Pretty Understandable Democracy - A Secure and Understandable Internet Voting Scheme

Jurlind Budurushi, Stephan Neumann, Maina M. Olembo and Melanie Volkamer
CASED / TU Darmstadt
Hochschulstraße 10, 64289, Darmstadt, Germany
Name.Surname@cased.de

***Abstract*—Internet voting continues to raise interest. A large number of Internet voting schemes are available, both in use, as well as in research literature. While these schemes are all based on different security models, most of these models are not adequate for high-stake elections. Furthermore, it is not known how to evaluate the understandability of these schemes (although this is important to enable voters' trust in the election result). Therefore, we propose and justify an adequate security model and criteria to evaluate understandability. We also describe an Internet voting scheme, Pretty Understandable Democracy, show that it satisfies the adequate security model and that it is more understandable than Pretty Good Democracy, currently the only scheme that also satisfies the proposed security model.**

***Index Terms*—Cryptography, Internet Voting, Code Voting, Security Model, Understandability**

I. Introduction

Internet voting continues to be a topic of interest with widespread use in different contexts, for example, university president elections at the Université Catholique de Louvain. Even in Germany, where voting machines have been rejected, a recent survey [1] reveals that more than 50% of eligible voters would cast their vote over the Internet for federal elections.

Despite this interest, and the fact that many Internet voting schemes are already available, further research is needed regarding security and understandability. The underlying security model of most existing schemes is not adequate for high-stake elections. The problem with these schemes is that one single entity can violate secrecy and/or integrity, while in traditional elections at least two entities control each other (the four-eyes principle). For instance, in the Estonian voting scheme [12], trust is placed in one server component; in the Norwegian voting scheme [25], trust regarding secrecy is placed in each individual voter's computer; and in VeryVote [19] trust is placed on each voter not to violate secrecy.

Little attention has been paid to the understandability of Internet voting schemes and related understandability criteria in research literature. Consequently, those schemes which provide adequate security for high-stake elections have not yet been evaluated with respect to understandability for the average voter. However, understandability directly affects the trust that voters place on a voting scheme [3], [5]. Therefore, although these schemes provide adequate security, they are not likely to be used in real-world elections. As a result of this state of affairs, there is a need for an adequate security model, understandability criteria, and an Internet voting scheme that meets both.

In this paper, we describe a security model and justify why it is adequate for Internet voting in established democracies. In addition, understandability criteria are proposed. An Internet voting scheme - Pretty Understandable Democracy (PUD) - is developed. We evaluate this scheme and show that it ensures secrecy and integrity under the proposed adequate adversary model. Furthermore, we evaluate PUD using the understandability criteria and show that it is more understandable than Pretty Good Democracy [26], currently the only Internet voting scheme that satisfies the proposed security model.

II. Adequate Security Model

A security model consists of security criteria and an adversary model. In this section, we introduce both parts and justify the adequacy of the adversary model.

A. Security Criteria

Internet voting literature provides a number of standard security criteria catalogs [9], [29]. Certainly, the most important security criteria of Internet voting schemes are secrecy and integrity. We use the following definitions:

Secrecy: For each voter who casts a vote for an arbitrary candidate c, it holds that the adversary[1] cannot get more evidence about the fact that the voter selected c or any other selection c' as he can get from the final tally.

Integrity: The aggregation of all participating eligible voters' intentions[2] matches the declared election result.

Integrity is ensured if the following sub-criteria are fulfilled:

Encoded-as-intended Integrity: The participating voter's intention is correctly encoded. Note that a voter's intention might be encoded by techniques like encryption or permutation of candidates. In the following, we refer to a voter's encoded intention as her encoded vote.

Cast-as-encoded Integrity: The participating voter's encoded vote is correctly cast, that is, it correctly leaves the voter's platform.

Stored-as-cast Integrity: The participating voter's cast vote is correctly stored for tallying during the whole voting phase.

[1] Note, obviously the adversary in that case cannot be the voter herself, as she always knows her own intention.

[2] If a voter is coerced and follows the coercer's instructions, we consider this to be the voter's intention.

This work has been published in Proceedings of the Eighth International Conference on Availability, Reliability, and Security (p./pp. 198-207): IEEE Computer Society. ISBN: 978-0-7695-5008-4. DOI: http://dx.doi.org/10.1109/ARES.2013.27

Tallied-as-stored Integrity: All participating voters' stored votes are correctly tallied.
Eligibility Integrity: Only eligible voters' intentions are included in the election result.
Democracy Integrity: Only one intention per eligible voter is included in the election result.

Note that if an integrity sub-criterion is ensured without posing restrictions on the adversary, the sub-criterion is referred to in literature as verifiable. Ideally, all integrity sub-criteria should be verifiable [2], [8], [23]. Verifiability conflicts with the secrecy criterion such that tradeoffs between secrecy and integrity must be accepted.

B. Adversary Model

The adversary has the following capabilities:

- The adversary is able to corrupt one single entity from the set of authorities[3], voters, and voters' platforms.
- The adversary controls network channels between all entities, i.e., the network between platforms involved in the scheme as well as the network between humans, e.g., postal mail.

On the other hand, we assume the adversary to be restricted in the following way:

- The adversary is not able to break standard cryptography, such as ElGamal or Diffie-Hellman. This assumption is justified by the fact that long-term secrecy is not a crucial problem in established democracies.
- The adversary cannot coerce the voter (according to the definition by Juels et al. [21]). More precisely, the adversary cannot force voters to abstain from the election, control the voter during the whole voting phase, or force the voter to cast a vote in a randomized way. These three assumptions are justified by the fact that, in established democracies, voters in these cases of coercion would alert the police.
- The adversary cannot convince voters to participate in integrity violations. This assumption is justified by the fact that the voter might always vote differently from her intention and consequently violate integrity trivially.
- The adversary cannot obtain authentication material from voters. This assumption is justified by the fact that the voting process is based on authentication material[4] that is used to access further services.
- The adversary cannot trick the voters into phishing websites. This is justified by the fact that strong authentication is in place and voters know the authentic website from media reports and voting instructions.
- The adversary cannot corrupt more than one entity. This assumption is justified by the fact that in traditional elections two malicious poll workers can violate secrecy and integrity.

[3] Authorities are composed of the human, the platform used by that entity, as well as all hardware and software developers of the platform.

[4] For instance, in Estonia and Norway, eIDs have been used to authenticate eligible voters. In non-political elections, one might consider student IDs or Facebook, Google, or other similar platforms for authentication.

III. Understandability Criteria

Maaten [22] proposes increasing the overall understandability of Internet voting schemes by making them as easy to explain as possible. She, however, does not provide concrete criteria to measure the degree of understandability of Internet voting schemes. Independently, Essex et al. [10] propose guidelines to increase understandability within voting schemes. According to their guidelines, voting schemes should rely on a small set of simple cryptographic algorithms[5]. While their work focuses on improving the understandability of the tallying phase, we propose to apply these guidelines to all phases of Internet voting schemes. Accordingly, we propose the criterion *number of cryptographic algorithms in use* as a measure for the overall understandability of Internet voting schemes. This sub-criterion identifies how many cryptographic algorithms are applied in the Internet voting scheme. Examples of cryptographic algorithms are encryption, re-encryption, signing, permutation, and zero-knowledge proofs. A verifiable re-encryption mix-net consists of the cryptographic algorithms re-encryption, permutation, and zero-knowledge proofs.

It becomes apparent that, even if the number of cryptographic algorithms is low, these algorithms might be used several times and in an interfering manner such that understandability of the overall Internet voting scheme decreases. Therefore, we propose as a second criterion to measure understandability of Internet voting schemes by the *number of essential process steps*. This sub-criterion identifies the number of essential process steps affecting an individual voter's vote. Essential process steps are those containing cryptographic algorithms. We focus on the number of applications of cryptographic algorithms affecting an individual voter's vote because these are the steps that the voter must understand. An example of an essential process step is the encryption of the voter's vote.

Note that as future work we will concentrate on simplicity of cryptographic algorithms and, based on the results, extend the proposed sub-criteria.

IV. Related Work

Internet voting has been studied since the early 1980s, when Chaum's seminal work [7] outlined the idea of using mix-nets to ensure secrecy of the vote. Many schemes have been proposed which look at conducting secure elections over the Internet, for instance Benaloh and Tuinstra [4], JCJ [21], the JCJ extension Civitas [8], and Helios [2]. One significant drawback of these schemes is that in order to ensure secrecy (in some schemes, even integrity) the voter's platform is assumed to be trustworthy. These schemes do not satisfy our security model, because one entity (voter's platform) can violate secrecy or integrity.

Securely voting over untrustworthy platforms was initially addressed by Chaum's SureVote scheme [6], the first code

[5] The authors in [10] also consider efficiency. However, we do not see a clear relation between efficiency and understandability, and the authors do not provide a derivation of this relation.

voting scheme. In such schemes, voters get a code sheet over an out band channel (e.g. snail mail). In the code sheets, candidates are assigned to random, unique codes, thus voters cast codes rather than candidates. Code voting has been extended in [16], [19], [17], [18], [20], [13], [14], [15] and [26]. The schemes in [6], [16], and [19] assume the voter to be honest in order to ensure secrecy. Other extensions of code voting, [17], [18], [20] assume a trustworthy voting- and voter-specific smart card for secrecy and integrity. All these schemes do not satisfy our security model, because one entity (either voter or smart card) can violate secrecy or integrity.

Finally, the code voting based schemes introduced in [13], [14], [15], rely on one voting server for integrity. Hence, these schemes also do not satisfy our security model. To the best of our knowledge, the only Internet voting scheme that meets the criteria of secrecy and integrity under our adversary model is Pretty Good Democracy (PGD) [26]. Our scheme will be shown to be more understandable than PGD in Section VII.

V. Description of Pretty Understandable Democracy

This section describes Pretty Understandable Democracy (PUD). This Internet voting scheme is based on the concept of code voting, the only concept to effectively defend against a malicious voter's platform[6]. Correspondingly, we first give a short overview of the code voting concept.

A. Code Voting

The concept of code voting was first introduced in Chaum's SureVote scheme [6]. The motivation of code voting is to enable Internet voting without the need to trust the voter's platform with respect to secrecy and integrity. Each eligible voter is issued, via an out of band channel (e.g. conventional mail), a code sheet as shown in Figure 1. Note that every voter gets a different code sheet. In contrast to other Internet voting schemes, in code voting the voter casts a voting code instead of her preferred candidate. In case a voter, who possesses the code sheet shown in Figure 1, wanted to cast a vote for Alice, she would submit the ballot ID, namely 34255, and the voting code next to Alice, namely 51948. The voting server would respond with the corresponding acknowledgment code, 71468.

Ballot ID: 34255		
Candidate	Voting Code	Acknowledgment Code
Alice	51948	71468
Bob	23766	53286
Eve	41948	35468

Fig. 1. Code sheet

As malware on the voter's platform does not know which candidate is represented by the voting code, an untrustworthy voter's platform cannot break secrecy. The acknowledgment code proves that the voting server received the correct voting code. Any modification by the voter's platform to the voter's code would be detected as the acknowledgment code will not match the one on the voter's code sheet.

B. Code Sheets in PUD

The code sheets used in PUD consist of three parts (i.e. three different pieces of paper), two parts containing codes and one part containing a permuted list of candidates. Each code sheet part is generated by a different authority. The three code sheet parts are linked by their index to one code sheet.

An example of one part of the code sheet containing codes is depicted in Figure 2. This part with accompanying index i is generated by authority A, whose identity is also indicated, next to the acknowledgment code. $Code_{A,i,1} \ldots, Code_{A,i,n}$ denote n random, unique codes and $Ack_{A,i}$ denotes a random, unique acknowledgment code. Similarly, an authority B generates the second part of the code sheet containing codes for index i.

i
$Code_{A,i,1}$
$\vdots$
$Code_{A,i,n}$
A: $Ack_{A,i}$

Fig. 2. Code sheet part generated by authority A with index i

The third part of the code sheet with index i is generated by an authority C and consists of the list of n candidates, randomized according to a secret permutation ϕ_i. The code sheet part containing the candidates is shown in Figure 3 and the complete code sheet for PUD is illustrated in Figure 4.

i
$\phi_i(Candidate_1)$
$\vdots$
$\phi_i(Candidate_n)$
–

Fig. 3. Code sheet part generated by authority C with index i

i	i	i
$Code_{A,i,1}$	$Code_{B,i,1}$	$\phi_i(Candidate_1)$
$\vdots$	$\vdots$	$\vdots$
$Code_{A,i,n}$	$Code_{B,i,n}$	$\phi_i(Candidate_n)$
A: $Ack_{A,i}$	B: $Ack_{B,i}$	–

Fig. 4. Code sheet in PUD

For a code sheet with index i, the voting code for the candidate in the p-th position is the concatenation of the corresponding codes in the p-th position:

$$Code_{i,p} = Code_{A,i,p} \parallel Code_{B,i,p}$$

Accordingly, the voting acknowledgment code of this code sheet is the concatenation of the acknowledgment codes:

$$Ack_i = Ack_{A,i} \parallel Ack_{B,i}$$

[6]Note that, the assumption that a voter's platform is trustworthy would violate the proposed security model because one single entity - the voter's platform - could already violate secrecy and integrity.

C. Entities

Here, we outline the involved entities and their key roles.

Authorities:

- Trustees (T) are involved in the setup phase, in particular in generating a threshold public/secret key pair (pk_T, sk_T) for encryption/decryption. Each Trustee possesses a share of the secret key. Trustees are also involved in the tallying phase.
- The Distribution Authority (*DA*) is involved in the setup phase; together with the Trustees, it anonymizes, audits and distributes code sheets. Thus, both know the election register.
- The Registration Authority (*RA*)[7], in the setup phase, generates the code sheet parts containing the permuted list of candidates. *RA* is also involved in the voting phase and knows the election register.
- The Voting Authority *1* (VA_1), in the setup phase, generates codes. VA_1 is also involved in the voting phase. Furthermore VA_1 holds a signing key.
- The Voting Authority *2* (VA_2) has a similar functionality as VA_1.
- The Bulletin Board (*BB*) is involved in all phases. Any entity has read access, all authorities (except *DA*) have write access. All data published on the *BB* are signed by the sending authority[8]. *BB* provides different sectors for all phases.

Voter: The Voter (V) is a citizen who is eligible to participate in the election and cast a vote.

Voter's Platform: The Voter's Platform (*VP*) is the platform from which the voter casts her vote.

D. Election Setup

The election setup phase consists of key generation as well as generating, committing on, auditing, anonymizing and distributing code sheets.

Generating Keys: The Trustees generate a threshold public/secret key pair (pk_T, sk_T) for encryption/decryption in a distributed manner. All authorities (except *DA*) generate SSL key pairs. In addition, *RA*, VA_1 and VA_2 generate signing keys.

Generating Code Sheets: *RA* generates the part of each code sheet containing the candidates: It randomizes the canonical order of the candidate list for each code sheet according to a secret permutation and prints the index and the randomized candidate list on a sheet of paper (ref. to Figure 3). *RA* inserts its sheets of paper into privacy-protected sealed envelopes. The corresponding indexes are printed on the envelopes and sent to *DA*.

VA_1 and VA_2 independently generate random, unique codes for each candidate and each code sheet. They also independently generate random unique acknowledgment codes for each code sheet. Note that the acknowledgment codes must not match codes for candidates. VA_1 and VA_2 independently print this information on a sheet of paper (ref. to Figure 2). VA_1 and VA_2 also insert their sheets of paper into privacy-protected sealed envelopes, print the corresponding indexes on the envelopes and send them to *DA*.

Note that more code sheets than eligible voters must be generated to enable auditing of code sheets.

Committing on Code Sheets: After generating the code sheet parts, *RA*, VA_1 and VA_2 'commit' on their respective parts: Committing is done by encrypting corresponding parts with the Trustees' public key pk_T and publishing the encryptions under the accompanying index in the setup phase sector of *BB*, see Figure 5. Note that committing is needed in order to detect malicious *RA*, VA_1, and VA_2 distributing invalid code sheets.

Bulletin Board Setup Phase Sector
$\vdots$
i
$\{\phi_i(Candidate_1)\}_{pk_T} \ldots \phi_i(Candidate_n)\}_{pk_T}$
$\{Code_1 \ldots Code_n; Ack_{VA_1}\}_{pk_T}$
$\{Code_1 \ldots Code_n; Ack_{VA_2}\}_{pk_T}$
$\vdots$

Fig. 5. Content of BB at the end of the setup phase

Auditing Code Sheets: Afterwards, *DA* and the Trustees start with the auditing process, shown in Figure 6: The Trustees randomly select code sheets to be audited. The corresponding data for each code sheet to be audited is downloaded from the setup phase sector of *BB*. The downloaded data is decrypted by a threshold set of Trustees. The decrypted data is matched against the content of the corresponding envelopes. The audited code sheets are then discarded. Note, this process can be observed by the general public, e.g., by video-streaming the process over the Internet.

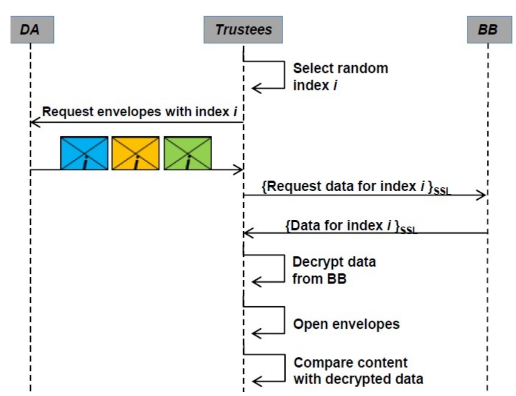

Fig. 6. Auditing process

Anonymizing and Distributing Code Sheets: After the auditing process, *DA* in cooperation with the Trustees anonymize and distribute the remaining envelopes to eligible voters, shown in Figure 7: All envelopes sharing the same index

[7] *RA* was referred to as authority C in the previous subsection.

[8] Sending authorities compute one signature over all data in one protocol step. Note, in Figures 5 and 9 the signatures are not illustrated.

are placed into neutral envelopes[9]. These are put into a box and shuffled. After permuting, *DA* and the Trustees take the anonymized neutral envelopes out of the box, print voters' addresses on the envelopes and send them to the corresponding addresses.

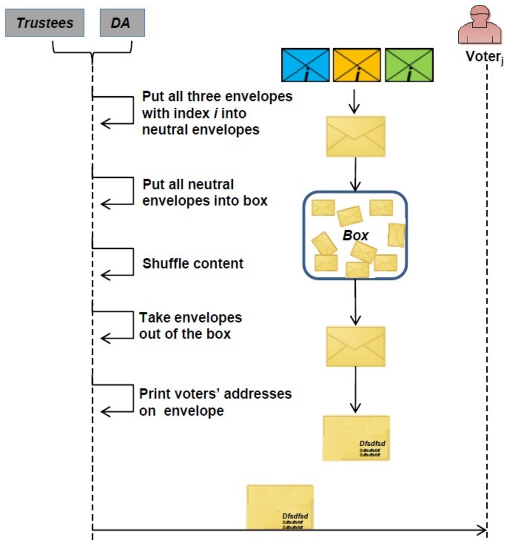

Fig. 7. Anonymization and distribution process

E. Voting

The voter receives an envelope and checks that it contains the three code sheet parts, that the three code sheet parts are in privacy-protected sealed envelopes, and that the envelopes share the same index. The voter opens the three envelopes and combines the three code sheet parts in an order that is publicly known.

The vote casting process is shown in Figure 8. In order to vote, the voter authenticates herself to the voting website, which is hosted by *RA*. For authentication, for instance, an eID card can be used. *RA* verifies that the voter is eligible to vote and that she has not yet cast a vote. The voter's communication with *RA* as well as any other communication in the voting phase are both secured by SSL. To cast a vote, the voter enters the voting code matching the candidate of her choice on the voting website.

RA forwards the first part of the voting code to VA_1 and the second part to VA_2. First, VA_1 and VA_2 check whether the received code is from a code sheet (index on *BB*) for which no code has yet been cast. Then, they deduce the index and the acknowledgment code of the code sheet (based on the received code) and the corresponding position of the code. Thereafter, they request and obtain the encryption of the candidate for the index and the position from *BB* (ref. to Figure 5, first row after the index i). VA_1 and VA_2 independently re-encrypt the received ciphertext to $\{\phi_i(Candidate_p)\}'pk_T$ and $\{\phi_i(Candidate_p)\}''pk_T$. After this, they send the re-encrypted ciphertexts to *BB*. *BB* publishes the received data and sends a confirmation to VA_1 and VA_2. Figure 9 illustrates the information on *BB*. After having received the confirmation, VA_1 and VA_2 forward the previously deduced acknowledgment codes to *RA*. *RA* concatenates these codes into the voting acknowledgment code, which it sends to the voter. *RA* changes the voter's status in the election register.

[9]By neutral envelopes we mean that the envelopes look the same.

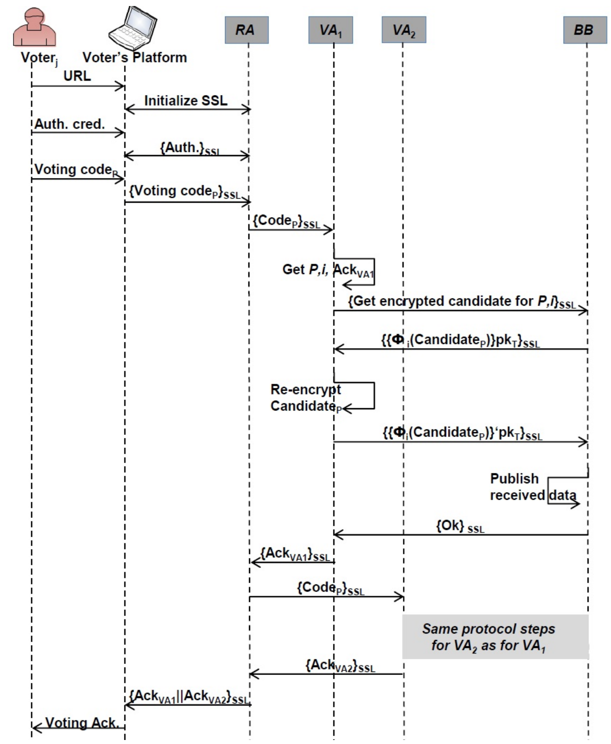

Fig. 8. Vote casting process

Bulletin Board Voting Phase Sector	
Column 1	Column 2
⋮	
$\{\phi_i(Candidate_p)\}'pk_T$	$\{\phi_i(Candidate_p)\}''pk_T$
⋮	

Fig. 9. Content on BB during the voting phase

F. Tallying

After the voting phase, each row of *BB* corresponds to a successfully cast vote (ref. to Figure 9). The tallying process is shown in Figure 10. Before the process starts, *RA* sends the total number of voters who have cast a vote to *BB*. The general public can check that this number matches the number of rows on *BB*. The Trustees request the re-encrypted ciphertexts and *BB* sends back the data re-encrypted by VA_1 and VA_2, corresponding to column 1 and column 2 of *BB*'s voting phase sector. The Trustees sum up the content of each individual column homomorphically. The encrypted sums are then decrypted by a threshold set of Trustees. The Trustees compare the decrypted sums, and if they match, the *election* *result* is declared to be the matching sum. Finally, the Trustees publish the ZKPs for correct decryption and the *election result* on *BB*.

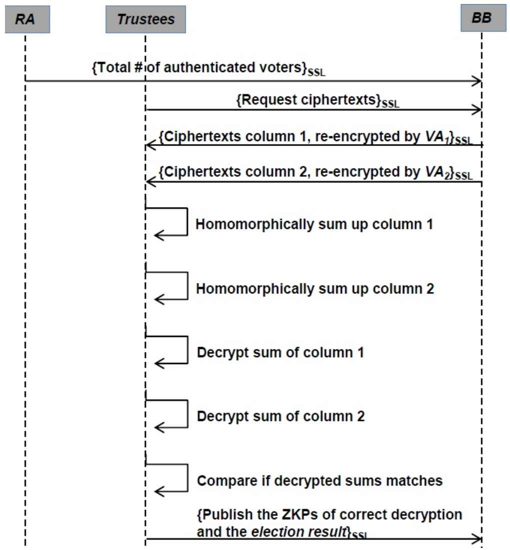

Fig. 10. Tallying process

VI. Security Analysis

This section is dedicated to the security analysis of Pretty Understandable Democracy (PUD). We show that PUD satisfies our security model. Before diving into the details of the analysis, we start with explaining the methodology we use for the analysis and some preliminary considerations.

A. Methodology

For our security analysis, we use *resilience terms* as proposed and considered for Internet voting schemes in [24], [27], [28] in order to show that no single entity can violate secrecy or integrity under the defined assumptions. Given a criterion $\mathcal{C}$ (in our case secrecy or integrity), the resilience term

$$(a+b) \text{ out of } (M,N); (c+d) \text{ out of } (O,P)$$

expresses that a entities out of the set of entities M and b entities out of N or that c entities out of O and d entities out of P, must collaborate in order to violate $\mathcal{C}$.

If, for secrecy or integrity, the resilience term of an Internet voting scheme is $t = t_1; \ldots ; t_n$ with $t_i = (a_{i,1} + \cdots + a_{i,m_i})$ out of $(A_{i,1}, \ldots , A_{i,m_i})$ for all $1 \leq i \leq n$ and $\sum_{j=1}^{m_i} |a_{i,j}| \geq 2$ holds, then these criteria are satisfied under the adversary model. Note, in order to break secrecy or integrity each entity can use the knowledge obtained as part of the scheme and any public knowledge. Entities can also deviate from the original protocol specification (e.g. modify or delete stored data). The voter is not considered throughout the integrity resilience term derivation because of the assumption that the adversary cannot convince voters to participate in integrity violations.

We determine the secrecy resilience terms according to the methodology described in [24]. We briefly explain this methodology and give a supporting example: All entities' knowledge (including public knowledge) is modeled in so-called knowledge sets. The basic knowledge sets of an adversary are determined to be elements of the power set of the entities' knowledge sets he is able to corrupt. Each basic knowledge set of an adversary is extended, based on the deduction system proposed in [24]. An adversary, being able to corrupt a particular set of entities, can break secrecy if the corresponding extended knowledge set contains the relation between a voter and her selected candidate. The entities being corrupt build the basis for deriving a resilience term.

For example, assume a scheme with two entities A and B. An entity X's knowledge set is denoted by $K(X)$. Assume $K(A)$ and $K(B)$ are defined as follows:

$$\begin{aligned} K(A) &:= \{R(voter(i), token(j))\} \\ K(B) &:= \{R(token(j), candidate(k))\} \end{aligned}$$

Note that R denotes the relation between two terms, while i, j, k are indexes of voters, tokens, and candidates. The set of basic adversary knowledge sets (*BAK*) is given as follows:

$$\begin{aligned} BAK \;:=\; \{&\emptyset, \{R(voter(i), token(j))\}, \\ &\{R(token(j), candidate(k))\}, \\ &\{R(voter(i), token(j)), \\ &\quad R(token(j), candidate(k))\}\} \end{aligned}$$

Applying the deduction system proposed in [24] for this simple example extends the fourth extended knowledge set by the term $R(voter(i), candidate(k))$. Correspondingly, an adversary being able to corrupt entity A and B would be able to break secrecy. The resilience term in this case is $t = 2$ out of $\{A, B\}$. Thus, under our adequate security model such a scheme would not violate secrecy.

Due to the fact that a similar approach for analyzing integrity is missing, an informal analysis is provided in the remainder of this work.

B. Preliminary Considerations

In this subsection, we discuss the impact on the security analysis of an adversary controlling the Internet and the postal channel.

The consequence of an adversary controlling the Internet is that all messages interchanged between entities become public knowledge as the adversary could publish this information (e.g. anonymously). However, in PUD, all communication over the Internet is secured by SSL with respect to secrecy and integrity, and the adversary is restricted with respect to breaking standard cryptography. Therefore, the adversary does not get any advantage which he can use to violate secrecy or integrity.

Due to the fact that in PUD code sheets are distributed in sealed and privacy-protected envelopes, it is assured that envelopes cannot be opened and closed again, neither can they be replaced without detection. Therefore, the adversary controlling the postal channel between the Distribution Authority (*DA*) and voters does not get any advantage which he can use to violate secrecy or integrity. In summary, controlling all channels between involved entities does not have any impact on the security analysis.

C. Result of the Analysis

In this subsection, we deduce the resilience terms and evaluate them according to our security model. We start with secrecy and then address the different integrity sub-criteria.

Secrecy: We first identify the entity's knowledge sets and provide the result of the resilience analysis. Note that |THR| is the number of Trustees that are needed to reconstruct their secret key, sk_T. *AV* denotes the set of all eligible voters, while *PV* denotes the set of participating voters.

The **voter** $v(i)$ knows the relation between her identity and her code sheet index *index*$(v(i))$. She also knows the relation between her code sheet index and all voting codes related to that index. The voter knows the relation between her identity and her acknowledgment code. Finally, the voter knows the relation between voting codes and the corresponding candidates due to the printed code sheet.

1) $R(v(i), index(v(i)))$
2) $R(index(v(i)), code_{VA_a}(index(v(i))))$, $\forall a \in \{1,2\}$
3) $R(v(i), ack{-}code_{VA_a}(v(i)))$, $\forall a \in \{1,2\}$
4) $R(code_{VA_a}(index(v(i))),$ $cand(code_{VA_a}(index(v(i)))))$, $\forall a \in \{1,2\}$

The **Registration Authority** *RA* knows the relation between candidates and encrypted[10] candidates stored in the voting phase sector of the *BB*. *RA* furthermore knows the relation between a voter's identity and her cast voting code. Finally, *RA* knows the relation between a voter's identity and her acknowledgment code.

1) $R(cand(code_{VA_a}(index(v(i)))),$ $enc_{RA}(cand(code_{VA_a}(index(v(i))))))$, $\forall a \in \{1,2\}, \forall i \in \{1,\ldots,|PV|\}$
2) $R(v(i), cast{-}code_{VA_a}(v(i)))$, $\forall a \in \{1,2\}, \forall i \in \{1,\ldots,|PV|\}$
3) $R(v(i), ack{-}code_{VA_a}(v(i)))$, $\forall a \in \{1,2\}, \forall i \in \{1,\ldots,|PV|\}$

The **Voting Authority** VA_1 knows the relation between all codes generated by VA_1 for each index. VA_1 also knows the relation between all cast codes intended for itself and the corresponding code sheet index. In addition, VA_1 knows the relation between codes and candidate encryptions on *BB*'s setup phase sector. Moreover, VA_1 knows the relation between ciphertexts containing candidates and re-encryptions of these ciphertexts posted on *BB*. Finally, VA_1 knows the relation between codes and acknowledgment codes generated by itself.

1) $R(code_{VA_1}(index(v(i))), index(v(i)))$, $\forall i \in \{1,\ldots,|PV|\}$
2) $R(cast{-}code_{VA_1}(v(i)), index(v(i)))$, $\forall i \in \{1,\ldots,|PV|\}$
3) $R(code_{VA_1}(index(v(i))),$ $enc_{RA}(cand(code_{VA_1}(index(v(i))))))$, $\forall i \in \{1,\ldots,|PV|\}$
4) $R(enc_{RA}(cand(code_{VA_1}(index(v(i))))),$ $re{-}enc_{VA_1}(enc_{RA}(cand(code_{VA_1}(index(v(i)))))))$, $\forall i \in \{1,\ldots,|PV|\}$
5) $R(code_{VA_1}(index(v(i))), ack{-}code_{VA_1}(v(i)))$, $\forall i \in \{1,\ldots,|PV|\}$

We do not provide the knowledge set for **Voting Authority** VA_2, because the knowledge is specified analogously.

Voter i's **Voter Platform** *VP* knows the relation between that voter's identity and her cast voting code. Furthermore, *VP* knows the relation between the voter's identity and her acknowledgment code.

1) $R(v(i), cast{-}code_{VA_a}(v(i)))$, $\forall a \in \{1,2\}$
2) $R(v(i), ack{-}code_{VA_a}(v(i)))$, $\forall a \in \{1,2\}$

The **BB** knows the relation between indexes, and encryptions of voting codes generated for that index as well as encryptions of candidates prepared for that index. Furthermore, *BB* knows the relation between re-encryptions of candidates generated by VA_1 and VA_2.

1) $R(index(v(i)),$ $enc_{RA}(code_{VA_a}(index(v(i)))))$, $\forall a \in \{1,2\}, \forall i \in \{1,\ldots,|AV|\}$
2) $R(index(v(i)),$ $enc_{RA}(cand(code_{VA_a}(index(v(i))))))$, $\forall a \in \{1,2\}, \forall i \in \{1,\ldots,|AV|\}$
3) $R(re{-}enc_{VA_1}(enc_{RA}(cand(code_{VA_1}(index(v(i)))))),$ $re{-}enc_{VA_2}(enc_{RA}(cand(code_{VA_2}(index(v(i)))))))$, $\forall a \in \{1,2\}, \forall i \in \{1,\ldots,|PV|\}$

Each **Trustee** *Tr* knows a share of the secret key sk_T. The **Distribution Authority** only knows voters' identities.

Applying the deduction system for the identified knowledge sets, the following secrecy resilience term results:

$$\begin{aligned} t_{sec} \;=\;& (1+1) \text{ out of } (\{V\}, \{VP, RA, VA_1, VA_2\}; \\ & (1+1) \text{ out of } (\{RA\}, \{VA_1, VA_2\}); \\ & (1+1+|THR|) \text{ out of } (\{VP\}, \{VA_1, VA_2\}, T); \\ & (1+1+1) \text{ out of } (\{VP\}, \{VA_1, VA_2\}, \{RA\}) \end{aligned}$$

Integrity: We consider encoded-as-intended, cast-as-encoded, stored-as-cast, tallied-as-stored, eligibility, and democracy separately.

a) Encoded-as-intended (eai) Integrity: The voter must be sure that the information printed on her code sheet with index i matches the encrypted information for index i on *BB*. Throughout the auditing process, any observer can verify that this information matches for randomly selected code sheets. Hence, the resilience term is:

$$t_{eai} \;=\; \infty$$

b) Cast-as-encoded (cae) Integrity: The only way for *VP* to successfully manipulate voting codes provided by the voter before they are cast, is to know another valid voting code of this voter. The voter's platform (*VP*) must collaborate with VA_1

[10] Ciphertexts are generated using the Trustees' public key pk_T.

and VA_2, in order to get this information[11]. Consequently, the resilience term for cast-as-encoded integrity is:

$$t_{cae} \quad = \quad 3 \text{ out of } \{VP, VA_1, VA_2\}$$

c) Stored-as-cast (sac) Integrity: There are two groups of entities identified as capable of violating stored-as-cast integrity. The first group involves VA_1 and VA_2. If both authorities agree on selecting the same encryption of a different candidate from *BB*, they can successfully violate this sub-criterion. The second group consists of *BB* and *RA*. *BB* might remove individual ciphertexts. Furthermore, if *RA* correspondingly adapts the number of voters who cast a vote, both authorities can successfully violate this sub-criterion. The resilience term for stored-as-cast integrity is:

$$\begin{aligned} t_{sac} \quad = \quad & 2 \text{ out of } \{VA_1, VA_2\}; \\ & 2 \text{ out of } \{RA, BB\} \end{aligned}$$

d) Tallied-as-stored (tas) Integrity: Throughout the tallying process, any observer can verify that the Trustees correctly built the homomorphic sum over all signed, re-encrypted candidates from *BB*'s voting phase sector and correctly decrypted the computed sum, which is the election result. Consequently, the resilience term for tallied-as-stored integrity is:

$$t_{tas} \quad = \quad \infty$$

e) Eligibility (e) Integrity: There are two groups of entities identified as capable of violating eligibility integrity. The first group involves *RA* and *V*. If the voter forwards her code sheet to *RA*, then *RA* can cast one voting code from that voter's code sheet, thereby violating this sub-criterion. The second group consists of *RA*, VA_1, and VA_2. Rather than receiving code sheets from the voters, *RA* might receive valid voting codes from VA_1 and VA_2. Thereby, this group would succeed in violating eligibility integrity. Consequently, the following resilience term results:

$$\begin{aligned} t_{e} \quad = \quad & 2 \text{ out of } \{RA, V\}; \\ & 3 \text{ out of } \{RA, VA_1, VA_2\} \end{aligned}$$

f) Democracy (d) Integrity: If a malicious voter intends to cast several votes, and *RA* allows the voter to cast voting codes several times, and furthermore VA_1 and VA_2 publish corresponding re-encryptions on *BB*, then they can violate democracy integrity.

$$t_{d} \quad = \quad 4 \text{ out of } \{V, RA, VA_1, VA_2\}$$

Table I summarizes the results of the security analysis on PUD, showing that PUD satisfies our security model.

VII. Understandability Analysis

In this section, we compare Pretty Understandable Democracy (PUD) with Pretty Good Democracy (PGD) [26] (the only Internet voting scheme meeting the security model from Section II), with respect to understandability and show that

[11]The outlined collusion would merely result in a randomization attack.

TABLE I
Summary of the Security Analysis of PUD.

Secrecy	$(1+1)$ out of $(\{V\}, \{VP, RA, VA_1, VA_2\}$; $(1+1)$ out of $(\{RA\}, \{VA_1, VA_2\})$; $(1+1+\lvert THR\rvert)$ out of $(\{VP\}, \{VA_1, VA_2\}, T)$; $(1+1+1)$ out of $(\{VP\}, \{VA_1, VA_2\}, \{RA\})$
Encoded-as-intended	∞
Cast-as-encoded	3 out of $\{VP, VA_1, VA_2\}$
Stored-as-cast	2 out of $\{VA_1, VA_2\}$; 2 out of $\{RA, BB\})$
Tallied-as-stored	∞
Eligibility	2 out of $\{RA, V\}$; 3 out of $\{RA, VA_1, VA_2\}$
Democracy	4 out of $\{V, RA, VA_1, VA_2\}$

PUD is more understandable. In order for the reader to follow the discussion, we first briefly summarize PGD. In this summary, cryptographic algorithms are highlighted with **bold** font, while the number of their applications is given in parentheses.

A. Pretty Good Democracy

Pretty Good Democracy is a code voting scheme. It consists of election setup, voting and tallying phases which we first describe. The entities involved are Bulletin Board, Voting Server, Trustees, Voting Authority, Registrar, Returning Officer, Voter, Voter's Platform, and Clerks. All proofs are posted on the Bulletin Board. The code sheet consists of the canonical order of candidates and corresponding voting codes, as well as one acknowledgement code.

1) Election Setup: Before the election, the Trustees run a distributed key generation protocol (according to [11], t^2 **encryptions** and $2{*}t^2$ **commitments** are deployed) to establish a common public key pk_T, such that each Trustee holds a share of the respective secret key sk_T. The Voting Authority generates $\lambda \cdot v \cdot (c+1)$ distinct voting codes and **encrypts** them with pk_T (each voter's code sheet contains $c+1$ voting codes which are encrypted). The factor λ serves to generate enough voting codes to enable auditing, v is the number of voters and c the number of candidates.

After the generation and encryption of voting codes, the Clerks anonymize these voting codes using a verifiable re-encryption mix-net. Each Clerk **permutes** the voting codes, **re-encrypts** them, and proves for each individual voting code the correct proceeding with a **zero-knowledge (ZK) proof** ($C{*}(c{+}1)$ permutations, re-encryptions, and ZK proofs, where C is the number of Clerks). They place the anonymized voting codes in a table, called the P-table, which has $c+1$ columns and $\lambda \cdot v$ rows. Each row of the P-table corresponds to a valid code sheet. The code sheet's *ID* corresponds to the row number. The first c columns correspond to the candidates and the last column to the acknowledgment code.

After the generation of the P-table, the Trustees distributively **decrypt** the P-table (each voter's code sheet contains $c+1$ codes to be decrypted by the Trustees). Together with the Registrar, they construct $\lambda \cdot v$ code sheets. Afterwards, the Registrar audits a subset of the code sheets and prints the remaining code sheets on paper. The Registrar puts the printed code sheets in privacy-protected envelopes and sends them to the Returning Officer, who distributes them to eligible voters.

To prepare a secrecy-maintaining tallying phase, the P-table must be further anonymized. Hence, the Clerks run a second verifiable re-encryption mix-net (**permutation**, **re-encryption**, and **ZK proof** as above), while only permuting voting codes within each row of the P-table. Furthermore, the Clerks add their encrypted, individual permutation factors to the encrypted factors of the previous Clerks (t **re-encryptions** and **ZK proofs**). The resulting factors are appended to each permuted code sheet, resulting in the Q-table. Both, the P-table and the Q-table are posted on the Bulletin Board.

2) Voting: After the voter receives her individual code sheet from the Returning Officer, she uses the code sheet to cast her vote. The voter first authenticates herself to the Voting Server. She then sends her code sheet's *ID* together with the voting code appearing next to her preferred candidate. After the Voting Server receives the voter's voting code, it **encrypts** the code (1 encryption) and generates a **ZK proof** of knowledge stating that it knows the corresponding plaintext (1 ZK proof of knowledge). The encryption of the voting code together with the zero-knowledge proof is posted within row *ID* of the Q-table (on the Bulletin Board). After the encrypted voting code has been posted, the Trustees carry out plaintext equivalence tests between the encrypted voting code posted by the Voting Server and the voting codes from the Q-table in row *ID*. In worst-case, the plaintext equivalence test is run c times to find a match. Throughout the plaintext equivalence test, each Trustee **commits** on a blinding factor used to obfuscate the underlying plaintexts ($c*t$ commitments). Afterwards, the Trustees deploy a distributed **blinding** (t blindings) and distributively **decrypt** (t decryptions) the blinded ciphertext. They generate a **ZK proof** of the equality of discrete logarithms in order to ensure correct decryption (t ZK proofs).

If a match is identified, the corresponding encryption is marked in the Q-table by the Trustees. The Trustees furthermore distributively **decrypt** the acknowledgment code within row *ID* (t decryptions), **prove** the correctness of the decryption (t ZK proofs). They return this code to the Voting Server, which forwards it to the voter. The voter checks that the returned acknowledgment code matches the acknowledgment code on her code sheet.

3) Tallying: After the voting phase, marked encryptions in the Q-table must be interpreted in order to compute the election result. Therefore, for each row in the Q-table, the column number of the marked encryption and the encrypted permutation factor are paired. Resulting pairs are anonymized using a third verifiable re-encryption mix-net (t **permutations**, t **re-encryptions**, and t **ZK proofs**). After the anonymization process, the encrypted permutation factor is distributively **decrypted** (t decryptions) and the correctness of decryptions is proven by the Trustees with **ZK proofs** (t ZK proofs). Finally, the marked column number can be associated to the original column number, namely the corresponding candidate selection. After all column numbers have been interpreted, the election result is the sum over all candidate selections.

B. Comparing Understandability of PUD and PGD

We compare PUD and PGD using the understandability criteria described in Section III: number of cryptographic algorithms in use and number of essential process steps. The results are summarized in Table II and outlined here:

TABLE II
CRYPTOGRAPHIC ALGORITHMS AND NUMBER OF ESSENTIAL STEPS

	PUD	PGD
Encryption	$2+c+t^2$	$2+c+t^2$
Distributed Key Generation	t^2	t^2
Code Sheet Generation	$c+2$	$c+1$
Voting		1
Re-Encryption	2	$2*C*(c+1)+2*t$
Code Sheet Generation		$C*(c+1)$
Setup		$C*(c+1)+t$
Voting	2	
Tallying		t
Decryption	t	$t*(2*c+3)$
Code Sheet Generation		$t*(c+1)$
Plaintext Equivalence Test		$t*c$
Voting		t
Tallying	t	t
Permutation	c	$2*C*(c+1)+1$
Code Sheet Generation	c	$2*C*(c+1)$
Tallying		t
ZK Proofs	t	$2*C*(c+1)+1+t*(4+c)$
Code Sheet Generation		$C*(c+1)$
Setup		$C*(c+1)+t$
Proof of Knowledge		1
Plaintext Equivalence Test		$t*c$
Voting		t
Tallying	t	$2*t$
Commitments	$2*t^2$	$2*t^2+t*c$
Distributed Key Generation	$2*t^2$	$2*t^2$
Plaintext Equivalence Test		$t*c$
Blinding		t
Plaintext Equivalence Test		t
Signatures	$5+t$	$2*C+t$
Code Sheet Generation	3	$2*C$
Voting	2	
Tallying	t	t

1) Number of cryptographic algorithms in use: As shown in Table II, both schemes rely on the cryptographic algorithms *encryption*, *re-encryption*, *decryption*, *permutation*, *zero-knowledge proofs*, *commitments*, and *signatures*. Those are the typically applied cryptographic algorithms if verifiability is provided for (some of) the integrity sub-criteria, while maintaining secrecy. In addition PGD relies on *blinding*.

2) Number of essential process steps: In Table II, the number of applications of cryptographic algorithms, denoted by essential process steps, is summarized. The total amount of essential process steps of PUD is

$$9+2*c+3*t^2+3*t,$$

while for PGD the total amount of essential process steps is

$$4+c+3*t^2+11*t+4*t*c+6*C*c+8*C.$$

Independent of the variable assignment for c, the number of candidates, C the number of Clerks, and the number of Trustees t, PGD has more steps than PUD. In order to satisfy our security model, the minimal number for t and C must be 2. For these assignments the total amount of essential process steps of PUD is $27 + 2 * c$ and of PGD is $54 + 21 * c$.

The results of our analysis show that PUD is more understandable than PGD.

VIII. Conclusion

Existing Internet voting schemes are based on different security models, in particular, different adversary models. However, most of these are not adequate for high-stake elections in established democracies. We therefore developed a security model and justified its adequacy for use in such contexts and environments. We proposed an Internet voting scheme, Pretty Understandable Democracy (PUD), and showed that it meets the proposed security model. Additionally, PUD provides some verifiability as two integrity sub-criteria are provided without posing restrictions on the adversary.

We have also proposed understandability criteria to evaluate the understandability of Internet voting schemes, and applied these criteria to evaluate the understandability of PUD in comparison to Pretty Good Democracy (PGD). PGD is the only other Internet voting scheme that meets the specified security criteria under the adequate adversary model. The evaluation according to the proposed understandability criteria has shown that PUD is more understandable than PGD. PUD is proposed for consideration in future Internet voting projects.

Here, we consider the most important security criteria for Internet voting schemes. In future work, further criteria for Internet voting schemes have to be taken into account, e.g., fairness and dispute-freeness.

In future work, concentrating on simplicity of cryptographic algorithms, metaphors will be designed to aid voter understanding. A user study will be implemented to evaluate PUD and the understandability criteria proposed herein, as well as to extend these criteria if necessary. We plan to apply the understandability criteria to evaluate other Internet voting schemes in the future.

References

[1] Forsa-Umfrage: Jeder zweite würde online wählen. Digitale Technologien stärken die Demokratie. Bürgerbeteiligung über das Internet fördert Vertrauen in die Politik . http://www.microsoft.com/germany/newsroom/pressemitteilung.mspx?id=533684., 2013. Online; accessed 26 February, 2013.

[2] B. Adida. Helios: web-based open-audit voting. In *Proceedings of the 17th conference on Security symposium*, SS'08, pages 335–348, Berkeley, CA, USA, 2008. USENIX Association.

[3] F. Bannister. A Risk Assessment Framework for Electronic Voting. In *European Conference on eGovernment*, pages 43–56, 2005.

[4] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, pages 544–553, New York, NY, USA, 1994. ACM.

[5] R. Casati. Trust, Secrecy and Accuracy in Voting Systems: The Case for Transparency. *Mind and Society: Cognitive Studies in Economics and Social Sciences*, 9(1):19–23, 2010.

[6] D. Chaum. Sure vote: Technical overview. In *Proceedings of the Workshop on Trustworthy Elections (WOTE 01)*, 2001.

[7] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Peudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[8] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a Secure Voting System. In *IEEE Symposium on Security and Privacy*, pages 354–368, 2008.

[9] Council of Europe. Legal, Operational and Technical Standards for E-Voting. Recommendation Rec(2004)11 adopted by the Committee of Ministers of the Council of Europe and explanatory memorandum. Council of europe publishing, 2004.

[10] A. Essex, J. Clark, U. Hengartner, and C. Adams. Eperio: Mitigating technical complexity in cryptographic election verification. *IACR Cryptology ePrint Archive*, 2012:178, 2012.

[11] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20:51–83, 2007.

[12] S. Heiberg, P. Laud, and J. Villemson. The Application of I-voting for Estonian Parliamentary Elections of 2011. In A. Kiyaias and H. Lipmaa, editors, *Postproceedings: 3rd international conference on e-voting and identity*, volume 7187 of *Lecture Notes in Computer Science*, pages 208 – 223. Springer-Verlag, 2012.

[13] J. Helbach. Code Voting mit prüfbaren Code Sheets. In *GI Jahrestagung*, pages 1856–1862, 2009.

[14] J. Helbach and J. Schwenk. Secure Internet Voting with Code Sheets. In *VOTE-ID*, pages 166–177, 2007.

[15] J. Helbach, J. Schwenk, and S. Schäge. Code Voting with Linkable Group Signatures. In *Electronic Voting*, pages 209–208, 2008.

[16] R. Joaquim, P. Ferreira, and C. Ribeiro. EVIV: An End-to-end Verifiable Internet Voting System. *Computers & Security*, 32:170–191, 2013.

[17] R. Joaquim and C. Ribeiro. CodeVoting: Protecting Against Malicious Vote Manipulation at the Voter's PC. In *Frontiers of Electronic Voting*, 2007.

[18] R. Joaquim and C. Ribeiro. CodeVoting Protection Against Automatic Vote Manipulation in an Uncontrolled Environment. In *VOTE-ID*, pages 178–188, 2007.

[19] R. Joaquim, C. Ribeiro, and P. Ferreira. VeryVote: A Voter Verifiable Code Voting System. In *Proceedings of the 2nd International Conference on E-Voting and Identity*, VOTE-ID '09, pages 106–121. Springer-Verlag, 2009.

[20] R. Joaquim, C. Ribeiro, and P. Ferreira. Improving Remote Voting Security with CodeVoting. In *Towards Trustworthy Elections*, pages 310–329, 2010.

[21] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant Electronic Elections. In *ACM Workshop on Privacy in the Electronic Society*, pages 61–70, 2005.

[22] E. Maaten. Towards Remote E-Voting: Estonian case. In A. Prosser and R. Krimmer, editors, *Electronic Voting in Europe*, volume 47, pages 83–100. GI, 2004.

[23] S. Neumann, J. Budurushi, and M. Volkamer. Analysis of Security and Cryptographic Approaches to Provide Secret and Verifiable Electronic Voting. In D. Zissis and D. Lekkas, editors, *Design, Development, and Use of Secure Electronic Voting Systems*. IGI Global, 2013.

[24] S. Neumann and M. Volkamer. Formal Treatment of Distributed Trust in Electronic Voting. In ThinkMind, editor, *Seventh International Conference on Internet Monitoring and Protection*, pages 47–55. IARIA, 2012.

[25] Organization for Security and Co-operation in Europe (OSCE)/ Office for Democratic Institutions and Human Rights (ODIHR). Norway: Internet Voting Pilot Project / Local Government Election - 12 September 2011: OSCE/ODIHR Election Expert Team Report., 2012.

[26] P. Y. A. Ryan and V. Teague. Pretty Good Democracy. In B. Christianson, J. A. Malcolm, V. Matyas, and M. Roe, editors, *Security Protocols Workshop*, pages 111–130. Springer, 2009.

[27] G. Schryen, M. Volkamer, S. Ries, and S. M. Habib. A Formal Approach Towards Measuring Trust in Distributed Systems. In *ACM Symposium on Applied Computing*, pages 1739–1745. ACM, 2011.

[28] M. Volkamer and R. Grimm. Determine the Resilience of Evaluated Internet Voting Systems. In *First International Workshop on Requirements Engineering for e-Voting Systems*, RE-VOTE '09, pages 47–54. IEEE Computer Society, 2009.

[29] M. Volkamer and R. Vogt. Basic set of security requirements for Online Voting Products. Technical Report BSI-PP-0037, 2008. Common Criteria Protection Profile.