSAC.2007.16
35. Kremer, S., Ryan, M., Smyth, B.: Election verifiability in electronic voting protocols. In: ESORICS (2010). https://doi.org/10.1007/978-3-642-15497-3_24
36. Kulyk, O., Volkamer, M.: Usability is not enough: lessons learned from 'human factors in security' research for verifiability. In: E-Vote-ID (2018). https://eprint.iacr.org/2018/683
37. Kumar, A., Choudhary, M., Singh, A.R., Kumar, S., Thakur, A.: Permissioned smart contract based e-voting system for university. In: NETCRYPT (2025). https://doi.org/10.1109/NETCRYPT65877.2025.11102528
38. Küsters, R., Müller, J.: Cryptographic security analysis of E-voting systems: achievements, misconceptions, and limitations. In: E-Vote-ID (2017). https://doi.org/10.1007/978-3-319-68687-5_2
39. Lundin, D., Ryan, P.: Human-readable paper verification of Prêt à Voter. In: ESORIC (2008). https://doi.org/10.1007/978-3-540-88313-5_25
40. Martínek, T., Pelikán, M., Tyrychtr, J., Konopásek, J.: Implementation of a secret and verifiable personal remote electronic election of an agrarian organization per the Recommendation of the Council of Europe. Agris on-line Papers Econ. Info. (2024). https://doi.org/10.7160/aol.2024.160305
41. McMurtry, E., Pereira, O., Teague, V.: When is a test not a proof? In: ESORICS (2020). https://doi.org/10.1007/978-3-030-59013-0_2
42. Moser, F., Grimm, R., Hilt, T., Kirsten, M., Niederbudde, C., Volkamer, M.: Recommendations for implementing IV in internet voting. In: E-Vote-ID (2024). https://doi.org/10.18420/e-vote-id2024_02
43. Neumann, S., Olembo, M., Renaud, K., Volkamer, M.: Helios verification: to alleviate, or to nominate: is that the question, or shall we have both? In: EGOVIS (2014). https://doi.org/10.1007/978-3-319-10178-1_20
44. Oechsner, S., Pereira, V., Scholl, P.: Who verifies the verifiers? Lessons learned from formally verified line-point zero-knowledge. IACR Communications in Cryptology (2025). https://doi.org/10.62056/a0wa0lmol
45. Panja, S., Roy, B.K.: A secure end-to-end verifiable e-voting system using zero knowledge based blockchain. Cryptology ePrint Archive (2018). https://eprint.iacr.org/2018/466
46. Perez, A.J., Ceesay, E.N.: Improving end-to-end verifiable voting systems with blockchain technologies. In: IEEE Cybermatics (2018). https://doi.org/10.1109/Cybermatics_2018.2018.00202

47. Rivest, R.L.: On the notion of 'software independence' in voting systems. Philos. Trans. Royal Soc. A: Math. Phys. Eng. Sci. **366**(1881) (2008). https://doi.org/10.1098/rsta.2008.0149
48. Ryan, P.: Prêt à Voter with Paillier encryption. Math. Comput. Model. (2008). https://doi.org/10.1016/j.mcm.2008.05.015
49. Ryan, P.: The computer ate my vote. Formal Meth.: State Art New Direct. (2010). https://doi.org/10.1007/978-1-84882-736-3_5
50. Ryan, P., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt à Voter: a voter-verifiable voting system. IEEE TIFS (2009). https://doi.org/10.1109/TIFS.2009.2033233
51. Ryan, P., Teague, V.: Pretty good democracy. In: Security Protocols (2013). https://doi.org/10.1007/978-3-642-36213-2_15
52. Rønne, P., Ryan, P., Smyth, B.: Cast-as-intended: a formal definition and case studies. In: FC (2021). https://doi.org/10.1007/978-3-662-63958-0_22
53. Shahandashti, S., Hao, F.: DRE-IP: a verifiable e-voting scheme without tallying authorities. In: ESORICS (2016). https://doi.org/10.1007/978-3-319-45741-3_12
54. Sturton, C., Jha, S., Seshia, S., Wagner, D.: On voting machine design for verification and testability. In: ACM CCS (2009). https://doi.org/10.1145/1653662.1653719
55. Suharsono, T.N., Gunawan, Sukmana, R.N., Agusiady, R.R., Garnida, H., Nasrudin, I.: Univer metric to measure the degree of universal verifiability in E-voting. In: IEEE TSSA (2024). https://doi.org/10.1109/TSSA63730.2024.10863897
56. Sutopo, A., Haines, T., Rønne, P.: On the auditability of the Estonian IVXV system and an attack on individual verifiability. In: FC International Workshops (2023). https://doi.org/10.1007/978-3-031-48806-1_2
57. Treier, T., Duuna, K.: Identifying and solving a vulnerability in the Estonian internet voting process: subverting ballot integrity without detection. IEEE Access (2024). https://doi.org/10.1109/ACCESS.2024.3521337
58. Treier, T.: Beyond the happy path: a safety-critical audit methodology for Estonia's I-voting processing application. IEEE Access (2025). https://doi.org/10.1109/ACCESS.2025.3638663
59. Weldemariam, K., Kemmerer, R., Villafiorita, A.: Formal analysis of an electronic voting system: an experience report. J. Syst. Softw. (2011). https://doi.org/10.1016/j.jss.2011.03.032
60. Xia, Z., Tong, Z., Xiao, M., Chang, C.: Framework for practical and receipt-free remote voting. IET IS (2018). https://doi.org/10.1049/iet-ifs.2017.0213
61. Yang, Y., Guan, Z., Wan, Z., Weng, J., Pang, H., Deng, R.: PriScore: blockchain-based self-tallying election system supporting score voting. IEEE TIFS (2021). https://doi.org/10.1109/TIFS.2021.3108494

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





FiltrumVote: Scalable, Verifiable, and Coercion-Resistant Internet Voting

Leonardo Kimura[1(✉)], Dimitri Leskow[1], Diego F. Aranha[3], Roberto Araújo[2], and Marcos Simplicio[1]

[1] University of Sao Paulo, Sao Paulo, Brazil
{lkimura,mjunior}@larc.usp.br
[2] Federal University of Para, Belem, Brazil
[3] Aarhus University, Aarhus, Denmark

Abstract. Internet voting can increase voter turnout and reduce operational costs; however, it also facilitates voter coercion. One compelling approach to address this issue is deniable vote updating, which allows voters to cast multiple ballots while counting only their final vote. Despite its intuitive appeal, achieving deniable vote updating together with verifiability and scalability has proven challenging. Therefore, we present FiltrumVote, a scalable, publicly verifiable, and coercion-resistant Internet voting system based on deniable vote updating. It has a quasi-linear cleansing phase, and it is built mainly from zero-knowledge proofs with standard $\Sigma-$protocol techniques. As a result, FiltrumVote is highly scalable: a standard computer takes less than 5 h to prove all zero-knowledge proofs for one million votes.

Keywords: Coercion-resistant voting · End-to-end verifiability · Encrypted sorting · Range proofs

1 Introduction

Internet voting has the potential to increase voter turnout, broaden accessibility, and reduce costs. Consequently, numerous protocols have been proposed for secure Internet-based elections, using cryptographic techniques to balance privacy, verifiability, and correctness. However, one critical requirement for secure elections is coercion-resistance: adversaries should not be able to pressure voters to vote for particular candidates. The remote nature of Internet voting significantly increases the risk of coercion compared to traditional, in-person voting environments.

A variety of approaches have been developed to mitigate coercion in Internet-based election systems. One of the earliest and most influential strategies is the fake-credential mechanism, first introduced in JCJ [15]. In this model, voters can generate fake credentials that are indistinguishable from their real ones, so that fake credentials can be handed to a coercer, while genuine credentials are used to cast a valid ballot. However, this model presents significant usability

© The Author(s) 2027
D. Duenas-Cid et al. (Eds.): E-Vote-ID 2026, LNCS 17083, pp. 108–124, 2027.
https://doi.org/10.1007/978-3-032-39512-2_7

challenges, since voters must confidently provide only the fake credentials to an adversary, while remembering to use the legitimate credentials only to cast valid votes. Although later works improved scalability, usability, and security of this approach [4–6], empirical studies show that the burden on usability is still substantial [7,23].

An alternative coercion-resistance strategy that has gained considerable attention is deniable vote updating (or revoting) [23]. In this model, voters can cast their ballots multiple times, and only the final vote is counted. This provides an intuitive strategy to resist coercion, and has inspired several real-world protocols, including the Internet voting system used in Estonia [25]. Yet achieving deniable vote updating while maintaining verifiability is challenging. In particular, the system must not reveal how many times an individual voter has voted; otherwise, a coercer could confirm whether a voter has updated a ballot or not after the coercion. Thus, traditional tallying mechanisms, which publish all valid votes, are inadequate. Existing solutions attempt to reconcile these requirements using noise votes [12,19], credential-hiding cleansing [1,18], or periodic vote updating [22]. However, these designs either impose substantial computational overhead [1,18,22] or rely on strong trust assumptions [12].

Our contribution. We propose FiltrumVote, an Internet voting protocol that simultaneously achieves verifiability, scalability, and coercion resistance through deniable vote updating. While prior work has explored verifiable deniable vote updating, to the best of our knowledge, FiltrumVote is the first protocol to scale to more than one million votes using only modest computational resources. The prover time, verification time, and proof size all grow linearly with the number of votes, with the sole exception of the sorting phase, which relies on a standard quasi-linear sorting algorithm that can be performed in plaintext. FiltrumVote is inspired by CHide [3,6], an enhancement of the JCJ voting system that employs a less revealing tally procedure. We adapt this approach to support revoting, thus reducing the responsibilities of trustees and substantially lowering both system complexity and computational overhead.

2 Related Works on Deniable Vote Updating

Revoting, or the ability for voters to update their votes, is already supported in many real-world deployed systems, such as Helios [2] and the Estonian Voting System (IVXV) [13]. Thus, voters have an intuitive mechanism to mitigate coercion attacks, offering clear usability advantages over alternative approaches (e.g., fake credentials) [23]. However, these systems provide only superficial protection against coercion. For example, information such as signed tracking codes or transaction logs may reveal how many votes a voter has cast [25]. Consequently, voters cannot hide whether they have updated their vote—a property known as *deniable vote updating*. As a result, a determined coercer can verify whether an attempted coercion was successful.

The first works that provide both verifiability and deniable vote updating are [1] and [18], which use plaintext equivalence tests (PETs) to enable a credential-hiding tallying procedure. In these schemes, distributed authorities compare all pairs of ballots and privately mark only the last ballot associated with each credential. However, this approach results in an $O(n^2)$ protocol, making it suitable only for small elections.

A subsequent work is VoteAgain [19], which hides ballots in a swarm of *dummy* ballots. Dummy and overwritten ballots are secretly yet verifiably canceled, leaving only valid ballots in the final tally. This approach results in a quasi-linear cleansing time $O(n \log(n))$. However, VoteAgain relies on strong trust assumptions to achieve public verifiability. In particular, even in the improved protocol [12], vote integrity holds only if there is no malicious collusion between the polling authority and the tally server. Otherwise, votes can be modified or discarded without detection.

Another approach by Farzaliyev and Willemson is introduced in [10], which combines private bulletin boards with vector commitments and Succinct Non-Interactive Argument of Knowledge (SNARKs). This results in a solution with extremely reduced verification time and a relatively small proof size. However, this efficiency comes at the cost of generating SNARK proofs. For example, processing 2^{24} votes would require a peak memory usage of approximately 500 GB. In addition, the system requires 16 TB to store some individual proofs, which requires 41 core-days to verify sequentially.

Finally, DeVoS achieves coercion-resistance by employing periodic vote updating [22]. In each microsubmission interval, all votes are reencrypted or updated in zero-knowledge, thus hiding whether the voter has revoted or not. This results in a cleansing time linear to the number of votes $O(n)$. However, DeVoS' scalability is limited by the need to generate many zero-knowledge proofs in *every* microsubmission interval. For example, a server with 16 GB of RAM can generate proofs for only $\approx 10,000$ to $80,000$ votes within a two-minute microsubmission interval. In addition, this estimate assumes only four candidates; supporting additional candidates further increases the computational cost. Even with optimizations, handling elections with over one million votes would require substantial computational resources.

3 FiltrumVote Model

In this section, we present a high-level description of FiltrumVote.

3.1 System and Threat Model

The participants of the FiltrumVote protocol are:

– *Election Authority (EA).* The entity responsible for conducting the election, including registering and authenticating voters, receiving, and processing ballots.

- *Trustees.* A set of authorities that mix and decrypt the votes.
- *Public Bulletin Board (PBB).* An append-only data structure where the EA publishes the list of received votes, and where both the EA and the trustees publish processed votes and associated proofs. The contents of the PBB are publicly readable at all times.
- *Voters.* Individuals who cast their ballots by interacting with the EA.
- *Auditors.* Independent parties responsible for verifying the correctness of the election, including the validity of the zero-knowledge proofs.

FiltrumVote assumes the following key trust assumptions:

1. The PBB is honest, that is, it never removes data and provides a consistent view of the board to all participants. Several approaches can be used to achieve this (e.g., [8,14,16]).
2. Voters securely store and protect their private keys.
3. Voters verify that their public keys are correctly registered.
4. There exists a public list of eligible voters (voter roll), and any participant can verify its correctness.
5. The voting device is fully trusted with respect to both integrity (e.g., through a second verification device [20]) and privacy.
6. The voter can cast at least one vote after coercion attempts and before the end of the election.

Although the second assumption is strong, it is standard in the state-of-the-art Internet voting protocols such as DeVos and JCJ. One approach to protect the keys is to store them on secure hardware devices; however, this facilitates forced abstention attacks, as the coercer may simply confiscate the device. Another mitigation is to reduce the risk of voter impersonation by requiring extra authentication with the EA when casting a vote (e.g., biometrics), so that the private key alone is not enough to cast a vote. Fully reducing this trust assumption remains an open problem.

We also assume that a voter can register their public key with the EA through an authenticated channel. In addition, we assume the voter can send one vote to the EA through a private channel, so that the adversary cannot learn whether or when a vote has been cast. These assumptions are standard in Internet voting protocols, such as DeVos.

Finally, we require our system to satisfy the following properties, defined informally:

- *Ballot secrecy.* Assuming a threshold number of trustees are honest, no collusion of malicious parties can learn the vote of any honest voter.
- *Verifiability.* Any party can verify that the reported election outcome is correct, even under collusion between a malicious EA and malicious trustees. In particular, ballot-stuffing attacks are detectable.
- *Coercion-resistance.* Assuming an honest EA and a threshold number of honest trustees, no coercer can determine whether a coercion attack was successful. In particular, forced abstention attacks are not possible.

Notably, we assume that the EA is honest with respect to coercion resistance. A malicious EA could carry out a successful coercion attack, for example, by learning how many times a voter has cast a ballot or by refusing to accept new votes after a coerced one. To mitigate this risk, the cleansing phase must be conducted privately (maybe in a controlled environment without human intervention), and any refusal to accept new votes should be investigated. This assumption is standard in the literature: other Internet voting protocols, such as DeVos and VoteAgain, likewise have a party fully trusted with coercion resistance.

3.2 High-Level Overview

Here, we provide a high-level overview of FiltrumVote. Intuitively, voters submit their ballots along with their encrypted credentials. The cleansing phase uses the credentials to identify and remove overwritten votes in five steps. Each step is done privately and verifiably using zero-knowledge proofs.

We denote by $[\![x]\!]$ an encryption of x under the EA's public key, and by $[\![x]\!]_T$ an encryption of x under the threshold-generated trustees' public key.

Registration Phase. Each voter with identifier vid_i generates a key pair (sk_i, pk_i) and sends their public key pk_i to the EA through an authenticated channel. The EA then assigns to the voter a unique credential number σ_i, and publishes the voter roll (vid_i, pk_i, σ_i). Auditors check if every σ_i is unique and every vid_i is from an eligible voter; voters check if their public key pk_i is correct.

Submission Phase. The voter V_i encrypts her individual vote v_i under the trustees' public key pk_T, resulting in $[\![v_i]\!]_T$. She also encrypts her public key pk_i and her credential number σ_i under EA's public key, resulting in $[\![pk_i]\!], [\![\sigma_i]\!]$. Then, the voter provides an *non-interactive zero-knowledge proof (NIZKP) of possession* π_{poss} (Sect. 4.1). This proof essentially ensures that the voter knows the secret key associated with the encrypted public key; i.e., that she knows sk_i. The voter also computes an NIZKP of plaintext knowledge π_{kw} for $[\![v_i]\!]_T$ and $[\![\sigma_i]\!]$. The resulting ballot $\beta = ([\![v_i]\!]_T, [\![pk_i]\!], [\![\sigma_i]\!], \pi_{poss}, \pi_{kw})$ is sent to the EA. The EA checks the ballot (its proof and whether it is unique) and sends it to the public bulletin board, PBB. Voters can vote multiple times, and only the last vote will be counted.

Note that π_{poss} is quite a permissive proof: it only proves that the ciphertext encrypts *a public key*, and the voter knows the corresponding secret key. Thus, a malicious voter may use a new, ephemeral public key or add to the ballot an encrypted credential number totally unrelated to her credential. At this point, we need to consider two kinds of attacks.

First, an *impersonation attack* may use a credential σ_i from a valid voter, while using a public key different from that of the valid voter, i.e., $pk' \neq pk_i$. To protect against it, we require the EA to check every submitted ballot during the submission phase. Thus, the EA must decrypt the $[\![pk']\!]$ and $[\![\sigma']\!]$ of submitted

ballots, and only accept them if the pair (pk', σ') is present in the voter roll. Note that EA is still considered untrusted; the correct behavior of the EA can be confirmed in the *consistency check step* (explained later).

Second, a *ballot stuffing attack* may use a credential σ' not existent in the voter roll. To protect against it, the *selection step* ensures that only ballots from eligible voters are considered (i.e., with credentials present in the voter roll). Thus, all other votes are verifiably canceled, even by a malicious EA. However, an honest EA might also reject these invalid votes in the submission phase to reduce the number of votes to be processed later.

Cleansing Phase - Preparation. Once the voting phase is finished, the EA starts the cleansing phase by processing all entries. First, the EA takes the submitted ballots $\beta_i = (\llbracket v_i \rrbracket_T, \llbracket pk_i \rrbracket, \llbracket \sigma_i \rrbracket)$ from the PBB in chronological order. Each ballot is appended with an encrypted zero (acting as a flag, $\llbracket f_i \rrbracket = \llbracket 0 \rrbracket$) and an encrypted sequential counter ($\llbracket ctr_i \rrbracket$).

Next, the EA augments this list by generating "flag tuples" derived from the voter roll. For each entry in the voter roll, the EA encrypts the voter's public key pk_i and credential σ_i. Then, it appends it with an encrypted dummy vote ($\llbracket v_i \rrbracket_T = \llbracket 1 \rrbracket$), an encrypted flag set to one ($\llbracket f_i \rrbracket = \llbracket 1 \rrbracket$), and the continuing encrypted counter ($\llbracket ctr_i \rrbracket$).

At the end, the actual ballots and flag tuples are in the format $t_i = (\llbracket v_i \rrbracket_T, \llbracket pk_i \rrbracket, \llbracket \sigma_i \rrbracket, \llbracket f_i \rrbracket, \llbracket ctr_i \rrbracket)$. Figure 1 illustrates the resulting board.

Cleansing Phase - Mixing. The EA mixes the list of tuples using a verifiable shuffle protocol.

Cleansing Phase - Sorting. The EA decrypts the credential σ_i, marker f_i, and the counter ctr_i of each tuple, and performs a sorting algorithm on the mixed list of tuples. To do so, we use the following relationship: for two tuples, t_i and t_j, we consider $t_i < t_j$ when the binary string $\sigma_i || f_i || ctr_i$ is less than $\sigma_j || f_j || ctr_j$.

This results in a list in which tuples from the same credentials are grouped and sorted chronologically. In addition, if a ballot was cast with an authorized credential (i.e., σ_i is in the voter roll), the last tuple from each credential group is a "flag tuple" (with $f_i = 1$).

Note that the EA can decrypt σ_i, f_i, ctr_i to perform the sorting in plaintext; however, it cannot reveal the plaintexts without compromising coercion resistance. Thus, after the sorting, the EA will provide a *NIZKP of greater-than* for every two consecutive tuples (Sect. 4.1). External auditors can check that the encrypted set is correctly sorted by verifying these proofs. Note that it is safe to leak the comparison result $t_i < t_j$ due to the previous mixing phase.

Cleansing Phase - Consistency Check. The EA proves that there are no impersonation attacks in the list by providing a *NIZKP of consistency* between every two consecutive tuples (Sect. 4.1). This proof shows that either both public

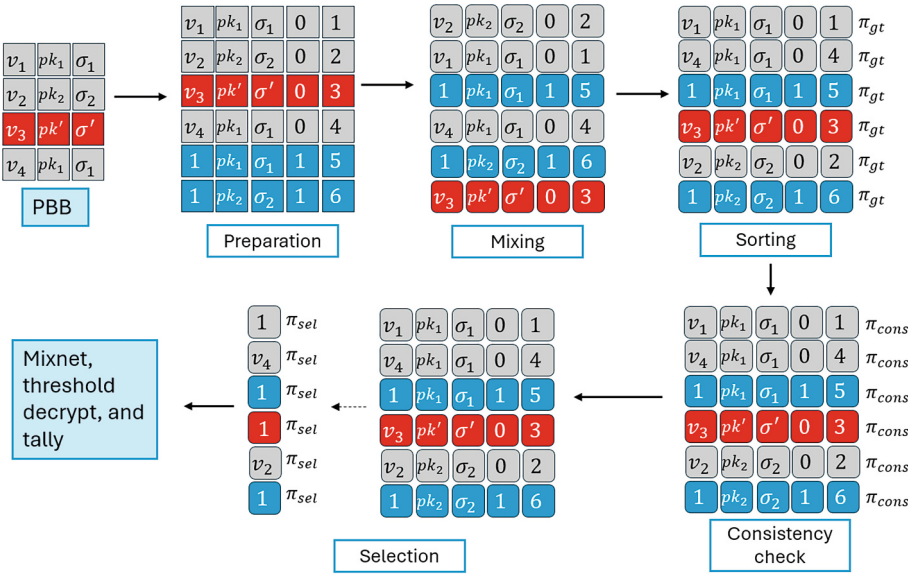


Fig. 1. High-level overview of the FiltrumVote cleansing phase. In this example, three tuples corresponding to valid votes are present: two submitted by voter (pk_1, σ_1) and one by voter (pk_2, σ_2). There is also a ballot-stuffing attempt (pk', σ'), shown in red. The blue entries correspond to the "flag tuples" generated from the voter roll.

keys and credentials are equal, or both are different, i.e., $(pk_i = pk_j \wedge \sigma_i = \sigma_j) \vee (pk_i \neq pk_j \wedge \sigma_i \neq \sigma_j)$.

To see that these *NIZKPs* effectively prove that no impersonation attacks are present in the list, consider the following: a tuple t' representing an impersonation attack has (1) an existing credential σ_i, and (2) a public key $pk' \neq pk_i$. Because of the sorting phase, all credentials are grouped: so t' is consecutive to some tuple t with credential σ_i and public key pk_i. Since $\sigma_i = \sigma_i$ and $pk' \neq pk_i$, it is impossible to provide a correct *NIZKP of consistency* for this tuple.

Cleansing Phase - Selection. For each tuple in the list, the EA generates a new encrypted vote V_i' that is either (1) a reencryption of $[\![v_i]\!]_T$, if the tuple is valid; or (2) an encryption of 1, if the tuple is invalid. Valid tuples are defined as those that immediately precede their corresponding "flag tuple" (i.e., $\sigma_1 = \sigma_2 \wedge f_2 = 1$). To demonstrate that each V_i' is correctly constructed, the EA provides a *NIZKP of selection* (Sect. 4.1).

This procedure effectively removes double votes, since only the last tuple from each credential group is retained. Moreover, ballot-stuffing tuples are rendered ineffective, as they do not have a corresponding "flag tuple".

Tally Phase. The trustees take the resulting list of encrypted votes and apply a mixnet. Every vote is then threshold decrypted, and invalid votes (i.e., votes

with decrypts to 1) are discarded. The remaining votes are tallied using standard techniques.

3.3 Design Decisions

Note that votes from *impersonation* and *ballot-stuffing* attacks are handled during the cleansing phase of the protocol, where NIZKPs are used to prove their correct treatment. An alternative approach would be to prevent such ballots in the submission phase already, by employing a stricter set of NIZKPs. For example, one could combine set-membership proofs with proofs of knowledge of discrete-logarithm representations, as in [17,18]. However, these proofs are significantly more complex than those currently used in FiltrumVote and would substantially increase both proof size and verification time. Instead, we opt for NIZKPs that can be constructed largely from standard Σ-protocols, thereby simplifying the work of both the EA and independent auditors.

4 Detailed Specification

In this Section, we present a detailed description of FiltrumVote.

Notation. Let λ denote the security parameter. Let $\mathbb{G}_p$ be a cyclic group of prime order p with generator g, and h be an additional independently chosen generator of $\mathbb{G}_p$. We write $a \in_R A$ to denote that a is sampled uniformly at random from the set A. We denote by $H(\cdot)$ a cryptographic hash function.

4.1 Cryptographic Building Blocks

FiltrumVote uses cryptographic primitives commonly found in modern secure e-voting proposals, such as ElGamal encryption and verifiable shuffle protocols. The main exceptions are the specially designed NIZKPs. Nevertheless, these proofs are built using standard Σ-protocol techniques combined with the Fiat-Shamir heuristic. FiltrumVote also employs range proofs, which are less common in e-voting solutions but widely used in blockchain and other zero-knowledge-based protocols [21].

ElGamal Public-Key Encryption Scheme. The ElGamal encryption can be presented in two forms: the multiplicative variant $\mathcal{E} = (\text{KeyGen}, \text{Enc}, \text{Dec})$ and the exponential variant $\mathcal{E}^\star = (\text{KeyGen}^\star, \text{Enc}^\star, \text{Dec}^\star)$. In both key generation algorithms, $\text{KeyGen}(\lambda)$ and $\text{KeyGen}^\star(\lambda)$, have the same specification, i.e., sample a secret key $\mathsf{sk} \in_R \mathbb{Z}_p$, compute the public key $\mathsf{pk} = g^{\mathsf{sk}}$, and output $(\mathsf{sk}, \mathsf{pk})$.

In the multiplicative variant, a message $m \in \mathbb{G}_p$ is encrypted by sampling $r \in_R \mathbb{Z}_p$, and outputting the ciphertext $\mathsf{ct} = (a, b)$, with $a = g^r$ and $b = m \cdot \mathsf{pk}^r$. Decryption of (a, b), is performed by computing $m \leftarrow b \cdot a^{-sk}$.

In the exponential variant, a message $m \in \mathbb{Z}_p$ is encrypted by sampling $r \in_R \mathbb{Z}_p$ and outputting the ciphertext $\mathsf{ct} = (a, b)$, with $a = g^r$ and $b = g^m \cdot \mathsf{pk}^r$). Decryption of (a, b) proceeds by computing $m \leftarrow \log_g(b \cdot a^{-\mathsf{sk}})$. Since it requires solving a discrete logarithm, this form is practical only for small values of m.

Both exponential and multiplicative ElGamal ciphertexts (a, b) support re-randomization: sampling $r' \in_R \mathbb{Z}_p$, one computes $(a', b') = (a \cdot g^{r'}, b \cdot \mathsf{pk}^{r'})$. Moreover, the exponential variant is additively homomorphic, i.e., component-wise multiplication of two ciphertexts results in an encryption of the sum of the corresponding plaintexts

$$\mathrm{Enc}^\star(m_1, r_1) \cdot \mathrm{Enc}^\star(m_2, r_2) = \mathrm{Enc}^\star(m_1 + m_2, r_1 + r_2).$$

NIZKP Proofs. Non-interactive zero-knowledge proofs (NIZKPs) are fundamental building blocks in cryptographic protocol design. They allow a prover to demonstrate knowledge of a witness w for a statement st without revealing any information about w itself. A common approach to constructing NIZKPs is to transform an interactive Σ-protocol into a non-interactive proof using the Fiat–Shamir heuristic. In this transformation, the prover first computes a commitment u using a random value r. Next, they derive the challenge by hashing the public parameters pp, statement st, and the commitment u. Finally, the prover computes the response using r and the witness w. Multiple $\Sigma-$protocols can be composed using logical connectives such as AND and OR.

In this section, we present the main NIZKPs employed in FiltrumVote. Each of them is obtained as a composition of simpler underlying Σ-protocols. Detailed descriptions and full constructions are provided in the full version of the paper.

NIZKP of Possession of Encrypted Public-Key. A proof of possession π_{poss} proves that a multiplicative ElGamal ciphertext (a, b) is an encryption of the voter's public key $\mathsf{pk}_i = g^{\mathsf{sk}_i}$, and the prover knows the corresponding secret key sk_i[1].

$$NIZKP[(r, sk_i) : a = g^r \wedge b = g^{sk_i} pk^r]$$

NIZKP of Consistency. The proof of consistency π_{cons} takes two ciphertext pairs $[\![pk_1]\!], [\![\sigma_1]\!]$ and $[\![pk_2]\!], [\![\sigma_2]\!]$ and proves that their underlying plaintexts are either both equal or both different. Thus, the proof is a composition of two proofs of plaintext equality with two proofs of plaintext inequality. We assume that the prover knows the secret key sk required to decrypt these ciphertexts.

$$NIZKP[(sk) : (pk_1 = pk_2 \wedge \sigma_1 = \sigma_2) \vee (pk_1 \neq pk_2 \wedge \sigma_1 \neq \sigma_2)]$$

NIZKP of Correct Selection. The proof of selection π_{sel} receives as input two ciphertexts $[\![\sigma_1]\!]$ and $[\![\sigma_2]\!]$, a ciphertext $[\![f_2]\!]$, a ciphertext $[\![v]\!]_T$ and a reencrypted ciphertext V'. It proves that V' is either (1) a reencryption of $[\![v]\!]_T$ when $\sigma_1 = \sigma_2$

[1] Note that $(\mathsf{sk}_i, \mathsf{pk}_i)$ refers to the voter's key pair, whereas $(\mathsf{sk}, \mathsf{pk})$ refers to the Electoral Authority's key pair.

and $f_2 = 1$; or (2) an encryption of $[\![1]\!]_T$ otherwise. This mechanism is used in our system to ensure that only the most recent valid vote from each user is selected. The proof is a composition of a re-encryption proof, a plaintext-equality proof, a plaintext-inequality proof, and three proofs of plaintext-value constraints.

$$NIZKP[(sk, r) : (V' = ReEnc([\![v]\!]_T, r) \wedge \sigma_1 = \sigma_2 \wedge f_2 = 1) \\ \vee (V' = [\![1]\!]_T \wedge (\sigma_1 \neq \sigma_2 \vee f_2 = 0))]$$

NIZKP of Greater-than. The proof of greater-than π_{gt} takes as input two ciphertext tuples $[\![\sigma_1]\!], [\![f_1]\!], [\![ctr_1]\!]$ and $[\![\sigma_2]\!], [\![f_2]\!], [\![ctr_2]\!]$ and proves that the second tuple is lexicographically greater than the first (i.e., $\sigma_2||f_2||ctr_2 > \sigma_1||f_1||ctr_1$). This proof is used in our system to sort encrypted votes. The proof is a composition of two range proofs, three plaintext-equality proofs, and two proofs of plaintext-value.

$$NIZKP[(sk) : (\sigma_2 > \sigma_1) \vee (\sigma_2 = \sigma_1 \wedge f_2 = 1 \wedge f_1 = 0) \\ \vee (\sigma_2 = \sigma_1 \wedge f_2 = f_1 \wedge ctr_2 > ctr_1)]$$

Range Proofs. Range proofs prove that, given a commitment $V = g^r h^v$, the value v is in $[0, 2^\ell - 1]$. One classical technique for range proofs consists of simply decomposing v in ℓ bits, committing each bit, and proving that each commitment indeed commits to either 0 or 1. This results in a proof size linear to the number of bits l. Some optimizations are described in [9].

In our application, we use range proofs to compare two ciphertexts $[\![x_1]\!]$ and $[\![x_2]\!]$ and prove that $x_2 > x_1$. To do so, we make a homomorphic subtraction $[\![x]\!] \leftarrow [\![x_2]\!] \cdot [\![x_1]\!]^{-1}$, create a new commitment V with the same plaintext as $[\![x]\!]$, and prove that it commits to a positive value $x \in [0, 2^\ell - 1]$. This assumes that $x_1, x_2 \ll q$, which must be checked by the verifier.

Verifiable Shuffle. A verifiable shuffle protocol takes a list of ciphertexts, re-encrypts them, and outputs a permuted list of ciphertexts. It is accompanied by a zero-knowledge proof of correctness that guarantees no ciphertext has been altered or replaced. FiltrumVote adopts the Wikström–Terelius protocol [24], which commits to a permutation matrix and proves in zero knowledge that this matrix was used in the shuffle. In addition, it is possible to shuffle not only a simple list of ciphertexts, but also a list of tuples of ciphertexts—one just needs to use the same permutation matrix across multiple verifiable-shuffle instances. This applies even if one position in a tuple uses a different encryption key from another position in the tuple.

4.2 FiltrumVote Protocol

In this section, we present a detailed description of FiltrumVote.

Setup Phase. The EA determines the security parameter λ, and the length of the credential to support the number of voters n_v : $\ell \geq \lceil \log(n_v) \rceil$. It also generates an encryption key pair (pk, sk) and publishes the pk. In addition, the n_T election trustees jointly run a distributed key generation, obtaining and publishing an encryption key pk_T while keeping their private shares sk_i (for $i = 1, ..., n_T$). Finally, the EA defines the list of valid candidates, reserving the identifier 1 to represent an invalid vote.

Registration Phase. Each voter vid_i samples a private key $sk_i \in_R \mathbb{G}_p$ and computes the public key $pk_i = g^{sk_i}$. The voter transmits pk_i over an authenticated channel to the EA. The EA then assigns the voter a unique, small credential $\sigma_i \in \{0,1\}^\ell$ and publishes the tuple (vid_i, pk_i, σ_i) in the voter roll R. Voters check that their public key pk_i is correctly registered in the voter roll.

Submission Phase. To cast a vote v_i, the voter does the following:

- encrypts their vote, using the trustee's public key: $[\![v]\!]_T = Enc_{pk_T}(v)$;
- encrypts their public key, using EA's public key: $[\![pk_i]\!] = Enc_{pk}(pk_i)$;
- encrypts their credential, using EA's public key: $[\![\sigma_i]\!]^\star = Enc^\star_{pk}(\sigma_i)$;
  - Note that the credential is encrypted using exponential ElGamal. This provides an additive homomorphism, necessary to compare two ciphertexts using range-proofs (Sect. 4.1).
- computes a proof π_v that v encrypts a valid candidate[2];
- computes a *NIZKP of encrypted public-key possession* π_{poss} for $[\![pk_i]\!]$;
- computes a proof π_σ that $[\![\sigma_i]\!]^\star$ encrypts a small $\sigma \in [0, 2^\ell - 1]$[3];
- sends the ballot $\beta_i = ([\![v_i]\!]_T, [\![pk_i]\!], [\![\sigma_i]\!]^\star, \pi_{kw}, \pi_{poss}, \pi_\sigma)$ to the EA through an authenticated channel.

The EA receives the ballot and checks if the proofs are valid and if there is no duplicate in the PBB. It also decrypts the credential and public key and checks if the pair (pk_i, σ_i) is present in the voter roll. If all checks succeed, the EA publishes the ballot on PBB. Voters can access PBB to check that their ballots were correctly registered.

Cleansing Phase. Once the voting phase is finished, the EA starts the cleansing phase. It can be divided into five steps.

[2] This is necessary to avoid forced abstention attacks. For example, if v is an unlikely, arbitrary string, the presence of v in the decrypted votes could indicate whether the voter has revoted. Range proofs could be used, with the optimizations described in Sect. 5.2.

[3] This is necessary since *NIZKP of greather-than* assumes $\sigma << p$. Again, range proofs can be used together with the optimizations described in Sect. 5.2.

Preparation. Let B be the list of all ballots in PBB, listed in chronological order, and let R be the voter roll. Then, the EA parses each element e_i of $B||R$ as a tuple $t_i = (\mathtt{v}_i, \mathtt{pk}_i, \boldsymbol{\sigma}_i, \mathtt{f}_i, \mathtt{ctr}_i)$, where

- If $e_i \in B$,

$$t_i \leftarrow ([\![v_i]\!]_T, [\![pk_i]\!], [\![\sigma_i]\!]^\star, Enc^\star_{pk}(0, r), Enc^\star_{pk}(ctr, r))$$

- Else if $e_i \in R$,

$$t_i \leftarrow (Enc_{pk_T}(1, r), Enc_{pk}(pk_i, r), Enc^\star_{pk}(\sigma_i, r), Enc^\star_{pk}(1, r), Enc^\star_{pk}(ctr, r))$$

Where *ctr* is an increasing counter and represents the chronological order of the votes, and $r = 0$ to facilitate verification from external auditors. Note that σ, *ctr* and f are encrypted using exponential ElGamal: σ and *ctr* for additive homomorphism, and f to simplify NIZKPs.

Mixing. The EA applies a verifiable shuffle in L_1. The resulting set L_2, and the proof of correct shuffle are published in PBB.

Sorting. The EA sorts the encrypted set L_2, producing L_3, as follows:

- It decrypts the credential, flag, and counter of each tuple in L_2: $(\sigma_i, f_i, ctr_i) = (Dec^\star(\boldsymbol{\sigma}_i), Dec^\star(\mathtt{f}_i), Dec^\star(\mathtt{ctr}_i))$. Since σ_i, f_i, and ctr_i are small numbers, this decryption is feasible. The EA keeps the resulting plaintexts private.
- It sorts the set according to the relation $t_i < t_j \iff \sigma_i||f_i||ctr_i < \sigma_j||f_j||ctr_j$, using any $O(n \log(n))$-complexity sorting algorithm, such as mergesort.
- It proves that the resulting set L_3 is correctly ordered by providing an *NIZKP of greater-than* π_{gt} for each pair of consecutive tuples.

The EA then publishes the sorted set L_3 together with the corresponding *greater-than* proofs.

Consistency Check. The EA proves that there are no impersonation-attack votes in L_3. To this end, it provides a *NIZKP of consistency* π_{cons} between each pair of consecutive votes. It then publishes the consistency proofs.

Selection. For each pair of tuple $t_i, t_{i+1} \in L_3$, the EA constructs a encrypted vote V'_i as follows:

- If t_i corresponds to a valid vote (i.e., $\sigma_i = \sigma_{i+1} \wedge f_{i+1} = 1$), $V'_i = ReEnc([\![v_i]\!]_T, r)$.
- If t_i corresponds to an invalid vote (i.e., $\sigma_i \neq \sigma_{i+1} \vee f_{i+1} = 0$), $V'_i = [\![1]\!]$.

The EA produces an *NIZKP of correct selection* for each V'_i. It then publishes the resulting set L_4 containing all V'_i, together with the corresponding selection proofs in PBB.

Tally Phase. The trustees take L_4 and apply a mixnet. Then, every vote V_i' is threshold decrypted. Votes whose decrypted plaintext is invalid (i.e., $v_i = 1$) are discarded. The remaining votes are tallied using standard techniques.

Verification Phase. Any participant, including the voter and external observer, can verify the correctness of the election by checking: (1) the voter roll correctness; (2) all *NIZKPs*; (3) the additional ciphertexts in the *preparation* step; and (4) the tally result.

5 Discussion

5.1 Security Analysis

In this section, we argue about the security properties of FiltrumVote. A formal security proof is left for future work.

Ballot Secrecy. Ballot secrecy relies on a threshold number of honest trustees. Votes are encrypted under the trustees' public key; thus, if the final mixing is sound, the shuffling is privacy-preserving, and the ElGamal encryption is CPA-secure, it is infeasible to identify the voter's selection from public information.

Verifiability. Election verifiability does not rely on trusting the EA or the trustees; rather, it is guaranteed by the correctness of the NIZKPs. Because the PBB is append-only, the mixing protocol is sound, and vote tuples remain public at every step, any attempt by the EA to modify or remove data is detectable.

In addition, the protocol ensures that only the final vote cast by each eligible voter is counted. This is achieved through the cleansing phase: a preparation step that assigns a "flag tuple" to eligible voters; an ordering step that groups and chronologically sorts voters' tuples so the flag tuple appears last; and a selection step that isolates and counts only the votes immediately preceding the flag tuple. The integrity of each of these steps is publicly verifiable via NIZKPs.

Finally, no one, including the EA, can cast a valid vote without knowing an eligible voter's secret key. This is enforced by multiple verification checks in the pair (pk, σ). First, in the submission phase, the NIZKP of possession ensures only the secret key owner can send a ballot with the corresponding public key pk. Second, the consistency step rejects mismatched pairings like (pk', σ), where σ is valid but pk' is incorrect for that voter (Sect. 3.2). Finally, the selection step nullifies any tuple with invalid credentials (σ') by only accepting votes followed by valid flag tuples.

Coercion-Resistance. Coercion resistance is ensured by an honest EA and a threshold number of honest trustees. This is achieved because all tuples on the PBB are encrypted, hiding the number of votes cast by any individual. Furthermore, the initial mixing ensures that the ordering process acts on what appears to be a random set, preventing any data leakage. Additionally, neither the NIZKPs

of consistency nor the NIZKPs of selection reveal voter-specific information. Finally, the final mixnet removes any link between the submitted vote tuples and the decrypted votes. Ultimately, the only publicly revealed information is the total number of cancelled votes, which does not offer any advantage to a coercer in a large enough election. Note that it does not even reveal if the voter has participated in an election, protecting against forced abstention attacks.

5.2 Performance Evaluation

To evaluate FiltrumVote's performance, we have implemented a proof-of-concept prototype[4] of the main NIZKP used in our solution: proof of possession π_{poss}, proof of greater-than π_{gt}, proof of consistency π_{cons}, and proof of selection π_{sel}. These are the only non-standard primitives in our solution. The prototype is implemented in Rust, and all operations are done over the elliptic curve P-256.

Experiments. All computations are done locally in a 12th Gen i7-12700H Intel Core CPU with 32GB RAM. We measured both proof generation and verification times for each NIZKP, averaging the results over 1000 repetitions. We use 24 bits to represent σ and ctr, which is more than enough to support an election with more than one million votes. Table 1 shows the results.

Since the range proofs in π_{gt} constitute the most computationally expensive component of the protocol, we introduce two optimizations. First, we reduce the bit-length of the credential σ_i and the counter ctr. Although our target remains one million votes, in practice, votes are tallied and published in batches (e.g., by geographic region). For example, one million votes can be partitioned into batches of 1024 voters, allowing σ_i to be represented with 10 bits. Furthermore, assuming a revoting rate of 0.5, a 12-bit counter ctr is sufficient. This alone reduces the proving time by almost half.

The second optimization we implement is fixed-base exponentiation precomputation. While all NIZKP proofs benefit from this technique, the impact is greater in π_{gt} due to the large number of exponentiations in the range proof. Specifically, we precompute a 16-bit multiplication table for both bases g and h. The table construction takes 350 ms, which can be performed only once before the election, and the resulting table requires 67 MB of storage.

Combining these two optimizations, we reduce the π_{gr} proof time from 26.64 ms to 2.89 ms, making it comparable to the simpler π_{cons} and π_{sel} proofs. Although more advanced techniques from [9] could be applied, they would introduce significant additional complexity while offering only marginal practical gains. In particular, even though such methods achieve asymptotically more efficient prover performance ($O(\log n)$), the small bit-lengths used in our range proofs limit the extent of these improvements in practice.

Analysis. We now estimate how long it takes to process 1 million votes in the main steps of the cleansing phase. If we consider a 0.5 rate of revoting and the

[4] https://github.com/no1-4now/FiltrumVote.

Table 1. Average time for proving and verifying main NIZKPs and their proof size. The optimization opt.1 refers to a smaller number of bits for σ and ctr, while the optimization opt.2 refers to fixed-base exponentiation techniques.

NIZKP proof	Prove (ms)	verify (ms)	proof size (bytes)
π_{poss}	0.33	0.53	96
π_{cons}	1.58	1.64	290
π_{sel}	2.14	2.36	321
π_{gt} (no-optimized)	26.64	26.68	6578
π_{gt} (opt.1)	15.68	13.77	3224
π_{gt} (opt. 2)	5.69	17.57	6578
π_{gt} (opt.1&2)	2.89	8.97	3224

additional flag tuples, we would need to process 2.5 million vote tuples. Considering the numbers in Table 1, this would require $\approx$ 1 hour in the consistency step, $\approx$ 1.5 hours in the selection step, and $\approx$ 2 hours in the ordering step to prove the NIZKP statements. Note that this estimate does not include other more standard tasks, such as encryption and decryption, and sorting tuples.

This results in a total of $\approx$ 4.5 hours to prove all NIZKPs to process 1 million votes. Note that this computation is highly parallelizable. For reference, DeVos voting can process only 80.000 votes in similar conditions—this considering a reduced, four-candidate election [22]. In addition, the NIZKPs for all 2.5 million tuples can be stored in only $\approx$ 9.15 GB. We therefore conclude that FiltrumVote is computationally feasible even for large-scale elections.

6 Conclusion and Future Works

We present FiltrumVote, a verifiable Internet voting system designed to resist coercion through deniable vote updating. The system is highly scalable, featuring a quasi-linear cleansing phase, and is capable of efficiently handling elections with up to one million votes. FiltrumVote relies primarily on zero-knowledge proofs using standard Σ-protocol techniques, enabling easier independent verification.

Future work includes developing formal security proofs to more rigorously assess FiltrumVote's security properties. Additionally, we plan to enhance the protocol to achieve quantum resistance, either by employing everlasting-privacy techniques [11] or by integrating post-quantum cryptographic primitives.

References

1. Achenbach, D., Kempka, C., Löwe, B., Müller-Quade, J.: Improved Coercion-Resistant electronic elections through deniable Re-Voting. In: USENIX JETS (2015)
2. Adida, B.: Helios: web-based open-audit voting. In: USENIX Security Symposium (2008)
3. Aranha, D.F., Battagliola, M., Roy, L.: Faster coercion-resistant e-voting by encrypted sorting. In: EVOTE-ID (2023)
4. Araújo, R., Barki, A., Brunet, S., Traoré, J.: Remote electronic voting can be efficient, verifiable and coercion-resistant. In: Financial Cryptography (2016)
5. Araújo, R., Ben Rajeb, N., Robbana, R., Traoré, J., Youssfi, S.: Towards practical and secure coercion-resistant electronic elections. In: CANS. (2010)
6. Cortier, V., Gaudry, P., Yang, Q.: Is the JCJ voting system really coercion-resistant? In: IEEE CSF (2024)
7. Cristiano, L., Longo, R., Spadafora, C.: Click and cast - assessing the usability of vote app. In: EVOTE-ID (2024)
8. Culnane, C., Schneider, S.: A peered bulletin board for robust use in verifiable voting systems. In: IEEE CSF (2014)
9. Devillez, H., Pereira, O., Peters, T.: How to verifiably encrypt many bits for an election? In: ESORICS (2022)
10. Farzaliyev, V., Willemson, J.: End-to-end verifiable Internet voting with partially private bulletin boards. In: EVOTE-ID (2025)
11. Haines, T., Mueller, J., Mosaheb, R., Pryvalov, I.: Sok: secure e-voting with everlasting privacy. In: Proceedings on Privacy Enhancing Technologies (PoPETs) (2023)
12. Haines, T., Müller, J., Querejeta-Azurmendi, I.N.: Scalable coercion-resistant e-voting under weaker trust assumptions. In: 38th ACM/SIGAPP SAC (2023)
13. Heiberg, S., Martens, T., Vinkel, P., Willemson, J.: Improving the verifiability of the Estonian Internet voting scheme. In: EVOTE-ID (2017)
14. Hirschi, L., Schmid, L., Basin, D.: Fixing the Achilles heel of E-voting: the bulletin board. In: IEEE CSF (2021)
15. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: ACM WPES (2005)
16. Kiayias, A., Kuldmaa, A., Lipmaa, H., Siim, J., Zacharias, T.: On the security properties of e-voting bulletin boards. In: SCN (2018)
17. Locher, P., Haenni, R.: Verifiable Internet elections with everlasting privacy and minimal trust. In: VOTE-ID (2015)
18. Locher, P., Haenni, R., Koenig, R.E.: Coercion-resistant Internet voting with everlasting privacy. In: Financial Cryptography (2016)
19. Lueks, W., Querejeta-Azurmendi, I., Troncoso, C.: VoteAgain: a scalable coercion-resistant voting system. In: USENIX Security (2020)
20. Marky, K., Zollinger, M.L., Roenne, P., Ryan, P.Y., Grube, T., Kunze, K.: Investigating usability and user experience of individually verifiable internet voting schemes. In: ACM-TOCHI (2021)
21. Morais, E., Koens, T., Van Wijk, C., Koren, A.: A survey on zero knowledge range proofs and applications. SN Appl. Sci. (2019)
22. Müller, J., Pejó, B., Pryvalov, I.: Devos: deniable yet verifiable vote updating. In: PoPETS (2024)

23. Nissen, C., Hilt, T., Budurushi, J., Volkamer, M., Kulyk, O.: Voting under pressure: perceptions of counter-strategies in Internet voting. In: EVOTE-ID (2025)
24. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 100–113. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12678-9_7
25. Treier, T.: Re-voting under surveillance: national eID transaction logs as a threat to coercion resistance in Estonian Internet voting. In: EVOTE-ID (2025)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



"In My Opinion, the Auditing Steps are Sufficient." Trust in Elections, Software, and RLAs

Christian Mack(✉), Tobias Hilt, and Melanie Volkamer

SECUSO, Karlsruhe Institute of Technology, Karlsruhe, Germany
{christian.mack,tobias.hilt,melanie.volkamer}@kit.edu

Abstract. The standard way to tally ballots in Germany is to manually count ballots at the polling station. However, some local elections use so complex ballots and voting rules that software is also used to capture the content of each ballot. The corresponding software itself nor the software's results in terms of number of votes per party and per candidate are subject to independent auditing. We wanted to understand how this situation influences voters' trust in such elections and what proposals would increase their trust. We conducted 16 semi-structured interviews with people who had participated in such complex elections. We make the following observations: trust in the election is overall very high, there is a lack of awareness regarding the use of software, and once made aware the main concern are the humans in the loop. Unprompted critical thinking about software which may manipulate the election outcome was observed only rarely. When talking about auditing approaches, our participants focused on low-effort approaches.

Keywords: interviews · trust · auditing · risk limited audit · election

1 Introduction

Federal and state elections in Germany are very simple elections in which one selects one candidate and one party. However, some local elections in Germany are characterized by a complex system for distributing votes. In these elections, voters usually have a double-digit number of votes, as well as the option to cast more than one vote for one or more candidates and to vote for candidates from different parties. While in German federal and state elections, ballots are counted manually at the polling stations, in these complex local elections each individual ballot is entered into a software. This step is conducted at one location per election district and takes several hours and sometimes even several days. In all three types of German parliamentary elections, software is used to sum up the results per election district.

Tallying software used for federal election has been criticized due to security issues, a lack of evaluation, and the use of closed-source software [8,18]. Software used in the complex local elections has been criticized for similar reasons [13,18].

© The Author(s) 2027
D. Duenas-Cid et al. (Eds.): E-Vote-ID 2026, LNCS 17083, pp. 125–140, 2027.
https://doi.org/10.1007/978-3-032-39512-2_8

The accuracy of the tallying software for federal elections can be audited because the results per polling stations are available in written form signed by the poll workers who have manually counted the ballots in the corresponding polling stations. However, there are no such records for complex local elections, as ballots are entered directly into the software without manual counting beforehand. Thus, the critique regarding the software is much more crucial in this case. Although the scientific community has proposed and discussed the use of risk-limiting audit (RLAs) approaches to significantly reduce risk [5], to our knowledge, there are no legal requirements to implement auditing processes, and there are no auditing processes implemented for such elections in Germany. Thus, there is no way to detect manipulations on the election result by the software.

Therefore, we wanted to understand how voters' perceive the current tallying procedure for complex local elections and which auditing processes would increase their trust in the election outcome. Specifically, we aim to answer the following research questions in the context of complex local elections:

Research Question 1: To what extent are voters aware of the current tallying procedures and to what extent do voters trust the current tallying procedure?

Research Question 2: How trustworthy do voters perceive various auditing processes to reduce the risk of undetected manipulations of the election result?

Research Question 3: How trustworthy do voters perceive various random number generators when these methods are used as part of the auditing process?

To answer these research questions, we conducted 16 semi-structured interviews with people who cast a vote in previous complex local elections. The interviews were audio-recorded, transcribed, and analyzed using an inductive coding approach. We find that (1) awareness of the use of software is not very high and that (2) overall trust in the accuracy of election results is high. (3) Furthermore, we find that most participants do not view additional auditing steps as necessary, even not once being informed about the use of software for capturing and tallying ballots. The majority of participants have a positive opinion about the presented RLA process, while they fear human error with other audit methods. For the generation of randomness, they favor software-based approaches over other physical approaches. We conclude that the trustworthiness of the election results is not likely to be questioned by voters. Thus, researchers should reach out to politics and lawmakers to require audit procedures.

2 Background and Related Work

Complex Local Elections in Germany. In Germany, the voting procedures for local elections vary from state to state. Some of these procedures are characterized by a particularly complex voting process, with ballots listing hundreds of candidates from different parties and, in some cases, allowing voters to cast a high number of votes, often in the double digits. These can be distributed among candidates of different parties. In some states, voters are also allowed to cast more than one vote for a single candidate [30], and /or to cross out candidates. Due to the complex nature of the ballots, election workers use software

for capturing each individual ballot. Election workers enter individual votes into the software in parallel on different machines at central locations. While doing so each ballot gets a unique identifier written on the paper ballot and stored with the electronic copy. The software generates an overall election result. A detailed description of the tallying process can be found in the supplementary material [20].

As for such complex local elections the amount of invalid ballots is higher than with any other parliamentary election in Germany, research has proposed voting systems to support voters filling out these complex ballots [6,7]. The focus of this work was on "cast as intended" verifiability given the fact that voting systems were introduced to local elections. There was also the idea of the digital pen for the complex elections in Hamburg [3,29]. The pen, which was equipped with a camera and a microprocessor, could transmit the voter's vote to a machine like a laptop, supporting the tallying process. However, the critique and the costs for such pens were too high for actual usage in elections.

Trust Related Research in Electronic Voting. The concept of trust is a broadly studied topic in relation to elections, especially for electronic voting [12]. In various papers, trust in electronic voting technology was identified as a key component for the willingness to adopt electronic voting [17,22,24,25]. Furthermore, research has shown that lack of trust can stop the introduction of electronic voting [11]. The relation between trust in electronic voting and various personality traits [27,31] as well as none-personality factors [21]. were studied, too. However, the focus is on electronic voting and quantitative studies while our focus is on tallying software and qualitative research. Other work explores electronic voting and the deployment of election software, assessing their consequences for trust in the electoral process and its legitimacy [16]. In this context, they look at electronic poll books and biometric identity verification in the U.S.

Risk-Limiting Auditing. Risk-limiting audits (RLA) is a much discussed approach [4], which can be used to audit the election outcome [23]. RLA samples paper ballots based on statistical methods and re-evaluating them. It has been used to audit elections in the U.S. (e.g., California [14]). The use of RLA to audit complex German local elections has been proposed in [5]. However, the authors did not discuss this approach with voters – which is what we did. Dalela et al. [9] studied U.S. voters trust in RLAs, showing high trust when hearing about the approach. Our research question 2 is similar while focusing on German voters of complex German elections and comparing RLAs with alternative approaches.

3 Methodology

Supplementary material is hosted on our servers [20]. It includes the interview guide, material we used during the interview, the codebook and further results.

3.1 Ethical Consideration, Recruitment & Pre-Study

The study received ethical clearance by the ethics board of our institution. Two pre-study interviews were conducted, in which unclear descriptions were discovered and the auditing descriptions were improved. Participant were recruited using a local study panel, specialized in conducting studies with citizens from the general public. Inclusion criteria were adulthood, having participated in local German election in the past and the ability to visit the study building. Prior to participation, participants received the study's data protection policy and the informed consent. After they signed the consent form they were informed, that the recording of the interview would now begin. At the end of the interview, the recording were stopped. Interviews lasted on average one hour and participants were reimbursed with €20. For the analyses, he interview transcripts were generated using a local instance of NoScribe. The generated transcripts were manually inspected by one researcher, who removed any potential identifiable information participants might have given. Note, at no point in time, participants' names were connected with recorded study data.

In total, 18 interviews for the main study were conducted. Saturation was reached after 16 interviews while two had to be excluded (one due to a mistake by the study conductor and one because the interview could not be finished as the participant had to leave. Two more interviews were conducted but no new topics were identified. The final data set comprises of 16 interviews.

3.2 Interview Procedure

First, participants were welcomed and asked to read the data protection policy as well as to read and sign the informed consent document. Next, participants were informed that the recording starts. The interview was divided into five phases (see Fig. 1). Each included a question about trust in the accuracy of the election results (5 point Likert scale from completely agree to strongly disagree). Note, it was clearly communicated that it is always about the complex local elections.

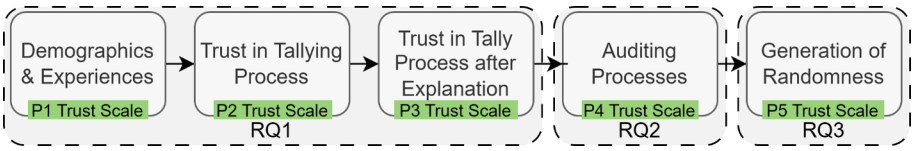


Fig. 1. Overview of the interview procedure

Phase 1 (Demographics and Experiences). Once the recording was started, gender and age were recorded, with age being collected only as categories. Afterwards, participants were asked general questions such as whether they have worked as a poll worker, and whether they have observed the tallying process. Afterwards, they were asked about their trust in the accuracy of the election results.

Phase 2 (Trust in Tallying Process). Next, participants were asked to familiarize themselves with the ballot and the instructions from the last local election in the city the interview took place in. Note, the instructions do not mention the tallying process at all. The goal was to remind them about the complexity of both the ballots and the voting rules which make manual tallying difficult and very time consuming. Afterwards, participants were asked to describe the tallying procedure in their own words and afterwards asked again the trust question. If a participant did not mention the use of software, they were subsequently asked whether they expected software to be deployed in the process. Furthermore, in case they did not mention any auditing step being in place, participants were asked whether they think such measures are in place and which ones.

Phase 3 (Trust in Tallying Process After Explanation). First, participants were provided with a detailed explanation of the tallying process [20]. We wanted to see whether having this information influences their overall trust in the election and therefore asked them to answer the trust question again. The following questions asked about which steps of the tallying process the participants trusted or distrusted particularly, what opportunities they saw for manipulation of the election results and which type of software provider they would trust in regard to the tallying software. Finally, we asked if participants consider additional auditing steps as necessary.

Phase 4 (Discussion of Various Auditing Approaches). In this phase, first, participants were made aware that the software could manipulate the election results. Furthermore, it is explained that such a manipulation would not be detected as there are no corresponding auditing procedures in place. Afterwards, participants were provided with a list of auditing approaches (name and description) – see Sect. 3.3. For each approach, participants were asked about their general opinion about the approach, the pros and cons they suspect as well as if they would trust them and whether they would welcome the deployment respectively. After each approach was discussed, they were instructed to rank them by sorting printed cards corresponding to the different approaches and read them aloud. Finally, they were asked to answer the trust question again assuming that their preferred approach is implemented.

Phase 5 (Discussion of Various Approaches to Generate Randomness). In the sixth and last phase of the interview we focused on the processes relying on randomness and how to to generate randomness. Similar to phase five, participants were presented with different approaches (see Sect. 3.3), where asked to answer questions about pros and cons as well as their trust towards those approaches and instructed to rank them by sorting printed cards corresponding to the different approaches and read them aloud. They were again asked to answer the trust scale under the assumption that their favored approach is implemented. Note, the descriptions of auditing processes as well as approaches to generate randomness were given in a random order, different from participant to participant.

Finally, participants were asked about any open questions or feedback they might have and they were given the €20 compensation.

3.3 Used Material for Auditing and Randomness Approaches

We presented the participant with five ways to audit the election results: 1) *2nd Software*: the use of a second tallying software from an independent provider on an independent machine to enter the paper ballots again, 2) *Predefined Percentage*: the re-evaluation of a predetermined percentage of ballots (e.g. 5–10% of all ballots) independent of the actual result of the election, comparing the paper ballot with the corresponding digital entry 3) *RLA*: the RLA approach that re-evaluates a percentage of ballots based on the election results (a higher percentage for closer elections), comparing the paper ballot with the corresponding digital entry 4) *Manual (Hand)*: a complete manual count parallel to the tallying software on paper, or 5) *Manual (Excel)*: a complete manual count parallel to the tallying software but with the support of software, e.g. Excel. The manual count was supplemented with Excel, a tool many participants already know. The presented approaches to generate randomness included dice, software, writing down numbers by different people, measuring radioactive decay and the use of books. Descriptions of the approaches as shown to the participants, can be viewed in the supplementary material [20].

3.4 Data Analysis

The interviews were analyzed with an inductive coding approach [28] using MaxQDA [15]. Two researchers analyzed four (25%) of the interviews independently and created individual codebooks for the three research questions. The generated codebooks were then assessed and unified during a discussion between the two researchers. Afterwards the adapted codebook was applied to these four interviews which reached high consensus (88%). One researcher then proceeded to code the remaining 12 interviews. This process resulted in three additional codes, that were added to the codebook and explained to the second researcher. The second researcher then reviewed the 12 coded interviews. Disagreements in coding were noted and discussed with the other researcher. Over the 12 interviews, the second researcher disagreed on 31 coded passages. These were resolved through discussion, resulting in perfect agreement.

4 Results

The study included 16 participants, predominantly women (n = 12) and older adults. The majority of participants were over the age of 50 (n = 12). Regarding specific age distributions: two participants were aged 31–40, two were 41–50, three were 51–60, eight were 61–70, and one was 71–80. One participant had experience as a poll worker, while no one had exercised their right to observe the tallying process. In the following subsections we present the results per research question. Note, quotes from the interviews listed as examples were translated.

4.1 Trust Scale (related to RQ 1, 2, 3)

Table 1 shows the results of the trust scales participants answered during the interview. We will discuss the remaining participants in more detail: Ten of the participants, who initially had complete trust, did not change their rating during the interview and consistently gave a score of five. The trust of the two remaining participants, who also initially had complete trust, dropped to 4 at one point but rose again to a full five points, provided that their preferred auditing process was used. For both participants, who started with a confidence level of four, trust in the election results rose to five with the introduction of a potential audit process they preferred. However, ID19 expressed a neutral view in phase 2:

Table 1. Responses to the 5-point Likert scale. Rows correspond to the specific interview phase in which the trust assessment was completed.

Part. ID:	4	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Phase1	5	5	5	4	5	5	5	5	3	5	5	5	5	2	4	5
Phase2	5	5	5	4	4	5	5	5	2	5	5	5	5	2	3	4
Phase3	5	5	5	4	5	5	5	5	1	5	5	5	5	3	4	4
Phase4	5	5	5	5	5	5	5	5	4	5	5	5	5	4	5	4
Phase5	5	5	5	5	5	5	5	5	5	5	5	5	5	2	5	5

> "I'll stay neutral, because I might have heard that things have gone missing before." (ID 19, about trust in election results and lost ballots)

Two participants (ID 13 and ID18) initially expressed no positive attitude towards trusting the election result.

While participant ID 13 originally has neutral stance due to general skepticism without a clear cause, e.g.

> "It just gives me a weird feeling [...] on the whole, Germans are generally dissatisfied with the elections or with our politics, yet the same thing keeps getting elected." (ID 18)

Participant ID 18 calls for a complete overhaul of the electoral process:

> "I mean, this whole process with paper and counting and people sitting there— it's bound to be error-prone anyway. [...] But it is the process itself that need changing. [...] The only way I would truly trust that the election results are 100% accurate is if they were conducted on a blockchain or via a smart contract. Because you simply can't cheat on a blockchain." (ID 18)

If participant 13's preferred auditing process were implemented, their level of trust would reach the maximum. Under the condition that a secondary software system was employed for auditing, Participant 18 assigned a trust score of 4 on a 5-point scale. The low rating for the random-sampling method stems from ID

18's dislike of auditing only examining a subset of ballots; the context in which trust in randomness was asked.

4.2 Awareness of and (Dis)trust in Tallying Process (RQ1)

The subsection is structured according to themes, we identified.

Lack of Awareness. Only two out of 16 participants knew about the use of software during the tallying process. Eight of the remaining 14 participants considered it plausible, when asked directly.

Distrust in Humans with Concrete Scenarios of Errors and Fraud. When talking about potential issues with the election result, the majority of participants mentioned the human component (10/16), specifically human error while entering votes into the software.

> "I thought that reading it aloud and entering it into the software left a lot of room for errors." (ID 4)

Participant also mentioned the transport of the ballot case from the polling place to the central tallying place as a potential risk.

> "Probably the most critical part is transporting the sealed suitcases containing the cards, because something might get lost" (ID 12)

Very Abstract Distrust in Tallying Software After being informed about the usage of tallying software, five participants referred to it as potential cause of errors.

> "I can't verify what the software does. Of course, there could be a bug in there somewhere, or something like that. I don't know." (ID 16)

Potential Manipulations During Tallying: When asked about steps at which manipulation could occur, four mentioned the tallying software as a potential risk in terms of attempts to manipulate the election results. However, human factors, such as poll workers misbehaving, are still the leading risk factors, identified by 13 participants.

> "The five election workers, there are a lot of people, but maybe they know each other, maybe the five of them are colluding [...] But that's relatively unlikely, I'd say off the top of my head. The other steps actually all already have some form of control." (ID 19)

Most participants, 14 out of 16, considered it possible to manipulate election results using the tallying software, but deemed it highly unlikely.

> "Well, it's certainly possible, Russian hackers are everywhere, after all. But I'd still say the likelihood is pretty low. [...] It's not exactly a high-stakes election. If this were a federal election, it might be a different story. But at the local level, I don't think anyone would go to the trouble of doing that." (ID 9)

Certification versus Auditing. When asked about the need for some kind of certificate, most participants, 13 out of 16 viewed it as a necessity or at least very important. Generally, the government was cited as the preferred authority for certification with multiple mentions of the German Federal Office for Information Security (in German: "Bundesamt für Sicherheit in der Informationstechnik" BSI).

> "Private companies, not so much. But there's always the BSI, which could perhaps look into it, if it's being used in a relatively critical area" (ID 17)

In total eleven did not see a need for additional auditing, one participant had a neutral view and four did welcome additional auditing steps.

> "Auditing steps are not the problem. The auditing steps are good. In my opinion, the auditing steps are sufficient." (ID 18)

> "Right now, it's probably still enough, but a lot of things are changing, and maybe it won't be enough much longer." (ID 12)

Further Requirements to Tallying Software. When asked about the software developer, many participants preferred companies based in Germany or the EU (6/16 German, 5/16 EU - non-exclusive), while there was no clear preference regarding whether participants preferred software developed by the government or by private companies.

> "It would be preferable to have a local or regional solution, at least something German or European." (ID 9)

> "I would try to keep this under my own control as much as possible, that is, under state control [...]. In the end, perhaps even at the European level." (ID 15)

4.3 Auditing Processes (RQ2)

Audit Process Ranking. After discussing each of the auditing processes individually, we asked the participants to rank the level of trust they would place in them. The RLA process ranked first in eight instances and second in four instances. A second tallying software was ranked first five times and second three times. The re-evaluation of a predetermined percentage of ballots was the preferred choice four times and five times the 2nd most popular while both manual processes were put lowest by most participants. The ranking for each participant can be found in the supplementary material [20].

Concerns Regarding Effort and Costs. During the discussion of various auditing processes, the most frequently cited concern was the additional effort required. The fixed-percentage recount method was the only process exempt from this criticism. Financial costs were also raised as an objection, particularly regarding the Software-based approach and, to a lesser extent, RLA.

So that means doing the same work twice when counting votes using two different software programs. I also find the process very labor-intensive. (ID 4, about a second software)

"You can do that, but I think it all costs a fortune and ends up being a bit of a penny-pinching or nitpicking affair." (ID 7, about RLA)

The additional effort was regularly cited in connection with the human errors that would result during the manual processes: e.g.

"I would rather trust the software than spend more time on it, because that would involve a manual count, which could also be prone to errors. " (ID 10, about manual count)

"I don't even want to know how much of it is just thrown in there after the thousandth record without even checking it. Because you say, 'I have to finish this at some point.' " (ID 17, about manual count with supporting software, e.g. Excel)

4.4 Random Number Generation (RQ3)

Ranking of Approaches. Similar to the auditing processes, we asked the participants to rank the approaches to generate randomness once each was addressed individually. The use of software to generate randomness was the preferred choice of nine participants and second-most-preferred choice of four. The radioactive decay as means to generate randomness was considered most trustworthy by five participants and second-most-preferred choice of four. Two participants placed most trust in dice. Dice, Books and the writing down of numbers by different people was considered equal by multiple participants and the radioactive decay was positioned in last place by four participants. The complete ranking for all participants can be found in the supplementary material [20].

Concerns Regarding Effort and Costs. As with the auditing processes, the effort involved was often the primary consideration for the randomness generation approaches as well. Especially for the radioactive decay participants mentioned effort frequently as well as the need for additional equipment:

"That sounds even more complicated. I wouldn't be too keen on that. The effort involved would also be a drawback" (ID 13)

"I imagine that would be kind of cumbersome. I don't know if that's proportional to anything " (ID 9)

With the exception of software-based randomness, the effort was one of two consistent arguments against the presented approaches. The software approach was mostly considered less time consuming:

"To be honest, I think that's better than the dice, because it's probably just faster." (ID 19)

Perceived as Inappropriate. Some proposals were perceived as inappropriate for a serious setting such as an election. Most frequently, participants related approaches to gaming:

> "It seems kind of weird, because it's like something you'd do at a game night with friends. Using it for such formal occasions just feels a bit odd." (ID 9, about writing down numbers)

> "We can do that in kindergarten, that is certainly feasible. We would never use it in a federal election" (ID 11, about writing down numbers)

> "Oh, those are nice little games, sure. But that one? No way." (ID 18, about using books)

Perceived as Too Easy to be Manipulated by Humans. Participants also commented on trust in the form of possible manipulation and how approaches are not random enough. Mentions of possible manipulation of the approaches included:

> "The book is opened at random. I've turned to that page a hundred times already. I've hardly ever turned to the others. Which page will it open to?" (ID 15, about using books)

> "But rolling dice at random is also stupid. Why? There might be a weight inside the 6." (ID 15, about using dice)

> "I mean, people can just agree on which number to write on the slips, right?" (ID 18, about writing down numbers)

> "People always pick the same numbers—maybe even subconsciously. " (ID 13, about writing down numbers)

There are some more quotes showing that some participants prefer to trust the software rather than the election workers:

> "There's really nothing a person can intervene in. That's why I think it's pretty safe" (ID 10)

> "It's definitely better than leaving it up to people." (ID 17)

Issues with Using Software. Participants raised concerns about manipulation in the context of the software-based approach, such as:

> "The conspiracy theory that it's no coincidence. That someone says they don't trust the software." (ID 6)

> "If you want to rule out the possibility that the software could be tampered with, something I hadn't considered yet, it would be bad if it's tampered with." (ID 12)

5 Discussion

5.1 Overall Observations Regarding Critical Thinking

Our participants emphasize human error and intentional manipulation by election workers as the main concerns – both in the current tallying process and when considering additional auditing steps. Software is generally perceived as a reliable component. Participants' critical scrutiny of approaches are strongly correlated with how well they are understood. The better an approach is understood, the more critically it is evaluated and, in general, the less trustworthy it is considered to be. In the tallying stage, this is reflected in concrete examples of human mistakes, such as shifting a line, intentionally invalidating a ballot, or similar manipulations. In contrast, criticism tends to be vague because a comprehensive understanding of the software is likely absent. This inverse relationship between understandability and trust corroborates the argument by Siegrist et al. [26] that trust and complexity influence each other.

5.2 Trust in Tallying Process

Trust may remain high even once made aware of the software in place and once having mentioned errors and manipulations by election workers because any mentioned error or manipulation is considered not very likely in particular given the fact that the cause of this study are local elections.

It might be that trust could be more increased when introducing certification of the software (by the BSI or German institutions in general) than by introducing auditing approaches. As auditing is required to detect manipulated tallying software as well as errors/manipulations by election workers, people would need to be made aware of the limitations of certifications (such as the inability to ensure that exactly the evaluated peace of software is in place and that the certification is based on assumptions which might be more or less realistic) to change their opinion about certification versus auditing. This is inline with research in the context of electronic voting where certification is perceived as strongly required while a lack of awareness for its limitations was observed.

5.3 Criteria for Auditing Approaches

We noticed that our participants care about the number of ballots being audited. Participants preferred RLA over re-evaluating a fixed amount of 5–10% of ballots. Participants also saw effort as a concern for RLA, while the fixed percentage approach was considered to be less effort. This trend suggests that participants expect a higher percentage for RLA. Which is inline with related work: Dalela et al. [9] showed that their participants overestimate the amount of ballots that will be audited through RLA and that their trust in the election outcome decreased after learning about the true amount. This aspect needs to be further researched to make it more likely that voters accept RLAs even once known the actual number.

Time and in particular effort was mentioned by various participants when discussing the different auditing approaches. This may have been influenced by the fact that the cause of this study were complex local elections, meaning: tallying anyway takes days, meaning that it is even more difficult to find election workers while at the same time the risk of manipulations might not be perceived very high [2,10]. Furthermore, participants worried that additional effort increases the likely hood of human errors.

Lastly, participants were in favor of auditing approaches which they perceive as less prone to human error. They preferred software based auditing approaches supporting the actual task of the election workers. Notably, participants favored abstract software-based approaches over the Excel-based auditing approach. This might be explained with the overall observation discussed in Sect. 5.1

5.4 Approaches to Generate Randomness

Randomness is required by various auditing approaches. Similar to auditing, effort of different approaches was frequently discussed. As such, in particular, radioactive-decay methods were not considered adequate by many given the effort to apply it. In particular in this part it was difficult to convince participants to focus on trust rather than effort when it comes to their ranking of the different approaches. This indicates some relation between what can be expected to be done and what needs to be done to make them trusting in the result.

Participants' perceptions of an approach's appropriateness appeared to directly influence their assessment of its trustworthiness. Approaches reminiscent of games or play, namely those involving dice, books and writing down of numbers, resulted in lower trust levels in the context of election auditing. This suggests that auditing processes must project an image of formality and seriousness to be deemed credible.

Many participants were in favor of software for the same reasons as when auditing approaches were discussed. The same explanation might hold as discussed in the general observations in Sect. 5.1. As approaches to securely apply the software were not mentioned at all, we believe it is worth to raise awareness for it, too. For instance, the process could involve several independent parties, each supplying a software-generated random segment. The final random value is the combination of these segments, so the outcome stays unpredictable as long as at least one participant is honest.

5.5 Limitations

Several limitations regarding our sample composition must be acknowledged: First, the age distribution was skewed toward older participants. The perspectives of younger voters, particularly those under 30, are likely underrepresented. Given their potentially higher exposure to digital technologies [19], their trust dynamics regarding tallying software might differ significantly from the patterns observed in our study. However, at least for the electorate in Germany the examined age group represents both the largest demographic and the one with the

highest voter turnout [1]. Second, we decided to only interview voters who have participated in such complex elections and thus, did not interview people who may have decided to not vote in such elections due to their lack of trust in the election outcome. Third, qualitative analysis is inherently subjective and prone to analyst bias. To maximize objectivity, we had two researchers independently develop the codebook and systematically resolve all coding disagreements. Nevertheless, as with any qualitative study, a degree of personal interpretation remains an inherent characteristic of the findings. Finally, We did not mention any numbers of ballots to be audited when it comes to the application of RLA. It would be valuable to investigate how voters' trust in the RLA process changes when getting to know the exact numbers; as Dalela et al. [9] observed a decrease of trust as numbers were lower than people expected them to be.

6 Conclusion

This interview study aimed to understand the trust of voters in the tallying process of local elections in Germany, including the use of tallying software. Trust in the accuracy of election results was generally very high. The use of tallying software did not affect trust negatively for our participants. Vulnerabilities and opportunities for manipulation were identified mainly in connection with election workers rather than software. Participants did not see the need for additional auditing steps, but deemed certification of the software crucial. While we agree that certification (by the BSI) contributes to the level of trust voters have, we argue that auditing processes are still required as it cannot be reliably ensured that the certified software is the one in place. Overall, we have to acknowledge that the trustworthiness of the election results is not likely to be questioned by voters. Thus, we conclude that researchers should reach out to politics and lawmakers to require audit procedures.

In addition, we looked at various auditing processes. Participants had a mostly favorable opinion about RLA, while approaches involving additional manual counts were dismissed due to high probability of human error. Furthermore, participants favored the use of software for the generation of randomness. Options such as the use of dice were viewed as less trustworthy and deemed inappropriate in the context of an official election. Throughout the interviews, it seemed there is greater trust in processes that participants seem to view in more abstract terms. A tallying software that is not described in further detail is preferred over Excel, a software that most participants know and have personal experience with. Therefore, we conclude that politics and lawmakers should require RLAs while using independent software and devices to generate randomness. A variety of elections, whether they use software to tally votes or involve complex ballots susceptible to errors, can benefit from including RLA. The applicability extends beyond any single election type or test case.

Acknowledgments. This work was funded by the Topic Engineering Secure Systems, subtopic 46.23.01 Methods for Engineering Secure Systems, of the Helmholtz Association (HGF) and supported by KASTEL Security Research Labs, Karlsruhe.

References

1. Demografieportal - Fakten - Altersspezifische Wahlbeteiligung—demografie-portal.de. https://www.demografie-portal.de/DE/Fakten/wahlbeteiligung.html. Accessed May 2026
2. Wahlen: Warum dauern Stimmauszählungen nach Kommunalwahlen so lange?—zeit.de. https://www.zeit.de/news/2026-03/15/warum-dauern-stimmauszaehlungen-nach-kommunalwahlen-so-lange. Accessed May 2026
3. Arzt-Mergemeier, J., Beiss, W., Steffens, T.: The digital voting pen at the Hamburg elections 2008: electronic voting closest to conventional voting. In: EVOTEID. Springer (2007)
4. Bernhard, M.: Risk-limiting audits: a practical systematization of knowledge. In: Proceedings of 6th International Joint Conference on Electronic Voting, pp. 162–177 (2021)
5. Blom, M., et al.: Assertion-based approaches to auditing complex elections, with application to party-list proportional elections. In: EVOTEID. Springer (2021)
6. Budurushi, J., Renaud, K., Volkamer, M., Woide, M.: An investigation into the usability of electronic voting systems for complex elections. Ann. Telecommun. (2016)
7. Budurushi, J., Volkamer, M., Renaud, K., Woide, M.: Implementation and evaluation of the easyvote tallying component and ballot. In: EVOTE. IEEE (2014)
8. Analyse einer Wahlsoftware, Chaos Computer Club. https://www.ccc.de/system/uploads/230/original/PC-Wahl_Bericht_CCC.pdf. Accessed May 2026
9. Dalela, A., Kulyk, O., Schürmann, C.: Voter perceptions of trust in risk-limiting audits (extended) (2021)
10. deutschlandfunk.de: Kommunalwahlen in Hessen - Auszählung der Stimmzettel kann mehrere Tage dauern—deutschlandfunk.de. https://www.deutschlandfunk.de/auszaehlung-der-stimmzettel-kann-mehrere-tage-dauern-100.html. Accessed May 2026
11. Duenas-Cid, D.: Trust and distrust in electoral technologies: what can we learn from the failure of electronic voting in the Netherlands (2006/07). In: DG-O (2024)
12. Erb, Y., Duenas-Cid, D., Volkamer, M.: Identifying factors studied for voter trust in e-voting-review of literature. In: EVOTEID (2023)
13. Hessen: Wahlsoftware auf dem Prüfstand, Frankfurter Rundschau. https://www.fr.de/rhein-main/hessen-wahlsoftware-auf-dem-pruefstand-90461664.html. Accessed May 2026
14. Hall, J.L., et al.: Implementing risk-limiting post-election audits in California. In: EVT/WOTE (2009)
15. MAXQDA, VERBI GmbH. https://www.maxqda.com/. Accessed May 2026
16. Murray, G.R., Albert, C.D.: In cyber we trust? Understanding election legitimacy in the age of electronic election systems. J. Inf. Technol. Polit. (2025)
17. Nemeslaki, A., Aranyossy, M., Sasvári, P.: Could on-line voting boost desire to vote?-Technology acceptance perceptions of young Hungarian citizens. Gov. Inf. Q. **33** (2016)
18. Alle Wege führen nach Aachen, Leonhard Pitz. https://netzpolitik.org/2025/wahlsoftware-alle-wege-fuehren-nach-aachen/. Accessed May 2026
19. Olson, K.E., O'Brien, M.A., Rogers, W.A., Charness, N.: Diffusion of technology: frequency of use for younger and older adults. Ageing International (2011)
20. Supplementary Material, https://files.secuso.org/secuso/2026-07-Interview_Study_Local_Election_Auditing/. Accessed July 2026

21. Palas Nogueira, J., de Sá-Soares, F.: Trust in e-voting systems: a case study. In: Rahman, H., Mesquita, A., Ramos, I., Pernici, B. (eds.) MCIS 2012. LNBIP, vol. 129, pp. 51–66. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33244-9_4
22. Romanov, B., Cid, D.D., Solvak, M.: I Know, Therefore, I Trust? Quantitatively modelling how knowledge shapes the reliance on trust and confidence in the case of internet voting usage in Estonia. Interact. Comput. (2025)
23. Sarwate, A.D., Checkoway, S., Shacham, H.: Risk-limiting audits and the margin of victory in nonplurality elections. Stat. Polit. Policy **4**(1), 29–64 (2013)
24. Serdült, U., Kryssanov, V.: Internet voting user rates and trust in Switzerland (2018)
25. Serdült, U., Milic, T.: Disentangling digital divide and trust: internet voting affinity in Switzerland. In: Parycek, P., et al. (eds.) ePart 2017. LNCS, vol. 10429, pp. 37–52. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64322-9_4
26. Siegrist, M.: Trust and risk perception: a critical review of the literature. Risk Anal. **41**(3), 480–490 (2021)
27. Sindermann, C., Rozgonjuk, D., Solvak, M., Realo, A., Vassil, K.: Internet voting: the role of personality traits and trust across three parliamentary elections in Estonia. Curr. Psychol. (2023)
28. Thomas, D.R.: A general inductive approach for analyzing qualitative evaluation data. Am. J. Eval. **27**(2), 237–246 (2006)
29. Volkamer, M., Vogt, R.: New generation of voting machines in Germany–the Hamburg way to verify correctness. In: PRE-PROCEEDINGS, Hamburg, Germany (2006)
30. Wahlrecht.de, Matthias Cantow, Dr. Martin Fehndrich, Wilko Zicht. https://www.wahlrecht.de/kommunal/. Accessed May 2026
31. Wermelinger, C.C., Mariano, A.M., Da Silva, J.M.: Trust and security in critical digital infrastructures: a conditional analysis of acceptance and intention to change in electronic voting systems. IEEE Access (2026)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



École: Privacy in Diversity

Florian Moser[1(✉)] and Léo Louistisserand[2]

[1] famoser GmbH, Switzerland, Switzerland
florian.moser@famoser.ch
[2] CNRS, Inria, LORIA, Université de Lorraine, France, France
leo.louistisserand@loria.fr

Abstract. As part of the political process, citizens may participate in (signature) collections to influence policy changes. Depending on the legislative basis, such collections may even result in legally binding acts. Unlike e-voting systems, the security of such collecting systems remains unexplored. In this work, we first derive a realistic setting for e-collecting based on e-voting. Then, we propose the secure protocol École in this setting, achieving both privacy and verifiability under realistic trust assumptions. Notably, participation privacy is guaranteed without the need for an anonymous channel, by assuming that at any given point in time, many diverse collections are active in parallel.

Keywords: E-Collecting · Privacy · Verifiability

1 Introduction

In democracies, elections form the backbone of how voters decide how their government moves forward. However, besides electing their representatives, citizens often have other avenues to directly or indirectly participate in policy changes. This includes collection systems where citizens are able to express their political views [5,29], and depending on the legislative basis, directly or indirectly influence policy setting. For successful European Citizens Initiatives (ECI), the largest collection system we are aware of, which impacts 450 million citizens [14], the European Commission is bound to issue an official statement [13]. In Switzerland, as a part of their system of direct democracy, successful collections even lead to a direct vote of all citizens on the topic [32]. Both systems aim to ensure only eligible citizens participate, while keeping secret (besides verification purposes) who participated [13,32].

Overall, collecting systems have therefore similar targets to voting systems. The system needs to make sure the voter's choices are properly reflected in the system, and their privacy is guaranteed. For electronic voting systems, such intuitions have led to the state-of-the-art systems that are verifiable privacy-preserving voting systems. Verifiability guarantees the election result is auditable by third parties. Privacy ensures, despite the auditability, that the voting choice

© The Author(s) 2027
D. Duenas-Cid et al. (Eds.): E-Vote-ID 2026, LNCS 17083, pp. 141–157, 2027.
https://doi.org/10.1007/978-3-032-39512-2_9

of the individual voter remains protected. These mechanisms are mature and employed in binding political elections [19,27,33].

However, collection systems also have some important differences in functionality. First, a collection may start and end at any time, in particular, multiple times per year, and many collections run in parallel.[1] A dedicated setup process per collection, such as sending postal mail to the eligible voters [33], is therefore unrealistic. Second, collections are long-running (many months [13,32]). In particular, the eligibility set may change throughout the collection period (coming of age, relocation, death), while the electorate is generally assumed fixed by voting systems. Finally, signatures cannot be revoked, and always correspond to "yes" votes; there is no possibility to vote "no". Revoting is therefore not applicable, and keeping the voting choice private is insufficient. Instead, a collecting system needs to provide the stronger *participation privacy*. E-voting protocols in this setting generally need to assume an anonymous channel (e.g., [4,17,24]), which is considered a strong and somewhat unrealistic assumption [18].

In academia, the security of e-collecting has not received any attention so far. Further, actual deployed e-collection systems seem to be lagging behind what is considered state-of-the-art for voting systems. The e-collecting system for the ECI needs to comply only with a high-level directive concerning the general security of information systems in the European Union [12,13], but in particular verifiability guarantees are not a requirement. The e-collecting system by the Swiss canton of St. Gallen, to enter production in 2026 [22], seems to assume a fully trusted server[2], and also does not provide any notion of verifiability [1].

It is important to document that the security of (e-)collection systems is a concern in practice. The e-collecting system for the ECI was developed as one of the recommended measures to better safeguard the personal data of supporters [29, VI.3.1.], and this same study also documents incidents of data loss and abuse happening before. In Switzerland, a report by a newspaper highlighted that many signatures (currently all given on paper) are invalid [3,21]. The suspected reason is attempted fraud by commercial signature collection firms [21], who are notably reimbursed per collected signature [26]. As a consequence, on a federal level, the Federal Chancellery has been tasked to implement e-collecting [2,15].

First, we derive a realistic setting for secure e-collecting, based on the strong setting for e-voting as used in Switzerland [16]. Second, in this demanding setting, we propose a protocol achieving privacy and verifiability. Crucially, despite achieving participation privacy, we are able to avoid assuming an anonymous channel, which would limit its applicability in practice [18]. Instead, we are able to hide participation in some specific collection as a potential participation in

[1] For the ECI, 3 – 18 initiatives have been in their active collection phase in parallel under the current law [13]. In Switzerland, 5 – 31 over the last 20 years.

[2] The system provider neither confirmed nor denied an attack found when the server is assumed dishonest, but instead asserted that the system is compliant with the legal basis. The cantonal legal basis omits a concrete threat model and to-be-reached properties [31, Art. 27]. Notably, in national legislation, e-voting systems must not rely on a fully trusted server, and are required to be verifiable [16].

any currently active collection. The proposal is the first verifiable and privacy-preserving e-collecting system we are aware of.

2 Related Work

We are not aware of any academic work exploring the security of e-collecting. As e-collecting systems face similar challenges and aim for similar properties as e-voting, we explore here primarily works written in the context of e-voting.

As argued in the introduction, participation in a collection already exposes the preference. Hence, one might aim to participate anonymously. E-voting protocols providing privacy by anonymizing the participation however assume an anonymous channel to cast the vote (e.g., [4,17,24,25]). This is considered a strong assumption, with limited applicability in practice [18], and we therefore avoid it in this work. The same limitation applies to proposals for anonymous survey systems [20], which additionally also do not aim for verifiability.

Another approach to achieve privacy in e-voting is (deniable) vote updating. This approach accepts that the adversary may be able to observe some voting procedure of the voter, but allows the voter to mitigate by updating their vote later on (e.g., [19,25,28]). However, the vote updates must still be hidden from the adversary and therefore need to rely on the anonymous channel assumption. Further, collection systems do not allow to revoke the participation, therefore once participation has been granted, revoting is no longer meaningful.

From a purely cryptographic point of view, our ballots are closest to DeVoS [28]. Concretely, DeVoS' ballots are also either a re-encryption of the previous ballot, or a fresh (proven valid) ballot from the voter. However, on the protocol level, we differ substantially. First, we do not rely on a trusted authority to hide the participation of voters. Instead, we hide participation in a specific collection by running the protocol over all collections in parallel (i.e., have multiple DeVoS-style ballots in parallel). Further, we require functionality out of scope for DeVoS, including mutations to eligibility and collections that start and end anytime.

3 Defining the E-Collecting Setting

No setting for e-collecting has been clearly defined yet. We intend to do this here, while profiting from the extensive body of knowledge already available for e-voting. Therefore, we motivate our definitions in Sect. 3.1 based on established notions in e-voting. The same holds for the parties and acceptable trust assumptions in Sect. 3.2, which we primarily derive from the Swiss setting. We choose the Swiss setting as it is strong and well-defined by law [16], and because there is a binding and therefore impactful collection system in place.

Definition 1 (E-Collecting scheme). *Let $\mathcal{V}$ be a set of voters, and $\mathcal{C}$ be a set of collections. An e-collecting scheme is formed of processes* `voter`(V)*, for $V \in \mathcal{V}$, that trigger* $\mathsf{Cast}(V, C)$ *when the voter V intends to sign the collection C, and* $\mathsf{Verified}(V, C)$ *when V successfully verified that their participation to C has been recorded, and a process* `tally`*, that triggers* $\mathsf{Counted}(V, C)$ *when the participation of V to C is considered in the final result.*

3.1 Properties

End-to-End Verifiability of Participation. In internet voting, end-to-end verifiability captures the notion that the result of the election is correct. Concretely, the result of Counted votes is composed of all votes of honest voters who Verified successfully, a subset of the votes of honest voters that only Voted but did not verify, and some additional votes bound by the number of adversarial voters $\#\{\mathcal{V}_{\mathcal{A}}\}$. The default strong flavour of the definition further prescribes that the content of the ballot of the honest voters must be correct [8,9], while the weak variant (e.g., used in [7]) only requires this for voters who verify.

We can reuse the same approach for e-collecting. However, as participation always expresses support (i.e., the vote is always 1), we do not need to consider the adversary possibly changing votes. Therefore, we can use a weaker notion that only considers whether participation has been Cast. We adapt the injective formulation of the definition [6]. Note how the definition implicitly encodes eligibility, as we understand honest voters are indeed eligible.

Definition 2 (E2E-Verifiability of Participation). *An e-collecting scheme provides* E2E-Verifiability of Participation *if it verifies two subproperties:*

- Individual verifiability*: any verified participation to a collection C is part of the collection result, i.e., it holds that* $\mathsf{Verified}(V, C) \Rightarrow \mathsf{Counted}(V, C)$.
- Universal verifiability*: the voting result is composed solely out of at most one participation per honest voter and per attacker-controlled voter, i.e., it holds injectively that* $\mathsf{Counted}(V, C) \Rightarrow \mathsf{Cast}(V, C) \vee V \in \mathcal{V}_{\mathcal{A}}$.

Participation Privacy. In electronic voting, the prevalent secrecy notion is vote privacy [30]. It captures the intuition that the voting system should not expose more information about how individuals voted than what is known from the result. However, participation overall in the system is not protected. This is expressed as an indistinguishability property that states that the attacker should not be able to distinguish between two settings that lead to the same election result. Concretely, in each setting, the same voters vote overall for the same multiset of votes. The adversary is however able to choose for each setting which voter votes for which voting option.

Directly reusing this definition for e-collecting is unfortunately not possible. The reason is that participation directly expresses support (i.e., the vote is always 1). Therefore, the adversary would not be able to define multiple settings, i.e., in both settings, the voters would vote for the same voting option. However, we can make the definition useful again by considering multiple collection events C_1, C_2 as the different votes. As before, participation overall in some collection remains unprotected with this definition.

Definition 3 (Participation Privacy). *An e-collecting schemes provides* Participation Privacy *if, for all* $A, B \subseteq \mathcal{V} \times \mathcal{C}$ *such that* $\forall C \in \mathcal{C}, \#\{V \in \mathcal{V}, (V, C) \in A\} = \#\{V \in \mathcal{V}, (V, C) \in B\}$*, then every run with* $\mathsf{Verified}(V, C)$ *for all* $(V, C) \in A$ *is indistinguishable from the same run with* $\mathsf{Verified}(V, C)$ *for all* $(V, C) \in B$.

3.2 Parties

At the core of the Swiss setting are the groups of control components. Each group is intuitively one distributed server, and only one of the components per group is assumed trustworthy (while it must remain unspecified which one). We take over this concept, and use two groups of control components: one group establishes an append-only Bulletin Board, and the other group shares key material and is called the Talliers.

During the setup of the election, an additional fully trusted setup component defines the electorate and receives key material from the control components. Then, the setup component prints out authentication and verification material and sends them over postal mail to the voters. To adapt this to the e-collecting system, we first note that e-collecting runs continuously, and not at pre-defined election events. Concretely, since a new collection may start anytime, it is tedious to each time send fresh verification material via postal mail to the voters. Further, the electorate continuously changes, rather than being fixed after a cut-off date. Therefore, we transfer the tasks of the Setup component to other components. First, to (continuously) define the electorate, we define the trusted Electoral Roll. Second, we abstract authentication of the voter from the protocol, and instead assume there is a non-repudiable Authenticated Channel from the voter to the Bulletin Board (e.g., using an electronic signature via an E-ID). Authenticating citizens online is an independent problem from e-collecting, and the optimal approach, including recovery or refresh of authentication material used for a prolonged time, heavily depends on existing infrastructure.

As for elections, we continue to assume the voters use their own personal devices and trust them for privacy. Concretely, we call the primary device used the Participation Device. Further, we assume an independent Audit Device that the voter can use to audit their participation. While both the Audit Device and the Participation Device are trusted for privacy, only one of them is assumed honest for verifiability (Table 1).

Table 1. Trust assumptions for Verifiability and Privacy in the e-collecting setting we defined, derived out of the Swiss e-voting setting [16]. ● for trusted, ◐/ ◑ for 1-out-of-2 trusted, ○ for 1-out-of-n trusted. Stronger trust assumptions are worse.

	Verifiability	**Privacy**	*Difference* to e-voting setting
Bulletin Board	○	○	*same* as 1-out-of-n trusted Control Components
Talliers	○	○	
Electoral Roll	●		*weaker* than fully-trusted Setup and Postal Channel
Authenticated Channel	●		
Participation Device	◐	●	*stronger* than privacy-trusted Voting Device
Audit Device	◑	●	

4 Presenting École

In this section, we present École, the first proposal for privacy-preserving and verifiable e-collecting. The core idea of the protocol is to hide participation in a concrete collection by making this participation indistinguishable from participations in other collections (see Sect. 4.1). École relies only on common cryptographic primitives like public key encryption and zero-knowledge proofs, and well-understood subprotocols for distributed key generation and decryption that we recall in Sect. 4.2. Then, we present the protocol, from start to end of a collection, while also considering key rotation aspects to support indefinite running of the protocol (see Sect. 4.3). Finally, we introduce an extension of École to the hybrid setting, important for its practicality in real-world use cases (see Sect. 4.4).

4.1 Core Idea

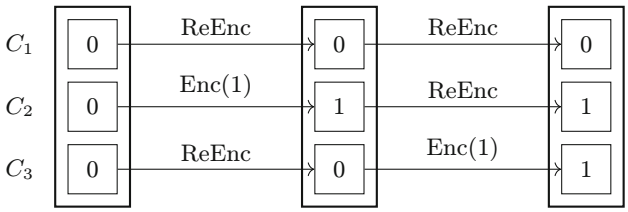


Fig. 1. Each column represents a full ballot. For each active collection C_1, C_2 and C_3, there is a dedicated ciphertext. To participate, the Participation Device replaces the corresponding ciphertext by an encryption of 1. All other ciphertexts are re-encrypted.

The ballot consists of a ciphertext for each active collection. To sign one or more new collections, the Participation Device encrypts 1 for these specific collections, and re-encrypts the ciphertext for all other collections. A zero-knoweldge proof ensures the Participation Device formed the ballot correctly. The new ballot then contains a fresh ciphertext for each active collection, and it is therefore indistinguishable to the adversary, which collection(s) were signed (Fig. 1).

To enable both tallying and individual verifiability, the Voter' choices are encrypted towards both to the Talliers public key and the Voter' public key. Another zero-knowledge proof ensures both encryptions behave the same (i.e., are either both a re-encryption of the previous ciphertext, or both an encryption of 1). The Voter can then check on an independent Audit Device that the ciphertext encrypted under their public key contains the expected value. To tally, the ciphertext of all Voter corresponding to the same collection are aggregated, and then the Talliers decrypt this aggregated ciphertext.

The approach has a separate ciphertext per Voter and per collection, and it is thus straightforward to support eligibility mutations and arbitrary collection

periods. Concretely, the Electoral Roll continuously adapts the whitelists of Voter per collection, and only Voter on the whitelist are allowed to replace their ballots. Further, the Talliers can open or close collections at any time, and the Voter can only replace ciphertext of open collections.

4.2 Preliminaries

Our protocol relies on the following common cryptographic primitives:

- *PKE*: A public-key encryption system with the symbols $(sk, pk) \leftarrow \mathsf{KeyGen}()$, $c \leftarrow \mathsf{Enc}(pk, p, r)$, $p \leftarrow \mathsf{Dec}(sk, c)$ for ciphertext c, plaintext p and randomness r. Further, we define additional non-standard functions. The function $c' \leftarrow \mathsf{ReRand}(pk, c, r')$ reencrypts c under randomness r', such that $\mathsf{Dec}(sk, c) = \mathsf{Dec}(sk, c')$. We write $c'' \leftarrow c + c'$ to homomorphically add ciphertext c and c' such that it holds that $\mathsf{Dec}(sk, c'') = \mathsf{Dec}(sk, c) + \mathsf{Dec}(sk, c')$. In particular, exponential El Gamal can be used to implement this.
- *ZKP*: We use zero-knowledge proofs to prove operations on ciphertext without revealing the plaintext or the encryption randomness. We denote such proofs with $\pi \leftarrow \mathsf{ZKP}[s]$, and it holds that if and only if statement s is true, then $\mathsf{VerifyZKP}(\pi) = 1$. We prove statements of the form $c = \mathsf{Enc}(pk, n)$ and $c' = \mathsf{ReRand}(pk, c)$, where the first one asserts that c is an encryption of n, and the second that c' is a re-encryption of c. Additionally, we compose statements with logical AND ($\wedge$) and OR ($\vee$). Implementing this is straightforward, in particular for DeVoS a similar ZKP was implemented and also shown to be very efficient [28].

Besides the cryptographic primitives, we assume distributed verifiable key generation and decryption as subprotocols. $\mathsf{DKeyGen}()$ outputs an encryption public key pk. Its counterpart is $\mathsf{DDec}(c)$, which will, given a ciphertext $c \leftarrow \mathsf{Enc}(pk, m, r)$, return the corresponding plaintext p. These two subprotocols are executed by multiple parties, and assume at least one party is honest. Such protocols are implemented for example in Belenios [10].

4.3 Protocol

We divide the protocol description into multiple parts. The *Setup* protocols include the Talliers introducing a collection, the Voter registering, and the Electoral Roll defining a whitelist of eligible Voter per collection. The *Participation* section describes how an eligible Voter participates in some of the many collections, and how the Voter can verify this participation. As the system is designed to run continuously, we further must account for users losing or rotating their keys, as well as mutations in eligibility (e.g., coming of age, naturalization, death). Finally, the *Tally* describes how the participations in the collections are counted. Note that these parts of the protocol are subsequent phases per collection, but different collections may be in different phases at the same time.

We assume an established PKI in between the parties.[3] All messages exchanged in the protocol are signed under this PKI, and sent to the public append-only Bulletin Board. The Bulletin Board publishes only valid messages (i.e., the signature under the PKI passes, the zero-knowledge proof of each ballot is valid, etc.). Further, we describe the Bulletin Board to execute all other computations that do not require any secret or random input. As it therefore operates deterministically only on public data, it is feasible to distribute the Bulletin Board to multiple independent parties, of which only one needs to be assumed honest. Honest parties then always act given the most recent state of the Bulletin Board (e.g., concerning pk_v).

Setup. The Voter registers their Audit Device by generating a key pair $(sk_v, pk_v) \leftarrow \mathsf{KeyGen}()$, and publishing pk_v on the Bulletin Board. Similarly, the Talliers introduce a collection by generating a distributed encryption key with $pk_t \leftarrow \mathsf{DKeyGen}()$, and publishing pk_t on the Bulletin Board. The Electoral Roll defines for each collection a whitelist of voters $\mathcal{V} \leftarrow \{v, \dots\}$ who are eligible.

For all voters v in the whitelist $\mathcal{V}$, the Bulletin Board generates an initial ballot for each collection. Concretely, the Bulletin Board encrypts 0 into $c_t \leftarrow \mathsf{Enc}(pk_t, 0, 0)$ and into $c_v \leftarrow \mathsf{Enc}(pk_v, 0, 0)$, using randomness 0 (to render the operation deterministic and therefore trivially verifiable). These two values form the initial entry of the ballot $b \leftarrow [(c_v, c_t)]$ corresponding to this collection, and are published on the Bulletin Board (Fig. 2).

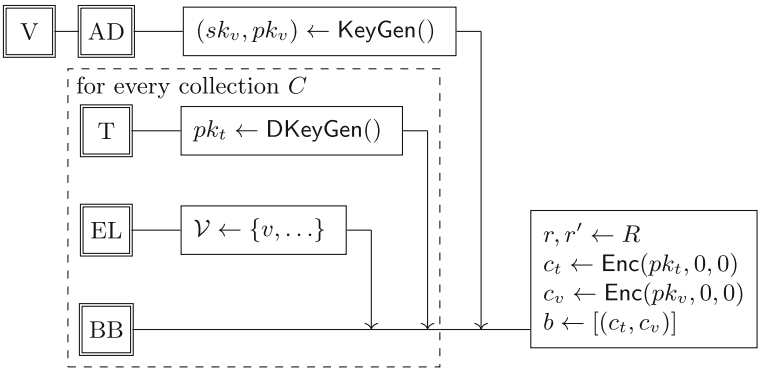


Fig. 2. The *Setup*. Audit Device and Talliers register their corresponding public keys, and the Electoral Roll defines the whitelist of eligible voters. Then, the initial ballot is created as an encryption of 0. The public key of the Audit Device is reused over collections.

[3] To authenticate the messages of the voter, for example an electronic id may be used.

Participate. To participate, the Voter uses their Participation Device to encrypt 1 into $c'_t \leftarrow \mathsf{Enc}(pk_t, 1, r)$ for some randomness r, and into $c'_v \leftarrow \mathsf{Enc}(pk_v, 1, r')$ for some randomness r'. Then, the device creates an OR zero-knowledge proof attesting *either* that 1 has been encrypted into c'_t and c'_v, *or* that c'_t and c'_v are reencryptions of c_v and c_t from the previous ballot entry $b = [_, (c_v, c_t, _)]$. Of course, only the first statement is true. Concretely, it forms the proof $\pi' \leftarrow \mathsf{ZKP}[(c'_t = \mathsf{Enc}(pk_t, 1) \wedge c'_v = \mathsf{Enc}(pk_v, 1)) \vee (c'_t = \mathsf{ReRand}(pk_t, c_t) \wedge c'_v = \mathsf{ReRand}(pk_v, c_v))]$. These three values form the newst ballot entry (c'_t, c'_v, π').

To hide the participation, the Participation Device now reencrypts the other ballots. Specifically, for all other collections' last ballot entry $(c_t, c_v, _)$, the ciphertext are re-randomized into $c'_t \leftarrow \mathsf{ReRand}(pk_t, c_t, r)$ for some randomness r, and into $c'_v \leftarrow \mathsf{ReRand}(pk_v, c_v, r')$ for some randomness r'. Then, the same OR zero-knowledge proof π' is created as for participation, but now with the second statement being true.

Finally, the Participation Device sends all new triplets over the authenticated channel to the Bulletin Board. The payload is considered valid if all ballots have received a single new entry, and if the zero-knowledge proofs are valid and therefore form a consistent chain until the initial entry. Note that when submitting new ballots, the Voter may also choose to sign no collection, or multiple collections at the same time. However, the zero-knowledge proofs prevent to remove the participation once it has been granted (Fig. 3).

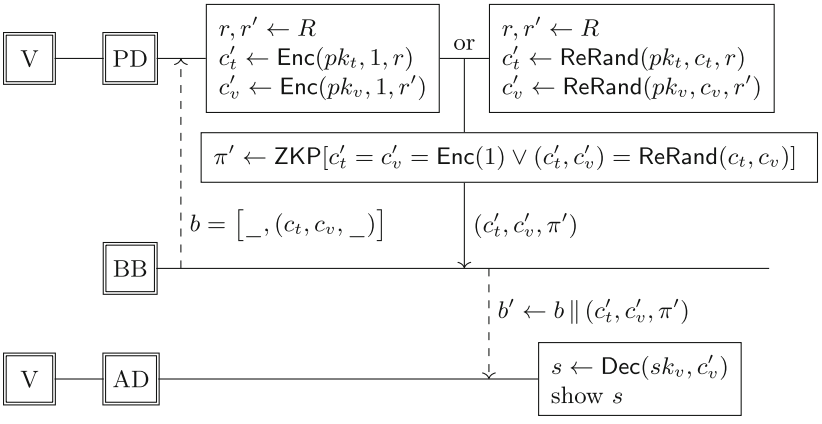


Fig. 3. The *Participation*. For every collection C, the Participation Device either encrypts 1 or re-encrypts the previous ciphertext. It creates a zero-knowledge proof that the ballot is well-formed and then sends the new entry to the Bulletin Board. To perform the individual verification, the Audit Device decrypts the ciphertext of the last ballot entry and shows the plaintext s to the Voter.

To individually verify that their ballots have been formed correctly, the Voter can use the registered Audit Device. Concretely, the device downloads the ballots from the Bulletin Board, and then decrypts each last entry's c_v into $\{0, 1\}$.

The Voter then verifies whether the Audit Device shows for all collections their intended choice. As any other party with access to the Bulletin Board, the Audit Device may additionally verify the ballot entries, i.e., all zero-knowledge proofs are valid and form a consistent chain until the respective initial entry.

Key Rotation and Eligibility Mutations. As each collection runs for an extended period of time, and as the collection system runs continuously, we need to account for key rotation. We first note that a fresh pk_t per collection is derived, and as the collection period is limited, no rotation is necessary here.

To rotate the voter key pk_v, the voter (re-)registers their Audit Device. Concretely, the voter derives a new key pair $(sk'_v, pk'_v) \leftarrow \mathsf{KeyGen}()$, and sends pk'_v over the authenticated channel to the Bulletin Board. From then on, all operations are done on pk'_v instead of pk_v. This increases the complexity of the individual verification, but it remains possible, which we will detail here.

For participations created after key rotation, individual verification will work as before: The c_v of the last entry of the ballot is decrypted, and results in 1. For the other ballots, created under the old public key pk_v and re-encrypted under pk'_v, the decryption is impossible. However, due to π, these ballots must be re-encrypted ballots, and are therefore unmodified since key-rotation. Hence, to audit these ballots, the Audit Device would decrypt the last c_v encrypted under the old public key pk_v. In case the Voter has lost their previous secret key sk_v, they will be unable to decrypt the old entries, and therefore can no longer verify participations from before pk'_v was registered. However, they can still be assured that the ballot has not been modified since the key rotation (Fig. 4).

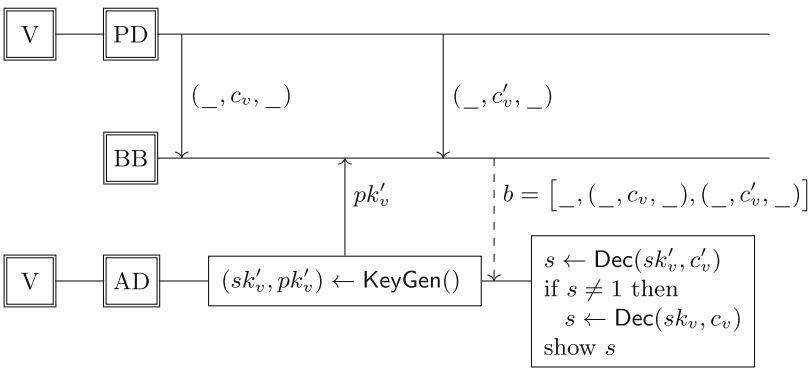


Fig. 4. The impact of rotating the voter encryption key on individual verification. Participations created after rotating the key can be decrypted. Otherwise, the last ciphertext from before the key rotation can be decrypted using the previous secret key.

In case the Voter is de-authorized by an Electoral Roll (e.g., moving, death), they will be removed from the corresponding whitelists. From then on, the Bulletin Board will not accept new entries for these ballots. Conversely, the Voter

will be added to new whitelists when a new Electoral Roll authorizes them, or they gain eligibility to additional collections. The Bulletin Board will again accept entries for the existing ballots, and create fresh initial ballots for the collections the voter has not been eligible before.

Tally. To tally a collection, the c_t values of the last entries are collected out of all voter's ballots. They are homomorphically added into c_t^T. Then, the Talliers execute the distributed decryption to get the sum $s \leftarrow \mathsf{DDec}(c_t^T)$ (Fig. 5).

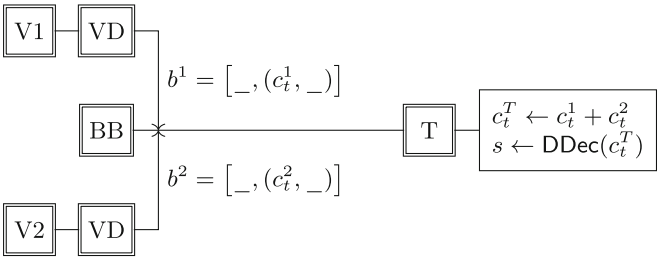


Fig. 5. The *Tally*. For every collection C, Talliers aggregate the ciphertexts from the respective last entry of the ballots. Then, the aggregated ciphertext is decrypted.

As the ballots are attributed to the voters, groups of ballots may be formed and decrypted separately, e.g., for statistical purposes in age groups or per electoral roll. Further, instead of just tallying at the end of the collection period, the tally may be executed regularly to learn whether a collection has already collected sufficient signatures. However, building these kinds of partial sums introduces privacy leaks, which, in the worst case, may make voters' participation again identifiable. Techniques from differential privacy may help to reduce these kinds of side-channels, but we consider this to be out of scope for this work.

4.4 Hybrid Setting

As described in Sect. 3, hybrid systems are common in the e-collecting setting. For example, the Swiss setting will continue to support physical signatures even if an e-collecting channel becomes available. As the use of online-only systems seems very limited, the protocol is designed to work well in the hybrid setting.

When the Voter participates over the hybrid channel HC, e.g., via signed declarations of support on paper, HCwill execute the same participation protocol as the voter would. Concretely, for the collection in which the voter participated, HCcreates encryptions of 1, and re-encrypts all the other ballots. Together with the corresponding zero-knowledge proofs, these updated ballots are sent to the Bulletin Board. The Bulletin Board accepts them as the newest ballots of the corresponding voter. The Voter can then execute their individual verification check to see what participations the HChas created in their name.

If participations over HCare auditable by third parties, these audits can continue to be executed by the Talliers. Concretely, for all voters for whom there is evidence that they participated over HC, their last entry on the Bulletin Board can be checked to indeed encrypt 1. Further, for all voters for whom no evidence of participation over HCexists, the tallier may aggregate all c_t of the ballot entries submitted by HCinto c_t^{α}, and all c_t of the respective ballot entries immediately before into c_t^{β}. Then, the Talliers decrypt $s \leftarrow \mathsf{DDec}(c_t^{\alpha} - c_t^{\beta})$, and they verify that $s = 0$. Note that neither audit gives the tally any additional information besides what is already known from the evidence of HC. In particular, if the voter did not participate over HC, the Talliers do not learn whether the voter participated online (Fig. 6).

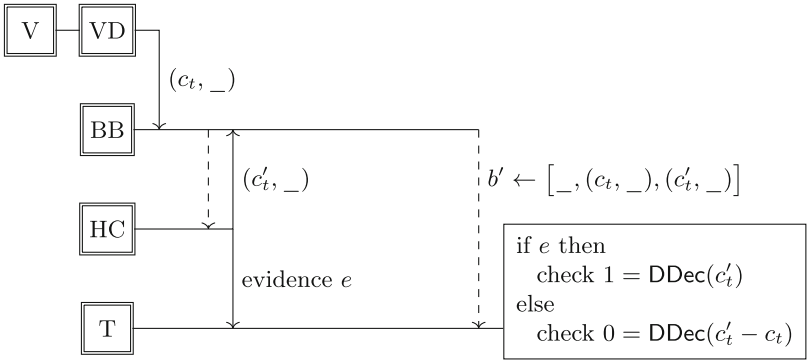


Fig. 6. Hybrid participation over another channel HC. When the HCcreates a participation for the Voter, they send corresponding evidence to the Talliers. If the evidence indicates the Voter has participated over HC, the Talliers verify that the last (tallied) ciphertext indeed encrypts 1. Otherwise, the Talliers verify that HCdid not stuff the participation. To ease presentation, this Figure simplifies the audit procedure (we omit the aggregation of the ciphertext from multiple voters).

5 Security Claims

In this section, we discuss the security guarantees that our protocol offers regarding verifiability and privacy, and provide an intuition of how our claims follow. Overall, the proposal fulfils the setting as introduced in Sect. 3.

We consider a Dolev-Yao adversary [11] able to reorder, compose or drop messages, but that cannot break cryptography (e.g., without access to the decryption key, cannot decrypt messages). In particular, the adversary has (read-only) access to the Bulletin Board and may compromise some of our parties. For all security claims, we assume the Bulletin Board and the Talliers overall to be honest. Distribution of these parties (e.g., as groups of control components) is practical, hence this is not a particularly strong assumption.

5.1 Verifiability

Individual Verifiability. When the voter Verified their participation successfully, their participation is guaranteed to be on the bulletin board, unless the adversary compromised both the Participation DevicePD and the Audit DeviceAD. Participations on the bulletin board cannot be revoked due to the zero-knowledge proof, and therefore all will eventually be Counted.

Universal Verifiability. Every participation that is Counted is attributed to a concrete voter. For the honest voters, this means they actually Cast the participation or their PD has stuffed votes. In the hybrid setting, we need to trust the hybrid channel HC for the same purpose. As each participation is attributed to exactly one voter, including for attacker-controlled voters $\mathcal{V}_{\mathcal{A}}$, injectivity holds. However, overall, we need to trust the Electoral RollER to not stuff voters.

The trust assumptions almost completely disappear when audits are performed. A verifying voter using an honest AD would discover ballot stuffing by the PD or over the HC. Further, if participations over HC are auditable by the Talliers, then they would discover any dishonest behavior by HC (both stuffing and dropping of participations). Lastly, the ER publishes the whitelists of eligible voters $\mathcal{V}$ on the bulletin board, and $\mathcal{V}$ can therefore be audited, too (Fig. 7).

	Trust assumptions	*without any audits*
Individual verifiability	PD $\vee$ AD	—
Universal verifiability	PD $\vee$ AD	PD $\wedge$ HC $\wedge$ ER

Fig. 7. Verifiability claims for École. We list the agents that have to be honest for each property to hold (PD: participation device, AD: audit device, HC: hybrid channel, ER: electoral roll). For universal verifiability, an honest audit device AD would discover ballot stuffing by HC or PD. Further, the Talliers would discover a misbehaving HC, and the ER's actions are public, and can therefore also be audited.

5.2 Privacy

The PD (and in the hybrid setting, the HC) records the participation of the voter and therefore fundamentally needs to be trusted for privacy. Similarly, the AD will also learn of the participation, in particular as it manages sk_v. The participation is however otherwise hidden within the ballot, as to the adversary a re-encryption is not distinguishable from an encryption of 1. The zero-knowledge proof attached to the ballot does by construction not leak which operation was executed. To ensure indeed all ballots reach the tally[4], we must assume that verifiability holds[5], and that all honest voters indeed verify. Then, the homomorphic aggregation before decryption makes it impossible to learn which voters participated in which collection and which did not (Fig. 8).

[4] Otherwise, there is a trivial attack where the adversary drops all ballots except one.
[5] Observe that this is possible under the same trust assumptions, i.e., PD $\wedge$ AD.

	Trust assumptions
Participation privacy	PD ∧ AD
Participation privacy HC	HC ∧ AD ∧ all talliers

Fig. 8. Privacy claims for École. We list the agents that have to be honest for each property to hold (PD: participation device, AD: audit device, HC: hybrid channel). We denote *all talliers* here explicitly, as for the audit each individual tallier will learn the evidence e of HC, and therefore could break the privacy of the impacted voters.

Note that when the honest Talliers audit the HC, then all Talliers also learn which voters participated over HC. However, they learn nothing in addition, in particular not which voters participated over the main protocol.

6 Conclusion and Privacy Limitations

The presented approach protects participation in some specific collection by hiding it in a potential participation in all open collections. This follows the privacy notion in internet voting, which implicitly assumes that the voting result does not reveal how individual voters cast their vote. Already for internet voting systems, this assumption is however not always fulfilled (e.g., when the short ballot assumption does not hold [23]).

For collections, privacy leaks may, however, be more pronounced. As collections run over an extended period of time, each collection may have its periods of popularity, where a participation overall is likely a participation for this specific collection. If additionally the tally is executed often, these patterns of popularity are clearly observable, and may conversely also single out participations for unpopular collections. Further, non-participation overall leaks that no collection has been signed at all, which seems to be a semantically more important information than abstention in an election setting. It is important to note that our protocol makes overall participations public, even to a passive attacker who reads only the bulletin board long after the voter interacted with the system.

Hence, we may need to adapt a fundamentally stronger privacy notion, that even participation overall must not leak. Observe that the current proposal implicitly supports this, as voters can cast a ballot with just re-encryptions (i.e., a non-participation). However, the usability of this is likely poor, as voters would need to be assumed to regularly take a (non-)action, e.g., once within each tally period. To improve usability, we could transfer this re-encryption to another authority, but this would likely need much stronger trust assumptions[6] (e.g., as those needed for DeVoS [28]).

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

[6] In particular, at least a secret or anonymous channel to hide the real participation.

References

1. Abraxas: VOTING E-Collecting docs. https://github.com/abraxas-labs/voting-ecollecting-docs
2. Année Politique Suisse: Dossier: Missbräuchliche Unterschriftensammlung. https://anneepolitique.swiss/dossiers/1607
3. Année Politique Suisse: Missbräuchliche Unterschriftensammlungen. https://anneepolitique.swiss/prozesse/68174
4. Arapinis, M., Cortier, V., Kremer, S., Ryan, M.: Practical everlasting privacy. In: Basin, D., Mitchell, J.C. (eds.) POST 2013. LNCS, vol. 7796, pp. 21–40. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36830-1_2
5. Bundesrat: Elektronische Unterschriftensammlung für eidgenössische Volksbegehren (E-Collecting): Bericht des Bundesrates in Erfüllung des Postulates 21.3607 Staatspolitische Kommission NR vom 27. Juni 2021. Bericht e-parl 21.11.2024 09:19, Bundesrat der Schweizerischen Eidgenossenschaft (Nov 2024), bern, 20. November 2024
6. Cheval, V., Cortier, V., Debant, A.: Election verifiability with proverif. In: 2023 IEEE 36th Computer Security Foundations Symposium (CSF), pp. 43–58. IEEE (2023)
7. Cortier, V., Debant, A., Gaudry, P., Louistisserand, L.: Vote&check: secure postal voting with reduced trust assumptions. In: Proceedings on Privacy Enhancing Technologies (2025)
8. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Election verifiability for Helios under weaker trust assumptions. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 327–344. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11212-1_19
9. Cortier, V., Galindo, D., Küsters, R., Mueller, J., Truderung, T.: Sok: verifiability notions for e-voting protocols. In: 2016 IEEE Symposium on Security and Privacy (SP), pp. 779–798. IEEE (2016)
10. 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
11. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (2003)
12. European Commission: Commission Decision (EU, Euratom) 2017/46 of 10 January 2017 on the security of communication and information systems in the European Commission. https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32017D0046
13. European Union: Regulation (EU) 2019/788 of the European Parliament and of the Council of 17 April 2019 on the European citizens' initiative. https://eur-lex.europa.eu/eli/reg/2019/788/oj/eng
14. Eurostat: Population structure and ageing. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Population_structure_and_ageing
15. Federal Chancellery: E-Collecting. https://www.bk.admin.ch/bk/de/home/politische-rechte/e-collecting.html
16. Federal Chancellery: Federal Chancellery Ordinance on Electronic Voting (2013)
17. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Seberry, J., Zheng, Y. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57220-1_66

18. Haines, T., Mueller, J., Mosaheb, R., Pryvalov, I.: Sok: secure e-voting with everlasting privacy. In: Privacy Enhancing Technologies Symposium (PETS) (2023)
19. Heiberg, S., Willemson, J.: Verifiable internet voting in Estonia. In: 2014 6th international conference on electronic voting: verifying the vote (evote). pp. 1–8. IEEE (2014)
20. Hohenberger, S., Myers, S., Pass, R.: Anonize: a large-scale anonymous survey system. In: 2014 IEEE Symposium on Security and Privacy, pp. 375–389. IEEE (2014)
21. Häfliger, M., Knellwolf, T.: Tausende Daten für Initiativen gefälscht: Unterschriften-Bschiss erschüttert die Schweiz. https://www.tagesanzeiger.ch/384143367276
22. Kanton St. Gallen: Kanton startet Pilotversuch mit E-Collecting. https://www.sg.ch/news/sgch_allgemein/2025/12/kanton-startet-pilotversuch-mit-e-collecting.html
23. Küsters, R., Truderung, T., Vogt, A.: A game-based definition of coercion resistance and its applications. J. Comput. Secur. **20**(6), 709–764 (2012)
24. Locher, P., Haenni, R.: Receipt-free remote electronic elections with everlasting privacy. Ann. Telecommun. **71**(7), 323–336 (2016)
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 (2020)
26. Marc Bühlmann and Hans-Peter Schaub: Staatspolitische Auswirkungen von E-Collecting: Studie im Auftrag der Bundeskanzlei. Technical report, Année Politique Suisse, Institut für Politikwissenschaft, Universität Bern, Bern (2023). https://www.newsd.admin.ch/newsd/message/attachments/90666.pdf, study commissioned by the Swiss Federal Chancellery
27. Moser, F., et al.: A study of mechanisms for end-to-end verifiable online voting (stuve). Technical report, Federal Office for Information Security, Germany (2024). https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Cryptography/End-to-End-Verifiable_Online-Voting.pdf
28. Müller, J., Pejó, B., Pryvalov, I.: Devos: deniable yet verifiable vote updating. In: Proceedings on Privacy Enhancing Technologies (2024)
29. Optimity Advisors: Study on data requirements for the European Citizens' Initiative. https://citizens-initiative.europa.eu/document/download/ccfa34c7-2c17-4f81-ba49-58324bd8def3_en
30. Smyth, B., Bernhard, D.: Ballot secrecy and ballot independence coincide. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 463–480. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40203-6_26
31. Staatskanzlei St. Gallen: sGS 125.1 - Gesetz über Referendum und Initiative (RIG). https://www.gesetzessammlung.sg.ch/app/de/texts_of_law/125.1/versions/3837
32. Swiss Confederation: Federal Constitution of the Swiss Confederation. https://www.fedlex.admin.ch/eli/cc/1999/404/en
33. Swiss Post: Swiss Post Voting System: System Specification. Version 1.5.2. Technical report, Swiss Post (2025). https://gitlab.com/swisspost-evoting/e-voting/e-voting-documentation/-/blob/342ea9e3339a70168ccfa3fef16eb096e811f9ef/System/System_Specification.pdf

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





Does Internet Voting Benefit down-Ballot Candidates? A Look at Patterns from 1992 to 2023 in Estonia

Mihkel Solvak[1(✉)] and Erik S. Herron[2]

[1] Johan Skytte Institute of Political Studies, University of Tartu, Ülikooli 18, Estonia 51003, Estonia
mihkel.solvak@ut.ee

[2] Department of Political Science, West Virginia University, Morgantown, USA
esherron@mail.wvu.edu

Abstract. This paper examines whether candidate performance differs between paper and internet voting in Estonian parliamentary elections. We argue that differences in ballot design, voting environment, and voter composition may shape vote shares by list position. Using candidate-level data from all nine parliamentary elections held between 1992 and 2023, we compare relative paper and internet vote shares within party district lists. First-placed candidates consistently receive larger paper vote shares, while candidates in positions 2 to 9 initially show a small internet voting advantage. This advantage declines over time and is no longer observable in 2023, suggesting that early mode-related differences weakened as internet voting diffused.

Keywords: Internet voting · ballot effect · voting behavior

1 Introduction

The paper examines whether voting mode affects the distribution of candidate support within party lists in Estonia's open-list proportional representation system. Its central question is whether candidates placed lower on a party's district list perform comparatively better among internet voters than among paper voters. The authors begin from the observation that paper and internet voters encounter meaningfully different voting environments. Paper voters make their choice in a polling station, face at least some time pressure, and generally have access only to the candidate lists and information they brought with them. Internet voters cast their ballot remotely, typically from home or work, and can spend more time examining the list or searching for additional information about individual candidates. These differences may reduce internet voters' dependence on candidate list position as an informational shortcut.

The comparison is also complicated by voter self-selection. Especially during the early years of internet voting, online voters differed from paper voters in

© The Author(s) 2027
D. Duenas-Cid et al. (Eds.): E-Vote-ID 2026, LNCS 17083, pp. 158–163, 2027.
https://doi.org/10.1007/978-3-032-39512-2_10

their political engagement, education, demographic profile, and familiarity with digital technologies [1,2]. Internet voters may therefore support lower-ranked candidates not only because of the electronic interface or remote voting environment, but also because they are more politically interested and more likely to seek information about candidates. The paper consequently treats observed differences as the combined product of ballot presentation, voting environment, and voter composition rather than as a pure causal effect of the online ballot.

Empirically, we compare candidates' shares of paper and internet votes within party district lists. The underlying logic is that a candidate placed further down the list may be considered to benefit from internet voting when that candidate receives a larger share of the party's internet votes than of its paper votes. The analysis covers all nine Estonian parliamentary elections held between 1992 and 2023 and contains 9,467 candidate-level election outcomes. Internet voting was first available in Estonian parliamentary elections in 2007, having initially been introduced in the 2005 local elections. Its use subsequently expanded rapidly, reaching 51.1 per cent of all votes cast in the 2023 parliamentary election. The inclusion of elections before 2007 allows the authors to describe the pre-existing relationship between list position and candidate support before internet voting became available.

2 Theory and Expectations

The theoretical discussion is grounded in research on ballot-position effects, particularly on Brockington's distinction between three levels of information used by voters [5]. Primary information comes from outside the ballot, such as campaign communication, media coverage, and previous knowledge of candidates. Secondary information is contained on the ballot itself, including candidate names, party labels, and characteristics that voters may infer from names. Tertiary information concerns the ballot's structure, especially the order in which candidates appear. When voters lack stronger information, they may rely on candidate order as a fallback heuristic.

In open-list proportional representation, list position may be more informative than a purely arbitrary ballot-order cue. Parties commonly place their strongest, most prominent, or electorally attractive candidates near the top of the district list. Consequently, high list position may convey genuine information about candidate status, experience, party support, and expected electoral strength. Voters may therefore use list position both as a cognitive shortcut and as a reasonable proxy for candidate quality or prominence [4–8].

Previous research shows that ballot order is associated with vote choice and electoral performance, although the effect is generally smaller than the influence of more substantive candidate and party characteristics [5,9]. Position effects also tend to be concentrated around particularly salient locations. The first candidate may enjoy a primacy advantage, the final candidate may be unusually visible, and candidates appearing first in a new column may benefit from the visual organization of the ballot. The relationship is therefore not expected to be linear;

moving from first to second place may matter much more than moving from tenth to eleventh place.

The simultaneous availability of paper and internet voting offers a opportunity to test some of these effects. The same candidates compete in the same districts and election, but voters encounter them through different interfaces and voting environments. Any systematic difference between a candidate's paper and internet vote shares may therefore reveal how voting mode is associated with candidate choice. Nevertheless, the paper does not claim that this constitutes a causal effect, because voters select their voting mode and the two electorates may still differ systematically.

Internet voting may weaken reliance on list order through two mechanisms. First, the remote voting environment gives voters more opportunity to inspect the candidate list and seek outside information. Unlike paper voters in a polling booth, online voters can theoretically pause, compare candidates, consult party websites, or search for individual names. Second, early internet voters were generally more politically engaged and better educated than paper voters. Such voters may possess more prior information about candidates or be more willing to acquire it, reducing their reliance on the first name displayed on the list.

Three principal expectations follow. First, internet voters should rely less heavily on list order. Candidates in positions two and below should consequently receive relatively larger internet vote shares, while first-placed candidates should perform relatively better among paper voters. Second, differences between paper and internet voting should vary non-linearly across list positions. Third, the relationship should change across elections. Because early internet voters were more distinctive and the use of internet voting gradually expanded, voting-mode differences may decline over time. Overall, the expectation is that internet voting will provide a modest relative benefit to candidates below the first position.

3 Data and Method

The empirical model examines how candidate list position is associated with three candidate-level outcomes. Because we expect the relationship to change across elections, we estimated a pooled model containing election-year fixed effects and interactions between election year and list position. Candidate age, gender, number of previous parliamentary candidacies, and incumbency status are included as controls.

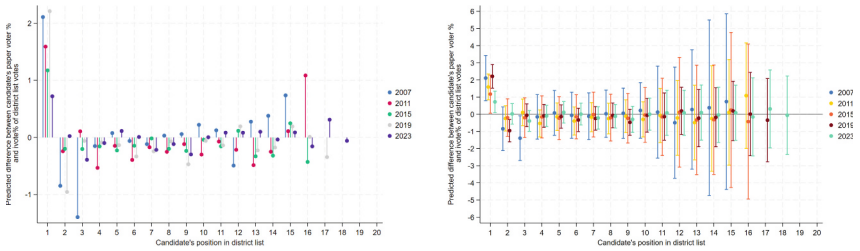
The analysis uses relative vote shares rather than raw vote totals to make candidates comparable across parties, districts, and elections. The data cover every Estonian parliamentary election held after the restoration of independence between 1992 and 2023. Elections before 2007 contain only paper voting but establish the historical structure of list-position effects.

4 Results

The descriptive results confirm a clear relationship between candidate list position and vote share. First-placed candidates normally receive approximately 40

to 50 per cent, and in some model estimates as much as 60 per cent, of all votes cast for their party in the district. Vote shares then decline sharply across the next few positions and flatten further down the list. This pattern appears in every election and for both paper and internet votes.

Despite the general similarity, first-placed candidates consistently perform slightly better among paper voters. Their paper vote shares exceed their internet vote shares by approximately one to two percentage points in the descriptive analysis. The regression estimates reproduced this pattern. Both paper and internet voting remain highly concentrated on the first-listed candidate, but the concentration is consistently somewhat lower among internet votes. Candidates in approximately positions 2 through 10 receive marginally larger internet vote shares than paper vote shares. The magnitude is small, and internet voting does not alter the underlying dominance of top-listed candidates.



(a) Point estimates (b) Point estimates with 95% CI

Fig. 1. Predicted paper an ivote share differences by candidate list position for all elections.

The difference between paper and internet vote shares provides the clearest test. Averaged across internet-voting elections, the first-placed candidate receives approximately 1.5% points more of the party's paper vote share than of its internet vote share. The second-placed candidate displays the reverse relationship, receiving around 0.5% points more from internet votes than from paper votes. A similar but progressively smaller internet-voting advantage is found for candidates in positions 3 through 9, see Fig. 1 for the estimated effects for the vote share difference.

This pattern only partly conforms to the theoretical expectation. Lower-ranked candidates do comparatively better among internet voters, but the advantage does not increase continuously as one moves down the ballot. Instead, the largest beneficiary is the second-placed candidate, after which the estimated difference generally decreases.

The most important result is the change over time. In 2007, the first-placed candidate received slightly more than two percentage points more of the party's paper vote share than internet vote share. By 2023, the difference had fallen to approximately 0.7 percentage points. The second-placed candidate had an internet voting advantage of roughly 0.8 percentage points in 2007, but this advantage

had disappeared by 2023. Comparable declines occur across the other positions. The election of 2019 is an exception because voting-mode differences were again comparatively pronounced, interrupting an otherwise relatively steady decline.

The substantive conclusion is therefore qualified. Internet voting was associated with a modest redistribution of candidate support during its early years. Paper voters concentrated slightly more of their support on the first candidate, while internet voters distributed marginally more support to candidates immediately below first place. Yet the differences were small compared with the overall effect of list position. First listed candidates remained dominant under both modes, and the basic structure of intraparty competition did not change.

The temporal decline suggests that the early differences may have reflected the distinctive composition of the first internet-voting electorate more than a permanent feature of the online ballot. Early internet voters were more politically engaged, more educated, and potentially more likely to consider candidate-level information. As internet voting became normalized and expanded to a majority of voters, online voters increasingly resembled the electorate as a whole. The corresponding disappearance of the down-ballot advantage is therefore consistent with a transitional adoption effect rather than a lasting transformation of candidate competition.

Disclosure of Interests. The authors have no conflict of interest to declare.

Disclosure of AI Usage. *ChatGPT 5.5 Thinking* was used to language edit the paper.

References

1. 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)
2. 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. Gov. Inf. Q. **33**(3), 453–459 (2016)
3. Brockington, D.: A low information theory of ballot position effect. Polit. Behav. **25**, 1–27 (2003). https://doi.org/10.1023/A:1022946710610
4. Jankowski, M., Frank, T.: Ballot position effects in open-list PR systems: the moderating impact of postal voting. Acta Politica **57**, 320–340 (2022). https://doi.org/10.1057/s41269-020-00192-2
5. Söderlund, P., von Schoultz, Å., Papageorgiou, A.: Coping with complexity: ballot position effects in the Finnish open-list proportional representation system. Elect. Stud. **71**, 102330 (2021)
6. Blom-Hansen, J., Elklit, J., Serritzlew, S., Villadsen, L.R.: Ballot position and election results: evidence from a natural experiment. Elect. Stud. **44**, 172–183 (2016). https://doi.org/10.1016/j.electstud.2016.06.019
7. Marcinkiewicz, K., Stegmaier, M.: Ballot position effects under compulsory and optional preferential-list PR electoral systems. Polit. Behav. **37**(2), 465–486 (2015). https://doi.org/10.1007/s11109-014-9294-0

8. Ortega Villodres, C.: Intra-party competition under preferential list systems: the case of Finland. Representation **40**(1), 55–66 (2008). https://doi.org/10.1080/00344890308523246
9. van Erkel, P.F.A., Thijssen, P.: The first one wins: distilling the primacy effect. Elect. Stud. **44**, 245–254 (2016). https://doi.org/10.1016/j.electstud.2016.09.002

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Author Index

A
Aranha, Diego F. 108
Araújo, Roberto 108

B
Blom, Michelle 1

E
Ek, Alexander 1
Erdmanis, Janis 18
Essex, Aleksander 35

F
Freestone, Jack 55

G
Garjan, Maryam Sheikhi 70
Giustolisi, Rosario 70

H
Herron, Erik S. 158
Hilt, Tobias 125
Høgåsen, Audhild 87

K
Kimura, Leonardo 108

L
Leskow, Dimitri 108
Leung, Dennis 55
Louistisserand, Léo 141

M
Mack, Christian 125
Moser, Florian 141

R
Ryan, Peter Y. A. 70

S
Silde, Tjerand 87
Simplicio, Marcos 108
Solvak, Mihkel 158
Spadafora, Chiara 87
Stark, Philip B. 35
Strømsnes, Amund Fredrik 87
Stuckey, Peter J. 1

T
Teague, Vanessa 1

V
Volkamer, Melanie 125
Vukcevic, Damjan 1, 55

© The Editor(s) (if applicable) and The Author(s) 2027
D. Duenas-Cid et al. (Eds.): E-Vote-ID 2026, LNCS 17083, p. 165, 2027.
https://doi.org/10.1007/978-3-032-39512-2