Formal Treatment of Distributed Trust in Electronic Voting

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

***Abstract*—Electronic voting systems are among the most security critical distributed systems. Different trust concepts are implemented to mitigate the risk of conspiracies endangering security properties. These concepts render systems often very complex and end users no longer recognize whom they need to trust. Correspondingly, specific trust considerations are necessary to support users. Recently, resilience terms have been proposed in order to express, which entities can violate the addressed security properties in particular by illegal collaborations. However, previous works derived these resilience terms manually. Thus, successful attacks can be missed. Based on this approach, we propose a framework to formally and automatically derive these terms. Our framework comprises a knowledge calculus, which allows us to model knowledge and reason about knowledge of collaborating election entities. The introduced framework is applied to deduce previously manually derived resilience terms of three remote electronic voting systems, namely Polyas, Helios and the Estonian voting system. Thereby, we were able to discover mistakes in previous derivations.**

***Keywords*-trust, distributed systems, formal methods, resilience, electronic voting, knowledge calculus, conspiracies**

I. Introduction

Recently, the interest in electronic voting systems increases as more and more states implement electronic voting, both with voting machines as well as remote internet voting schemes. In this paper, we only consider internet voting schemes and use the term electronic voting or eVoting interchangeably. Electronic voting schemes are complex distributed systems with particularly strict security requirements due to the nature of elections. It is therefore of great importance to evaluate these schemes prior to their use in legally binding elections.

Numerous analysis and verification techniques for electronic voting have been proposed over the past decades. The Common Criteria, in particular the Protection Profile for electronic voting [1], is an international standard for security evaluation, which has been successfully applied to electronic voting schemes, see [2] for an example. Additionally, many researchers evaluated proposed voting schemes, both with formal methods as in [3], [4], [5] and by cryptographic means as in [6], [7]. However, these techniques mainly investigate external attacks and do not address illegal collaborations between different entities.

However, specific trust considerations are necessary because the implemented trust concepts result in very complex systems and voters are faced with the problem whom to trust not to illegally collaborate with other entities. In [8] Volkamer et al. propose resilience terms to derive which entities need to be in particular trusted not to collaborate maliciously in order to ensure security properties. Thus, resilience terms express how robust a system is against conspiracies of entities that do not behave properly. Resilience terms have shown their benefit in the evaluation of different systems, while their derivation remains informal and potential conspiracies can be missed or misinterpreted, which leads to wrong resilience terms.

In this paper, we review resilience terms and introduce a logic, which allows us to formally and automatically derive resilience terms in electronic voting schemes. While there are many security properties that need to be satisfied by voting schemes, we focus on secrecy as property of crucial importance to most voting schemes. Our idea is based on formal methods and knowledge management. We establish distributed knowledge bases of entities in logical terms and propose a inference system, which incorporates the deduction rules to extend the obtained knowledge by the adversary. We apply our framework for three electronic voting schemes, namely Polyas, Helios and the Estonian one; compare the results with those from [8]; and thereby detect mistakes in the previously established resilience terms.

The remainder of this paper is organized as follows: In Section II, we review related work in the formal analysis of electronic voting schemes. In Section III we review the existing framework on resilience terms as proposed in [8]. In Section IV, we introduce our knowledge calculus, while Section V is dedicated to the formalization of the secrecy property. Thereafter, in Section VI we exploit the proposed knowledge calculus in order to derive resilience terms. In Section VII, we deduce the resilience term for the three electronic voting schemes based on our proposal. Section VIII concludes this paper and shows future directions.

II. Related Work

The formal security evaluation of electronic voting schemes has been approached by different means. In this section, we review related literature and check whether these

 ISBN: 978-1-61208-201-1

works can be adapted to our own needs. As this paper addresses only internet voting schemes, we do not take into account the verification of voting machines as proposed for instance in [9].

Several works build upon the applied pi calculus [10] and its formalization in ProVerif: Backes et al. [3] prove coercion-resistance of the JCJ protocol. Kremer et al. [4] prove fairness and eligibility of the FOO 92 [5] protocol and Cortier et al. [11] adapt the Helios system and prove secrecy of the system. These techniques assume (or encode this in the properties specification) a fixed scenario, where a given set of entities is assumed to be trusted not to collaborate maliciously. The goal of these works is to identify scenarios in which a security property can be verified rather than determining the smallest set of entities that might assure that property. Therefore, a derivation of resilience terms with respect to any illegal conspiracies can not be handled by these techniques without further adaptation.

Jonker et al. [12] propose a formalization of receipt-freeness in a state-orientated manner, which allows them to verify receipt-freeness for different electronic voting schemes. Their formalization is strongly generic, i.e., term structures underlying the schemes need to be generated independently for protocols. In [13] and [14] Bräunlich et al. follow the idea of Jonker and use the state-orientated model to identify state transitions not violating state invariants. Their main purpose is to support engineers in the design of protocols in a way that state invariants hold. Due to the abstract nature of their approach, it is currently not applicable for voting scheme evaluations.

Finally, we will review whether approaches proposed in the context of formal trust management can be adapted to our needs. Trust concepts and management systems have been approached from different directions; in [15] and [16], the authors rely on game-theoretic approaches to evaluate the cost-benefit relation for different entities to collaborate. We see the value of these approaches mainly in the evaluation of resilience terms. Once resilience terms have been determined, the risk or chance of entities ensuring or violating security properties can be estimated. We refer to [17] and [18] for a comprehensive overview on trust concepts.

III. Existing Framework

Electronic voting systems process sensitive data, so different trust concepts were proposed in order to mitigate the risk of conspiracies endangering security properties such as integrity or secrecy. There are mainly three trust concepts applied to electronic voting, namely separation of duty, the four eyes principle and the multiple execution of a duty. These trust concepts are usually combined and applied multiple times. This leads to complex trust distributions where the question whom to trust not to collaborate maliciously regarding certain security properties can become hard to answer.

The evaluation of distributed systems with resilience terms has been introduced in [19] and adapted to electronic voting in [8]. Resilience terms allow one to identify which entities a voter has to trust - in particular not to collaborate maliciously - in order not to violate an investigated security property. Terms however need to be determined independently for different properties. Entities can be voting servers, administration staff, developers and key holders. The framework assumes voters to behave properly and, thus, voters are not considered as entities in this framework[1]. Resilience terms can be derived on different levels, where level 1 corresponds to servers or key holders, level 2 to the local position of components and administration staff of servers correspondingly, and level 3 to the manufacturer of the voting software run on the servers. Terms of higher levels can be (formally) derived based on lower level terms in a straight-forward manner.

The framework of resilience terms has been applied to derive resilience terms for different voting systems, namely the Estonian voting system, the Polyas [20] and the Helios 2.0 [21] system. The framework has been extended in [22] towards post-processing of these terms by means of transformation into logical terms and the evaluation of terms with respect to trust metrics. It allows one to determine the probability with which an electronic voting system fulfills a security property. Figure 1(a) shows how resilience terms can be evaluated with respect to trust metrics. In order to determine the resilience term, in [8] the authors propose to manually and informally determine the knowledge of entities once the voting phase has terminated. These local knowledge sets of entities are thereby determined from a *worst-case* point of view, i.e., entities are assumed not to behave properly in the sense that they store all terms they obtain and furthermore they store all visible relations between terms. Based on the obtained local knowledge sets, attack scenarios are identified and the resilience terms are derived correspondingly. Resilience terms have the form

$$t = (k_1 + \cdots + k_m) \text{ out of } (N_1, \ldots, N_m)$$

where $N_1, \ldots, N_m$ is the list of identified entities that are able to violate a security property if k_1 entities out of N_1 and $\ldots$ and k_m entities out of N_m collaborate maliciously. If different terms allow the violation of a security property, then these terms $t_1, \ldots, t_m$ are separated by the ";" symbol, that is

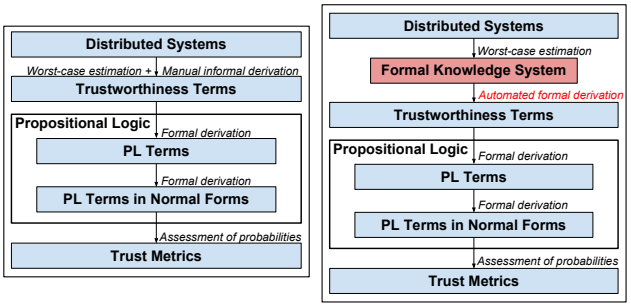
$$t_1; \ldots; t_m.$$

$t_1; t_2$ can be reduced to t_1 if $t_1 = i$ out of N and $t_2 = j$ out of M with $i < j$ and $N \subset M$ holds.

The informal derivation of these terms can miss or misinterpret attack scenarios, which leads to wrong resilience terms. Therefore, in this work, we adapt the framework of

[1]Correspondingly, the critics in [11] is not justified.

 ISBN: 978-1-61208-201-1



(a) Original Research Framework. (b) Extended Research Framework.

Figure 1. Research Framework.

[8] and propose the formal derivation of resilience terms based on formal knowledge representations. The highlighted parts of Figure 1(b) integrate our extension into the existing framework. Due to space constraints, in this paper, we focus on level 1 while we recall that higher levels can be (formally) derived in a straight-forward manner.

IV. Knowledge Calculus

In Section II, we reviewed literature in the context of formal analysis of electronic voting and argued that none of these approaches can be extended for our purpose. In this section, we therefore commit on basics to underlie our ideas and motivate the used methodology. We propose to apply the concept of knowledge representation and reasoning about knowledge as it is a well established concept of artificial intelligence and has been influenced and improved by formal methods. As Dolev-Yao (DY) [23] adversary models have been successfully used to analyze cryptographic protocols also in the context of electronic voting, we integrate a DY adversary model in our approach.

In this section, we first introduce the knowledge algebra to represent terms, which can be known by entities. Thereafter, the knowledge system is introduced in terms of a state transition system, which allows the adversary to extend his knowledge in terms of corrupting election entities. Finally, reasoning over knowledge is realized by the adversarial deduction rules used to extend adversarial knowledge. Due to space constraints, we restrict our attention to entities and inference rules, which will be used in the following examples rather than a more comprehensive specification.

Correspondingly, we do not settle our work in protocol analysis but rather see the contribution in trust and knowledge management in the field of electronic voting schemes.

A. Knowledge Algebra

In this section, we introduce the algebra composed by terms and equations making the semantics of terms.

1) Term Signature: We define the term signature to be

$$Sig = \left(\bigcup_{i \in \mathbb{N}} F^i \right) \cup R$$

where F^i represents the function symbols of arity i and R implements the relation between terms. The signature is later on used to represent known messages and to formalize security properties.

We define a subtype Ent, which embodies entities carrying out an election, namely voters [2] and election services as well as election authorities, such as key holders. We present the entities in extracts while a more detailed consideration depends on the voting scheme under investigation.

$$\begin{aligned} Ent \quad = \quad & \{voter(i) \mid i \in \mathbb{N}\} \cup \quad \text{(*Voters*)} \\ & \{KH(i) \mid i \in \mathbb{N}\} \cup \quad \text{(*Key holders*)} \\ & \{BBS\} \cup \quad \text{(*Ballot box server*)} \\ & \ldots \end{aligned}$$

We refer to $role_{type}$ as a set of entities of a certain type, e.g., $role_{KH} = \bigcup_{i \in \mathbb{N}} KH(i)$. The set of voters $role_{voter}$ is abbreviated by V. The complete function symbols are specified by the following signature:

$$\begin{aligned} F^0 &= Ent \cup \{vote, sk, pk, k, tan, token\} \\ F^1 &= \{hash, ss\} \\ F^2 &= \{sig, a{-}enc, enc\} \\ F^4 &= \{share\} \end{aligned}$$

Apart from entities, the signature provides symbols for votes, secret keys, public keys, symmetric keys, transaction authentication numbers (TAN), tokens, hash values, symmetric and asymmetric encryption. We will provide function $a{-}enc$ with explicit randomness whenever this is of importance to the protocol specification. The function ss denotes the secret sharing of a term into different shares. Each share contains information about the shared term, the index of the share, as well as information about how many shares need to be collected in order to reconstruct the shared term and how many shares exist. Below, we often use asymmetric key pairs where entities sometimes hold different key pairs for different use, such as encryption, signature or database key pairs. By sk_{ent}^{type} we denote the private key of entity ent to be used for $type$.

2) Equations: The semantics of function symbols are given by the following equations, which we abbreviate by E.

$$R(t_1, t_2) = R(t_2, t_1) \qquad (1)$$

$$ss(\overbrace{share(t, i, k, n), \ldots, share(t, j, k, n)}^{k \text{ times}}) = t \qquad (2)$$

Equation 1 indicates the commutativity of the knowledge relation. Following the four eyes trust principle, electronic

[2] Note that voters need to be considered in order to specify the secrecy property while they still remain incorruptible

 ISBN: 978-1-61208-201-1

voting schemes often distribute secrets among independent entities in order to mitigate the risk of small conspiracies violating security properties. Equation 2 prescribes how a distributed secret can be reconstructed using the secret shares. It holds $\forall share(t,a,k,n), share(t,b,k,n),$ with $i \leq a,b \leq j : share(t,a,k,n) \neq share(t,b,k,n)$. By $t_i^{k,n}$ we denote $share(t,i,k,n)$.

Signature Sig and the equation set E lead our electronic voting theory, which underlies the remainder of this paper.

B. Knowledge System

We model the knowledge system as state transition system where transitions between states model corruption of entities. We define $Knowledge$ to be a set of ground terms $T(Sig)$, which embodies the local knowledge of an entity. Global knowledge is defined as composition of the entities' local knowledge bases.

$$GlobalKnowledge \quad ::= \quad Knowledge^*$$

Accordingly, the intruder knowledge refers to type $Knowledge$:

$$IntruderKnowledge \quad ::= \quad Knowledge$$

1) State and Traces: A state is given by an execution trace, the global knowledge of entities as well as the intruder knowledge.

$$State \quad ::= \quad Trace \times GlobalKnowledge \times IntruderKnowledge$$

Collaboration is collectively embodied in the adversary, i.e., the corrupted participants' knowledge sets pass into adversarial ownership. The execution of our corruption model is carried out based on the initial distribution of knowledge. At this point, we only consider one adversarial event, the corruption of participants, which releases their knowledge to the adversary.

$$Event \quad ::= \quad corrupt(id)$$

Traces are composed inductively by sequences of events:

$$Trace \quad ::= \quad Event.Trace$$

Initial State: The initial state of a knowledge system with respect to electronic voting schemes is defined as:

$$s_0 \quad = \quad \epsilon \times K_{init} \times IK_{init}$$

Initially, no identity is corrupt, hence the execution trace is empty. The local knowledge is given by the scheme analysis and is generally abbreviated by K_{init}. After the successful completion of the voting phase, the adversary's knowledge, generally referred to as IK_{init}, is defined by the network model and infrastructural details. The initial intruder knowledge might consist of terms, which are publicly known or which are given by the curious behavior of the adversary, hence the interception of public channels and public bulletin boards.

2) Execution Model: Given a state defined by an event trace tr, the local knowledge states of entities collectively encoded in K and the adversary knowledge given by a set of terms IK, the adversary may issue a corrupt event targeted at entity ID. The execution of this event by the system results in a state s_{i+1} where tr is extended by the recent corrupt event, the local knowledge of entities remains unchanged and the adversary's knowledge set is extended by the local knowledge of the corrupted entity.

$$\frac{s_i = \langle tr, K, IK \rangle \qquad ev = corrupt(ID)}{s_{i+1} = \langle ev.tr, K, IK \cup K(ID) \rangle}$$

C. Adversary Deduction System

Based on the adversarial knowledge resulting from corruption of entities, we introduce a deduction system that allows the adversary to extend gained knowledge in a logical sense, hence based on acquired terms, the adversary is allowed to extend its knowledge according to an inference system, which is given by the rules below. As we only consider secrecy properties, the attacker can only decompose terms rather than synthesize them.

1) Basic Rules: Knowledge is given by means of sets over terms, hence elements of knowledge sets are derivable according to the inference system.

$$\frac{m \in IK}{IK \vdash_E m}$$

2) Asymmetric Encryption Rules: The rule enables the adversary to decrypt publicly encrypted messages if he holds the corresponding private key.

$$\frac{IK \vdash_E sk_i^{enc} \qquad IK \vdash_E a{-}enc(pk_i^{enc}, m)}{IK \vdash_E m}$$

3) Symmetric Encryption Rules: The adversary can derive a message if he holds the encryption of that message and the corresponding symmetric encryption key.

$$\frac{IK \vdash_E k \qquad IK \vdash_E enc(k,m)}{IK \vdash_E m}$$

4) Hash Rules: The adversary is allowed to derive hash values of messages he holds.

$$\frac{IK \vdash_E m}{IK \vdash_E hash(m)}$$

5) Signature Rules: Signatures reveal the relation between signer and the signed message.

$$\frac{IK \vdash_E sig(sk_i, m)}{IK \vdash R(i,m)}$$

 ISBN: 978-1-61208-201-1

6) Secret Sharing Rules: We allow the adversary to use the ss operator in order to reconstruct distributed terms.

$$\frac{IK \vdash_E t_1 \quad \ldots \quad IK \vdash_E t_n}{IK \vdash_E ss(t_1, \ldots, t_n)}$$

7) Relational Rules: Local knowledge sets of entities are considered to be faithful, i.e., the union of these sets never allows for inconsistencies. The rules given below specify the projection on relations and the transitivity of the knowledge relation.

$$\frac{IK \vdash_E R(a,b)}{IK \vdash_E a} \qquad \frac{IK \vdash_E R(a,b)}{IK \vdash_E b}$$

$$\frac{IK \vdash_E R(t,x) \quad IK \vdash_E R(t,y)}{IK \vdash_E R(x,y)}$$

V. Secrecy

The term secrecy often refers to different details while the underlying idea remains mainly the same with respect to electronic voting. In contrast to classical cryptographic protocols, secrecy properties of electronic voting schemes do not require the secrecy of terms, as the public availability of votes is central to the public nature of elections. In fact, secrecy in electronic voting resembles the idea of anonymity in cryptographic protocols, i.e., an adversary should not be able to link voters and their votes. Therefore, constructing relations between terms may enable the adversary to violate secrecy properties and is therefore of central importance in our approach.

The *intruder deduction problem* for secrecy denotes the problem to deduce a term t from a set of terms IK based on a given inference system. In general, this fact is abbreviated by

$$IK \vdash_{IS} t$$

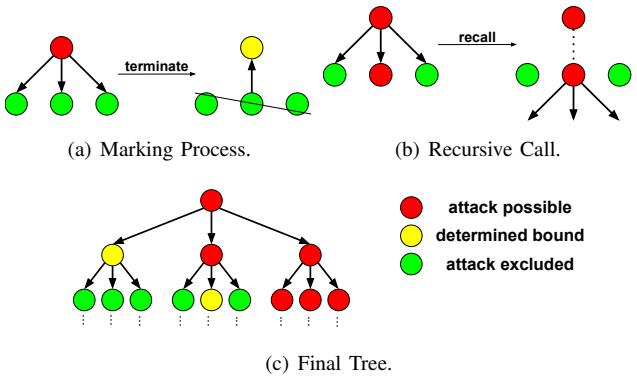
where IS denotes the corresponding inference system. Let a state $s = \langle tr, K_{init}, IK \rangle$ be given. The intruder deduction problem for secrecy in electronic voting refers to

$$IK_s \vdash R(voter(i), vote(j))$$

for some $i, j \in \mathbb{N}$. Therefore, assumptions about investigated states have to be made, which subsequentally allows for resilience term derivation. By logical means this can be expressed as follows: For a state $s = \langle tr, K_{init}, IK \rangle$, one needs to find $min|bound(role_r)|$ for all $role_r$ where $\left(\bigwedge_{i \in role_r}^{bound(role_r)} corrupt(role_r(i)) \notin tr\right)$ such that $IK_s \nvdash R(voter(i), vote(j))$.

VI. Determination of Resilience Terms

The ultimate goal of our approach is to automatically derive the minimal sets of entities that need to be trusted in order to ensure the security properties of interest. Before diving into details of the algorithm, we emphasize that the algorithm assumes a worst-case estimation about the obtained knowledge of local entities. Hence, we manually consider the entire protocol and formalize relational knowledge and knowledge of terms that entities might gather if they do not behave properly. Once, this estimation is available, the following recursive algorithm is executed:



(a) Marking Process. (b) Recursive Call.

(c) Final Tree.

Figure 2. The proposed algorithm.

Assume a scenario with n entities. In such a situation, the prover is initially called with a collaboration of all n entities in order to determine if secrecy is violated. If so, n instances of the prover are called each analyzing a different collaboration scenario, i.e., each call excludes one entity from the collaboration. Once, a collaboration can not violate secrecy, we cut the tree at this point and give the result back. In case no child of a node can violate secrecy, then this node is marked and will be used for the final computation as depicted in Figure 2(a). In case some children of a node can violate the property, while other children can not, the algorithm is recursively called for the violating children as shown in Figure 2(b). The final output of this algorithm corresponds to a tree of the form given in Figure 2(c).

Of crucial importance to our proposal is the handling of entities in identical roles, e.g., key holders. In order to reduce the computational complexity, the algorithm therefore is designed in the following way: Think of a 3 out of 6 threshold scheme for distributed decryption among $\{KH_1, \ldots, KH_6\}$. We therefore invent a new super entity $KH_{3,6}$, which correspond to the reconstructed decryption key. This entity is used throughout the algorithmic proceeding. In the final k-resilience value, this entity is than resubstituted by 3 out of $\{KH_1, \ldots, KH_6\}$. Finally, marked notes represent the determined resilience term.

In worst-case the number of prover calls is

$$\#calls(prover) = \sum_{i=1}^{n} \binom{n}{i}.$$

Note that we follow a top-down approach, although it can easily be adapted to a bottom-up proceeding.

 ISBN: 978-1-61208-201-1

VII. Evaluation of eVoting Schemes

We deploy the calculus in order to evaluate different electronic voting schemes by means of resilience terms. We briefly present the voting schemes that have been investigated also in [8], namely Polyas, Helios and the Estonian voting system. Thereafter, we deduce the obtained local knowledge sets from a worst-case, which allows us to automatically derive the corresponding resilience terms and contrast these terms with the previously informally derived terms.

A. Polyas

Polyas is a remote voting system developed by Micromata in 1996, with which many elections have been carried out. Polyas comprises the following components:

Printing Service (PS): The PS prints the election material and sends this material to the voters via postal mail.

Election Registration Server (ERS): The ERS implements the electoral roll, which is accessed to verify the eligibility of the voter.

Validation Server (VS): Similar to the ERS, the VS re-verifies the eligibility of the authenticating voter such that both the ERS and the VS control each other.

Ballot Box Server (BBS): The BBS stores the encrypted votes cast by eligible voters.

Tallying Component (TC): The TC is an offline component, which tallies the stored votes in the BBS after the election has terminated.

Key holders: There are two independent election officials (KH_1, KH_2), which hold the private key shares corresponding to the public key used to encrypt votes.

Figure 3 depicts the protocol interaction in form of a sequence diagram. For further information about the protocol specification, we refer to [24], [8], [20]. Note that the communication between different entities is secured by https connections. Once, the voting phase has terminated, the content of the BBS is carried over to the TC, which is offline, where the votes are collectively decrypted by the collaboration of both key holders.

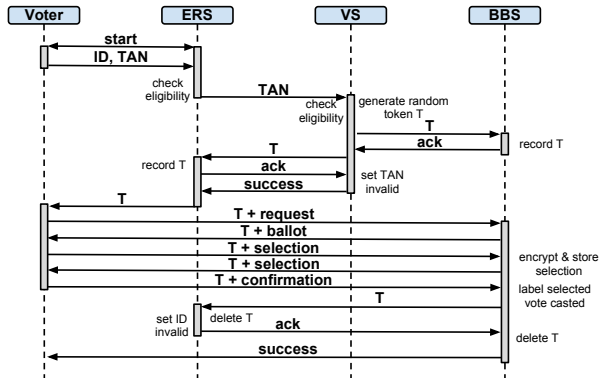


Figure 3. Polyas Voting Scheme.

The entities' knowledge can be formalized in the following way:

Election Registration Server: The hash values of TANs prepared for eligible voters are available to the ERS.

- $\forall i \leq |V|$ s.t. $R(voter(i), tan(j)) : hash(tan(j)) \in K(ERS)$

All voters' IDs are available to the ERS.

- $\forall i \leq |V| : voter(i) \in K(ERS)$

The ERS is aware of the voter-TAN relation.

- $\forall i \leq |V| : R(voter(i), tan(j)) \in K(ERS)$

The ERS knows the relation between TAN and tokens generated by the VS.

- $\forall i \leq |V|$ s.t. $R(voter(i), tan(j)) : R(tan(j), token(k)) \in K(ERS)$

Printing Service: The PS receives the voting material and distributes eligible TANs among the voters via postal mail, hence the service knows:

- $\forall i \leq |V| : R(voter(i), tan(j)) \in K(PS)$

Validation Server: TANs prepared for eligible voters are available to the VS.

- $\forall i \leq |V|$, s.t. $R(voter(i), tan(j)) : tan(j) \in K(VS)$

Furthermore is the relation between prepared TANs and prepared tokens known to the VS, as the VS generates these tokens.

- $\forall i \leq |V|$ s.t. $R(voter(i), tan(j)) : R(tan(j), token(k)) \in K(VS)$

Ballot Box Server: The tokens prepared for eligible voters are available to the BBS.

- $\forall i \leq |V|$ s.t. $R(voter(i), tan(j)), R(tan(j), token(k)) : token(k) \in K(BBS)$

We recall that the local knowledge sets are determined by a worst-case estimation about the complete protocol run. Hence, the BBS might store information about votes together with the respective token used to cast this vote.

- $\forall i \leq |V|$, s.t. $R(voter(i), token(k)) : R(token(k), vote(l)) \in K(BBS)$

The BBS might store the encrypted version of the cast votes, together with the corresponding tokens used to submit these votes.

- $\forall i \leq |V|$, s.t. $R(voter(i), token(k)) : R(token(k), a{-}enc(pk_{DB}, vote(l))) \in K(BBS)$

Tallying Component: The TC obtains the knowledge from the BBS. The TC furthermore stores the encrypted version of each cast vote together with the vote.

- $\forall i \leq |V|$, s.t. $R(voter(i), token(k)), R(token(k), a{-}enc(pk_{DB}, vote(l))) : R(enc(pk_{DB}, vote(l)), vote(l)) \in K(TC)$

Key holders: The private database key sk_{DB} is shared among two independent key holders.

- $\forall i \in \{1, 2\} : sk^{2,2}_{DB_i} \in K(KH_i)$, s.t. $sk^{2,2}_{DB_1} \neq sk^{2,2}_{DB_2}$

 ISBN: 978-1-61208-201-1

Resilience Term Derivation: The first boundary our algorithm detects leads to a state $s = \langle tr, K_{init}, K(BBS) \cup K(ERS) \cup \ldots \rangle$, hence the corruption of the ERS and the BBS. In s, the relational axioms allow the adversary to reason in the following way:

$$\frac{\begin{array}{c} IK_s^{ERS} \vdash R(voter(i), tan(j)) \\ IK_s^{ERS} \vdash R(tan(j), token(k)) \end{array}}{\begin{array}{c} IK_s^{ERS} \vdash R(voter(i), token(k)) \\ IK_s^{BBS} \vdash R(token(k), vote(l)) \end{array}}{IK_s \vdash R(voter(i), vote(l))}$$

Note, that our algorithm does not investigate any further conspiracies where ERS and BBS are involved as such conspiracies automatically also violate the secrecy property. Trace tr with $corrupt(PS)$, $corrupt(VS)$, $corrupt(BBS) \in tr$ results in s, allowing the adversary to reason in the following way:

$$\frac{\begin{array}{c} IK_s^{PS} \vdash R(voter(i), tan(j)) \\ IK_s^{VS} \vdash R(tan(j), token(k)) \end{array}}{\begin{array}{c} IK_s \vdash R(voter(i), token(k)) \\ IK_s^{BBS} \vdash R(token(k), vote(l)) \end{array}}{IK_s \vdash R(voter(i), vote(l))}$$

Hence, conspiracies between the PS, the VS and the BBS allow the adversary to violate the secrecy property.

Finally, the algorithm terminates and returns the following secrecy resilience term for Polyas:

$$\begin{array}{rcl} t & = & 2 \text{ out of } \{ERS, BBS\}; \\ & & 3 \text{ out of } \{PS, VS, BBS\} \end{array}$$

The informal resilience term derivation in [8] resulted in the fact that BBS can guarantee the secrecy of the vote. This coincides with our result. Furthermore, their result states that also ERS and VS together can ensure secrecy. As opposed to our consideration, they did not take into account the printing service, therefore our second attack is out of scope for their scenario. Hence, the entities ERS and VS should not be part of their resilience term for the sake of consistency.

B. Helios

The Helios voting system has been introduced in [21] by Ben Adida, while currently Helios version 3.1 is available. In contrast to prior and later versions, Helios 2.0 is based on homomorphic tallying [25]. This work is based on Helios 2.0 as this version has already been investigated manually and a resilience term has been determined [8]. The protocol is based on the idea of separating ballot preparation/encryption and authentication. Helios comprises the following components:

Election Builder (EB): The EB initially determines candidates and eligible voters and provides eligible voters with their login data and the URL.
Voting JavaScript: The script allows the voter to process his vote and to interact with the backend of the system. The JavaScript is launched by the Helios website. The randomness used to encrypt the voter's choice is stored within this script. We assume the script to behave properly and therefore do not distinguish between voter and his script.
Ballot Verifiers (BV): There are three BV, which can be involved by voters to audit the encryption process.
Authentication Server (AS): The AS allows the voter to authenticate himself in order to submit his encrypted vote.
Bulletin Board (BB): The BB is a public channel on which voters may verify if their cast votes are stored.
Tallying Component (TC): According to the multiple execution of a duty principle, there are two independent offline tallying components TC_1 and TC_2, which tally the stored votes on the BB once the election has been finished.
Key holders (KH): There are six independent election officials that hold private key shares in order to decrypt stored ballots.

An overview over the Helios system is given in Figure 4. At the beginning of the election, the Helios election builder

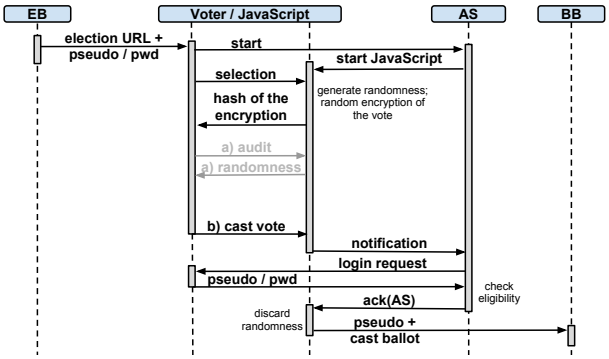


Figure 4. Helios Voting Scheme.

sends the voter an invitation e-mail containing a link to the election website together with his ephemeral login data. At the end of the election process, the BB contains all voters' pseudonyms together with the encrypted vote and the hash value of the encrypted votes such that voters may verify the process. Once, the election has terminated, the encryptions are homomorphically summed up and decrypted by at least three out of six key holders.

The local knowledge sets of the entities are given by:
Election Builder: The EB stores the association between voter and pseudonym for each voter.

- $\forall i \leq |V| : R(voter(i), pseudo(j)) \in K(EB)$

Voting JavaScript: The voting java script of voter i stores the association between the voter's pseudonym and his vote.

 ISBN: 978-1-61208-201-1

- $R(voter(i), pseudo(j))$:
  $R(pseudo(j), vote(l)) \in K(VJS_i)$

Furthermore, the encryption of this vote together with the used randomness is stored.

- $R(voter(i), pseudo(j))$:
  $R(pseudo(j), a{-}enc(vote(l), pk_{TC}, r)) \in K(VJS_i)$
- $R(voter(i), pseudo(j))$,
  $R(pseudo(j)$,
  $a{-}enc(vote(l), pk_{TC}, r)) \in K(VJS_i)$:
  $r \in K(VJS_i)$

Ballot Verifiers: We omit the consideration of ballot verifiers in the reasoning as the auditing of these servers causes a new voting and encryption step. These verifiers therefore do not influence the resilience term.
Authentication Server: The AS obtains the pseudonyms of eligible voters.

- $R(voter(i), pseudo(j)) : pseudo(j) \in K(AS)$

Bulletin Board: The BB stores and publishes the encrypted version of the cast votes, together with the corresponding pseudonym used to submit these votes.

- $\forall i \leq |V|$ s.t. $R(voter(i), pseudo(j))$:
  $R(pseudo(j), a{-}enc(pk_{TC}, vote(l), r)) \in K(BB)$

The BB stores and publishes the hash value of the encrypted votes, together with the corresponding pseudonym used to submit these votes.

- $\forall i \leq |V|$ s.t. $R(voter(i), pseudo(j))$:
  $R(pseudo(j)$,
  $hash(a{-}enc(pk_{TC}, vote(l), r)) \in K(BB)$

Tallying Component: The TC obtains the knowledge from the bulletin board.
Key holders: The private database key sk_{TC} is shared among six independent key holders in the following way:

- $\forall i \in \{1, 2\} : sk^{3,3}_{TC\_1} \in K(KH_i)$
- $\forall i \in \{3, 4\} : sk^{3,3}_{TC\_2} \in K(KH_i)$
- $\forall i \in \{5, 6\} : sk^{3,3}_{TC\_3} \in K(KH_i)$

We denote the groups of key holders that hold identical key shares by KH^1, KH^2, KH^3. We emphasize that the content of the bulletin board is public, which means that the attacker after the tallying is aware of relation $R(pseudo(j), a{-}enc(pk_{TC}, vote(l), r))$.

Resilience Term Derivation: The algorithm detects that in case one key holder of each key holder group is corrupt, the key sk_{TC} can be reconstructed. If additionally the EB is compromised this leads to state s, which allows the following reasoning:

$$\frac{\dfrac{IK_s^{KH^1} \vdash sk^{3,3}_{TC\_1} \quad \dfrac{IK_s^{KH^2} \vdash sk^{3,3}_{TC\_2} \qquad IK_s^{KH^3} \vdash sk^{3,3}_{TC\_3}}{IK_s \vdash sk_{TC}} \quad IK_s \vdash R(pseudo(l), a{-}enc(pk_{TC}, vote(m), r))}{IK_s \vdash R(pseudo(l), vote(m))} \quad IK_s^{EB} \vdash R(voter(o), pseudo(l))}{IK_s \vdash R(voter(o), vote(m))}$$

Finally, the algorithm terminates and returns the following secrecy resilience term for the Helios scheme:

$$\begin{aligned} t \;=\; & (1+1+1+1) \text{ out of} \\ & (\{EB\}, \{KH_1, KH_2\}, \\ & \{KH_3, KH_4\}, \{KH_5, KH_6\}) \end{aligned}$$

Level 1 resilience term in [8] expresses that the authentication server and one key holder of each group needs to be trusted, while the authentication server in their consideration plays the role of our EB. This observation coincides with our result.

C. Estonian Voting System

In 2005, Estonia was the first country in which electronic elections were legally binding for the municipal elections. Their system relies on the Estonian ID card, which is both the regular ID card and a smart card capable of pursuing legally binding digital signatures. The Estonian electronic voting system comprises the following components:
Voter Application (VA): Each voter runs a VA on which he selects his preferred candidates. After this, the application encrypts the vote by the election key and signs the ballot with the voter's private key stored on his national ID card.
Vote Forwarding Server (VFS): The VFS is directly accessible over the internet and once the voter prepared his signed ballot, this ballot is sent to the VFS, which then forwards the ballot to the vote storage server.
Vote Storage Server (VSS): The VSS receives ballots from the VFS. After the election, the VSS eliminates double votes and votes from ineligible voters. It then removes all signatures and stores the unsigned ballots on a CD.
Vote Counting Application (VCA): The VCA reads the encrypted ballots from the provided CD upon which the key holders collectively start the tallying process.
Key holders (KH): There are seven key holders among which four need to collaborate in order to decrypt cast votes.

A simplified overview of the Estonian internet voting system is given in Figure 5. Apart from the description presented here, the Estonian system allows vote updating, which is omitted in our consideration due to space constraints.

The electronic voting system deployed in Estonia implements the Estonian postal voting by electronic means. Once, the election has terminated, the stored encrypted votes in VSS are burned on a CD and carried over to the VCA.

 ISBN: 978-1-61208-201-1

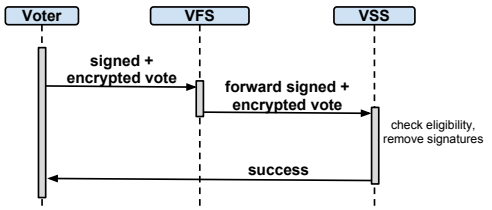


Figure 5. Estonian Voting Scheme.

There, the encrypted votes are mixed and decrypted by a collaboration between four out of seven key holders.

Vote Forwarding Server: The VFS receives signed encrypted votes from the voters.

- $\forall i \leq |V| : voter(i) \in K(VFS)$
- $\forall i \leq |V|$ s.t. $R(voter(i), vote(j)) :$ $sig(sk_i^{sig}, a{-}enc(pk_{VCA}, vote(j))) \in K(VFS)$

Vote Storage Server: The VSS stores the signed encrypted votes from the VFS.

- $\forall i \leq |V| : voter(i) \in K(VSS)$
- $\forall i \leq |V|$ s.t. $R(voter(i), vote(j)) :$ $sig(sk_i^{sig}, a{-}enc(pk_{VCA}, vote(j))) \in K(VSS)$

Vote Counting Application: The VCA only receives encrypted votes.

- $\forall i \leq |V|$ s.t. $R(voter(i), vote(j)) :$ $a{-}enc(pk_{VCA}, vote(j)) \in K(VCA)$

The VCA stores the relation between encrypted votes and votes.

- $\forall i \leq |V|$ s.t. $R(voter(i), vote(j)) :$ $R(a{-}enc(pk_{VCA}, vote(j)), vote(j)) \in K(VCA)$

Key holders: The Estonian Voting System implements a distributed threshold scheme in order to reconstruct the secret key of the VCA.

- $\forall i \in \{1, \ldots, 7\} : sk_{VCA_i}^{4,7} \in K(VCA_i)$ s.t. $sk_{VCA_k}^{4,7} \neq sk_{VCA_l}^{4,7}$

We emphasize that the tallying of ballots in the VCA is public, once the key holders provided their keys, which means that the attacker after the tallying is aware of relation $R(a{-}enc(pk_{VCA}, vote(j)), vote(j))$.

Resilience Term Derivation: The first boundary our proposed algorithm returns is the corruption of the VSS and the VCA as this allows the following reasoning:

$$\frac{\begin{array}{c} IK^{VSS} \vdash sig(sk_i^{sig}, a{-}enc(pk_{VCA}, vote(j))) \\ \hline IK^{VSS} \vdash R(voter(i), a{-}enc(pk_{VCA}, vote(j))) \end{array} \quad IK^{VCA} \vdash R(enc(pk_{VCA}, vote(j)), vote(j))}{IK \vdash R(voter(i), vote(j))}$$

Finally, the algorithm terminates and returns the following secrecy resilience term for the Estonian internet voting system:

$$\begin{array}{rcl} t & = & (1+1) \text{ out of } \{VFS, VSS\}, \{VCA\}; \\ & & (1+4) \text{ out of } (\{VFS, VSS\}, \{KH_1, \ldots, KH_7\}) \end{array}$$

The informal derivation of the resilience term in [8] led to the fact that VSS can guarantee the secrecy of the vote. Our result however shows that also VCA together with 4 out of T key holders can ensure secrecy. This possibility has not been discovered in [8].

VIII. Conclusion and Future Work

This paper takes up the resilience terms proposed by Volkamer et al. [8] used to evaluate distributed systems. Based on this approach we developed a knowledge calculus upon a theory adapted to most general electronic voting schemes. This calculus allows for formal reasoning over our proposed theory. On the basis of this calculus, we formalized the secrecy property and described how to determine resilience terms in this framework. Based on a worst-case knowledge estimation, we iteratively investigate collaboration scenarios and thereby deduce the resilience term. We finally applied our proposal to three electronic voting schemes and came up with mistakes in previously informally derived terms of two of these three schemes.

In future work, we plan to incorporate our theory into SPASS [26], an automated theorem prover, in order to fully automatize the deduction, which allows us to run performance tests on our proposal. The defined theory therefore has to be very precise, such that attacks or the absence of attacks may be decided and proven in an automated way. In protocol analysis, the secrecy property is undecidable in the general case. In this work, we consider completely passive adversaries, for which it has recently been shown that deciding knowledge in security protocols can be done in polynomial time under some e-voting theories [27]. We therefore plan to compare these theories with our own theories in order to obtain decidability results about our own theories. In addition, in the future we will consider the adversary's capability of linking voters and votes by means other than the proposed theory, e.g., an adversary might link voters and votes by IP addresses or even timestamps of messages. Furthermore, in order to integrate other security properties, e.g., integrity, a more adequate adversary model has to be considered. In future work, we therefore plan to allow the adversary to become active at an earlier stage, hence manipulating, dropping or injecting messages throughout the protocol run, thereby incorporating advances from the protocol analysis. Due to the nature of elections, the investigation of electronic voting schemes always comes along with legal considerations. Therefore, resilience terms compliant with legal frameworks need to be discussed and determined in close collaboration with legal scientists.

Acknowledgment

This paper has been developed within the project "ModIWa2" - Juristisch-informatische Modellierung von Internetwahlen - which is funded by the Deutsche Forschungsgemeinschaft (DFG, German Science Foundation).

 ISBN: 978-1-61208-201-1

REFERENCES

[1] Melanie Volkamer and Roland Vogt. Basic set of security requirements for Online Voting Products. (BSI-PP-0037). Common Criteria Protection Profile, 2008.

[2] Hugo Jonker and Melanie Volkamer. Compliance of RIES to the proposed e-voting protection profile. In *First International Conference on E-voting and Identity*, pages 50–61. Springer-Verlag, 2007.

[3] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *21st IEEE Computer Security Foundations Symposium*, pages 195–209. IEEE Computer Society, 2008.

[4] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *14th European Symposium On Programming*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.

[5] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, pages 244–251. Springer-Verlag, 1993.

[6] Jonathan Katz, Steven Myers, and Rafail Ostrovsky. Cryptographic counters and applications to electronic voting. In *Eurocrypt 2001*, pages 78–92, 2001.

[7] R. Küsters, T. Truderung, and A. Vogt. Proving coercion-resistance of Scantegrity II. In *12th International Conference on Information and Communications Security*, volume 6476, pages 281–295. Springer, 2010.

[8] Melanie Volkamer and Rüdiger Grimm. Determine the resilience of evaluated internet voting systems. In *First International Workshop on Requirements Engineering for e-Voting Systems*, pages 47 – 54. IEEE Digital Library, 2009.

[9] Komminist Weldemariam, Richard A. Kemmerer, and Adolfo Villafiorita. Formal specification and analysis of an e-voting system. In *Fifth International Conference on Availability, Reliability and Security*, pages 164–171, 2010.

[10] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 104–115. ACM, 2001.

[11] V. Cortier and B. Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. In *24th IEEE Computer Security Foundations Symposium*, pages 297 –311, 2011.

[12] H. L. Jonker and E. P. De Vink. Formalising receipt-freeness. In *Information Security*, volume 4176 of *LNCS*, pages 476–488. Springer, 2006.

[13] Rdiger Grimm, Katharina Hupf, and Melanie Volkamer. A formal IT-security model for the correction and abort requirement of electronic voting. In *4th International Conference on Electronic Voting, EVOTE*, volume 167 of *LNI*, pages 89–107. GI, 2010.

[14] Katharina Bräunlich and Rudiger Grimm. Formalization of receipt-freeness in the context of electronic voting. In *Sixth International Conference on Availability, Reliability and Security*, pages 119 –126, 2011.

[15] Walid Saad, Tansu Alpcan, Tamer Basar, and Are Hjørungnes. Coalitional game theory for security risk management. In *Fifth International Conference on Internet Monitoring and Protection*, pages 35–40. IEEE Computer Society, 2010.

[16] Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. The price of malice: A game-theoretic framework for malicious behavior in distributed systems. *Internet Mathematics*, 6:125–155, 2009.

[17] Tyrone Grandison and Morris Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4):2–16, 2000.

[18] Sebastian Ries. *Trust in Ubiquitous Computing*. PhD thesis, TU Darmstadt, 2009.

[19] 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. 2004.

[20] Kai Reinhard and Wolfgang Jung. Compliance of POLYAS with the BSI protection profile - basic requirements for remote electronic voting systems. In *First Conference on E-Voting and Identity*, pages 62–75, 2007.

[21] Ben Adida. Helios: Web-based open-audit voting. In *Proceedings of the 17th conference on security symposium*, pages 335–348. USENIX Association, 2008.

[22] Guido Schryen, Melanie Volkamer, Sebastian Ries, and Sheikh Mahbub Habib. A formal approach towards measuring trust in distributed systems. In *ACM Symposium on Applied Computing*, pages 1739–1745. ACM, 2011.

[23] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.

[24] Maina M. Olembo, Patrick Schmidt, and Melanie Volkamer. Introducing verifiability in the polyas remote electronic voting system. In *Sixth International Conference on Availability, Reliability and Security*, pages 127–134. IEEE, 2011.

[25] Ben Adida, Olivier Pereira, Olivier De Marneffe, and Jean Jacques Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of Helios. In *Electronic Voting Technology/Workshop on Trustworthy Elections*, 2009.

[26] Christoph Weidenbach, Uwe Brahm, Thomas Hillenbrand, Enno Keen, Christian Theobalt, and Dalibor Topić. SPASS version 2.0. In *18th International Conference on Automated Deduction*, volume 2392 of *LNAI*, pages 275–279. Springer, 2002.

[27] Mouhebeddine Berrima, Narjes Ben, Rajeb Veronique Cortier, Theme Sym, Mouhebeddine Berrima, Narjes Ben Rajeb, Veronique Cortier, and Equipe projet Cassis. Deciding knowledge in security protocols under some e-voting theories. In *20th International Conference on Rewriting Techniques and Applications*, pages 148–163. Springer, 2009.

 ISBN: 978-1-61208-201-1