A Comparison-Based Methodology for the Security Assurance of Novel Systems

Presentation proposal for NIST's Workshop on Formal Methods within Certification Programs

Jelizaveta Vakarjuk, Peeter Laud
Cybernetica AS
e-mail: firstname.lastname@cyber.ee

May 2024

Abstract

In this presentation, we propose the systematic comparison of systems as a method to transfer security assurances obtained for one system into assurances for a similar system, by arguing that any attack against the latter can be converted against an attack against the former, perhaps with the side conditions for the implementation of the latter system. We suggest that this method can give us good security assurances, and can be accessible with a relatively shallow learning curve. For comparing systems that are less similar to each other, we suggest the definition of intermediate systems, similar to the sequence of game-based proofs in cryptography. We have used our method to compare a threshold cryptography based authentication solution to a smartcard based solution, and internet voting to postal voting; we will present at least some of the details of these comparisons in our talk.

1 Introduction

Security evaluation is a costly and resource-consuming process, which can be even more complicated if we are dealing with systems built using novel technologies and/or cryptographic primitives. Therefore, it would be beneficial to have a methodology that allows us, having evaluated the security of certain system, to argue about the security of similar systems.

One of the most widely used frameworks for security evaluation is Common Criteria (CC) Certification [4]. In CC terms, one has to define a Target of Evaluation (ToE) – a system that is expected to pass certification. Passing the certification means that competent people have verified, up to a certain assurance level, that the ToE T is fulfilling a certain set of Security Functional Requirements (SFR). The methodology that we developed [5] allows to argue about the security of a modified ToE T', showing that T' is *at least as secure as* T. The paper [5] presents a methodology and provides an example of its application by comparing a smartcard based authentication system with an authentication system that relies on threshold cryptography [2]. A follow-up paper [6] compares voting through Internet and voting by mail (in Estonian setting), showing that our methodology is applicable also in contexts differing from the Common Criteria. Our methodology [5] can be divided into the following steps:

1. construct models of the systems that need to be compared;
2. identify the main data elements, processes and decision points that can be targeted by the adversary;
3. construct lists of potential weaknesses of the system by analysing / enumerating the identified data elements, processes and decision points in the system;
4. identify the relations between the listed weaknesses;
5. compare the sets of weaknesses for both systems.

Below, we give a short discussion of these steps. Appendix A contains copies of the papers [5, 6].

2 Systems and Attacks

The first step of the comparison process is to construct models of two systems T and T' that are chosen for comparison. We propose to use Business Process Model and Notation (BPMN) to create detailed models of the analysed systems. The BPMN diagrams allow detailed modeling the steps of the processes of T and T', depicting visually the sequence of activities performed by the different parts of the systems. Additionally, BPMN diagrams model the information flows that are needed to complete the process, while making it explicit which data elements are used for which processes. The BPMN models can also depict the resources that the processes use, and use annotations to describe non-functional properties of various elements. Therefore, constructed BPMN models allow us to perform the second step of our method: identifying all the information components (data elements), processes and decision points that can be targeted by the adversary.

Once the primary sources of weaknesses – main data elements, processes and decision points are identified, we can proceed to the third step: constructing lists of weaknesses in systems T and T'. To identify weaknesses, we consider the process steps where any data element is being stored, processed, or transmitted. These are the points where the adversary can interfere with the process, perform modifications, or get access to some data elements. We also consider computation and processing steps, which the adversary can interrupt or modify. Finally, we consider how the adversary can interfere with the decision-making (branching) steps in the process. The adversary may corrupt several of the system components and either exploit the identified weaknesses fully or not exploit them at all (e.g. if these are too expensive to exploit). To ease the analysis, we group weaknesses in the sets based on which party's process or data element is targeted by the adversary.

The fourth step consists of identifying relations between the identified sets of weaknesses W for system T and W' for T'. To each subset of weaknesses of T or T' we attach two quantities: the *difficulty* of exploiting this set of weaknesses, and the *seriousness* of the effects of successful exploitation by the adversary. We define two preorders on the set of all subsets of $W \cup W'$. Namely, for two sets of weaknesses $\mathbf{w}_1, \mathbf{w}_2 \in W \cup W'$ we write:

- $\mathbf{w}_1 \prec_{\mathrm{D}} \mathbf{w}_2$, if exploiting all weaknesses in $\mathbf{w}_1$ is no more difficult for the adversary than exploiting all weaknesses in $\mathbf{w}_2$,
- $\mathbf{w}_1 \prec_{\mathrm{S}} \mathbf{w}_2$ if the effects of exploiting weaknesses in $\mathbf{w}_1$ are no worse than the effects of exploiting weaknesses in $\mathbf{w}_2$.

A natural corollary of these definitions is, that if $\mathbf{w}_1 \subseteq \mathbf{w}_2$, then also $\mathbf{w}_1 \prec_{\mathrm{D}} \mathbf{w}_2$ and $\mathbf{w}_1 \prec_{\mathrm{S}} \mathbf{w}_2$.

The fifth step of our method is to compare the sets of weaknesses of both systems T and T'. For each set $\mathbf{w}' \subseteq W'$ we want to show that there exists some $\mathbf{w} \subseteq W$, such that $\mathbf{w}' \prec_{\mathrm{S}} \mathbf{w}$ and $\mathbf{w} \prec_{\mathrm{D}} \mathbf{w}'$. It is also likely that we only consider "reasonable" adversaries, where reason is defined by restricting the considered subsets of both W' (simplifying this step of our method) and W (making this step harder).

In order to compare sets of weaknesses and identify relations between the weaknesses, it is important to understand what is the result of a successful exploitation of a weakness (attack). Additionally, it is needed to identify which attacks can be combined with each other to result in a more serious attack. For example, learning one private key share in a threshold signature scheme (2-out-of-2) does not allow adversary to create valid signatures, but in combination with the second private key share, it gives the adversary the ability to create signatures on various messages. Additional important aspect to consider is the criticality of the identified attack. Critical attack is the one that is targeting the main security properties (e.g. for voting systems – vote secrecy, vote integrity, ballot box integrity).

3 Sequences of systems

The processes in the initial models for the systems T and T' can be too different for direct comparison, meaning that the relations $\prec_{\mathrm{D}}$ and $\prec_{\mathrm{S}}$ between the sets of weaknesses are too sparse. Therefore, we propose to introduce *intermediate systems* to be able to complete comparison process. We use the concept similar to the game-based security proofs in cryptography [1]. In these security proofs, it is common to introduce *intermediate* games, such that any two neighbouring games differ only slightly from each other. The zeroth game comes directly from the security definition, and the final game is the one where the adversary's win is obviously unlikely.

We use similar strategy in our methodology, the zeroth system T' is the novel one that we want to certify/reason about its security, the final system T is the one that has already been certified/analysed, and we may introduce "intermediate" systems that are easier to compare against the previous and next ones. Intermediate systems are created by introducing some smaller changes to the process making it successively more similar to the final system T. Systems that are introduced in this way are not intended to be implemented; they may make no economic sense, or may even be impossible to implement. However, they have their well-defined sets of weaknesses, such that the subsets of weaknesses can be compared against each other. We present intermediate systems as BPMN models again. We analyse which attacks are removed during a switch from one intermediate system to the next one (including T and T'), and which new attacks are introduced. The examples can be found in [5, 6].

4 Uncovered security requirements for the system T'

The proposed methodology may identify security requirements that should be in place for the novel system T', if we want to show it to be as secure as the system T. In the process of comparing certified system T to a novel one T', we may find that the components of system T employ certain security mechanisms, for which there should be corresponding mechanisms in T' in order for the comparison to go through and the necessary containment of the sets of weaknesses to hold. For example, if the communication channels used by these systems are different, then we may need to ensure that the channel used by the system T' has at least as good security properties as the channel used by the system T. Similar requirements may be discovered if the data storage for certain sensitive values is different. In longer sequences of systems, it may happen that those requirements are only applicable to the intermediate systems and do not matter for system T'. However, it is also possible that the discovered requirements propagate further in this sequence, all the way towards the system T' whose security we want to claim.

5 Performed analyses

The papers [5, 6] present the two analyses that we have performed using the introduced methodology. In the first one [5], we compared two authentication systems, where the user's browser establishes a secure connection with a relying party's (RP) server, and the user is authenticated during this process by signing a challenge generated by the RP. In this analysis, the secure system T was based on secure hardware: the private key was contained in, and the signing was performed by a smartcard that communicated with the user's browser through a card reader. The smartcard was activated by the user entering the PIN either through the device executing the browser, or through a PINpad on the card reader.

The system T' in our first analysis was based on threshold cryptography [2]: the two shares of the private key were stored in user's smartphone, and in a central server. The former was activated by the entry of a PIN, and the latter by the former. The smartphone was considered to be a device that does not provide strong security guarantees by its own. The signing protocol contained mechanisms to detect the compromise of the phone.

Our analysis showed that the threshold cryptography based mechanism may be at least as secure as the smartcard based mechanism, if certain subcomponents of T' compare favorably with components of T. We found that the security of the communication channel between the phone and the server must be as secure as the channel between the smartcard reader and user's device running the browser. We also found that the server must protect the keyshare and the integrity of the signing process at least as well as the smartcard protects the signing key and the process. Perhaps the most interesting finding was, that the leakage of the PIN during its entry to the smartphone had to be at least as unlikely as when entering it to the system with a smartcard. In case the PIN is entered through computer's keyboard in system T, this comparison is plausible. But it will be harder to argue, if a card reader with PINpad is employed.

In our second analysis [6], we compared two voting systems that allow to cast votes remotely, focusing on the vote casting phase. The first system T is postal voting system, where the voter fills in the ballot and seals it into the inner envelope. The inner envelope is then placed into the outer one. Next, the voter writes their name and identification code on the outer envelope and sends it to the foreign mission through the postal service. Such systems are in use in many countries of the world.

The system T' in the second analysis is an internet voting system [3]. The voter uses a dedicated voting application to cast the vote. The voter is first required to authenticate themselves through the application. Next, the voter makes the choice, which is first encrypted under the system's public key and then digitally signed with the voter's signing device.The voting application then generates and displays a QR code that the voter can use to verify the vote using the verification app running on a smartphone.

Our analysis showed that the considered internet voting system indeed has stronger security guarantees than the postal voting system. Additionally, the analysis identified that in the internet voting system critical attacks that are targeting the main security properties (vote secrecy, vote integrity, ballot box integrity) are distributed between many system components and may require learning different information from different parties in order to execute. In the postal voting system, on the other hand, there are less participating parties and some critical attacks can be executed by corrupting only one of those parties.

Acknowledgements. This work has been supported by Estonian Research Council, grant no. PRG1780.

References

[1] Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[2] Ahto Buldas, Aivo Kalu, Peeter Laud, and Mart Oruaas. Server-Supported RSA Signatures for Mobile Devices. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part I*, volume 10492 of *Lecture Notes in Computer Science*, pages 315–333. Springer, 2017.

[3] Sven Heiberg and Jan Willemson. Verifiable internet voting in Estonia. In Robert Krimmer and Melanie Volkamer, editors, *6th International Conference on Electronic Voting: Verifying the Vote, EVOTE 2014, Lochau / Bregenz, Austria, October 29-31, 2014*, pages 1–8. IEEE, 2014.

[4] ISO/IEC 15408-1/2/3:2005 - Information technology — Security techniques — Evaluation criteria for IT security.

[5] Peeter Laud and Jelizaveta Vakarjuk. A comparison-based methodology for the security assurance of novel systems. In Sokratis K. Katsikas, Frédéric Cuppens, Christos Kalloniatis, John Mylopoulos, Frank Pallas, Jörg Pohle, M. Angela Sasse, Habtamu Abie, Silvio Ranise, Luca Verderame, Enrico Cambiaso, Jorge Maestre Vidal, Marco Antonio Sotelo Monge, Massimiliano Albanese, Basel Katt, Sandeep Pirbhulal, and Ankur Shukla, editors, *Computer Security. ESORICS 2022 International Workshops - CyberICPS 2022, SECPRE 2022, SPOSE 2022, CPS4CIP 2022, CDT&SECOMANE 2022, EIS 2022, and SecAssure 2022, Copenhagen, Denmark, September 26-30, 2022, Revised Selected Papers*, volume 13785 of *Lecture Notes in Computer Science*, pages 625–644. Springer, 2022.

[6] Jelizaveta Vakarjuk, Nikita Snetkov, and Jan Willemson. Comparing security levels of postal and internet voting, 2024. In Review.

A Papers [5, 6] presenting the method

A Comparison-Based Methodology for the Security Assurance of Novel Systems

Peeter Laud and Jelizaveta Vakarjuk

Cybernetica AS
peeter.laud|jelizaveta.vakarjuk@cyber.ee

Abstract. In this paper, we advocate the position that the security certification of one system should make the certification of other similar systems easier, if one can present the evidence that the second system is at least as secure as the first system. We present a development of this idea, stating the components of such comparative evidence. We stretch the idea of propagating the certification to less similar systems, if one can present a sequence of systems from the certified one to the novel one, where each system is evidenced to be at least as secure as the previous one. We apply our methodology to authentication systems, where we show that a system based on threshold cryptography is at least as secure as widely used smartcard-based systems.

1 Introduction

Our critical systems are built on top of, and their security assurances depend on smaller subsystems, whose security is of utmost importance, and where the society has chosen security certification as the methodology to ensure that these subsystems satisfy the required properties. These small systems and devices include smartcards, hardware security modules, trusted execution environments, but also certain communication devices, operating systems and application software. A lot of trust is placed on the outcomes of certification, hence this process is expensive [9] and conservative [20]. When a certified system has been updated, then the new version has to pass the certification again. A system making use of novel security technologies may have hard time passing certification at all, because these technologies may not map cleanly to the categories specified in certification procedures.

In Common Criteria (CC) certification [8], we speak of *Targets of Evaluation* (ToE); this is the well-delimited system that is expected to pass certification. A *Security Target* (ST) document is compiled for certifying a particular ToE; it gives the boundaries of ToE, and lists the security requirements that ToE is expected to fulfill. These requirements are taken from standardised lists, enumerated in CC specifications. A security target may comply with, i.e contain one or more *Protection Profiles* (PP), which are themselves lists of security requirements. The security requirements are either *Security Functional Requirements* (SFR), referring to the security technologies that the ToE may use in a certain

manner in order to obtain certain security properties, or *Security Assurance Requirements* (SAR), which refer to the good practices used during the design, development, and operation of the ToE. In this paper, we use CC-related terms, but our proposed methodology should be applicable to other certification frameworks, too.

When a ToE evolves, then it may still fulfill the same SFRs listed in the ST of the old ToE, but it may fulfill them in novel manner. When a ToE makes use of novel technologies, it may be difficult to map the technical protections it offers to the fulfilment of SFRs in the ST. In both cases, it may be helpful to not match the new ToE directly against SFRs, but to compare the new ToE to the old one and make sure that the new one is *at least as secure*. This is the position we advocate in this paper: comparing two ToE-s — one with existing certification against a ST, and the other that we desire to get certified against a similar ST — can simplify the certification process, while preserving the assurances of the existing certification. The comparison establishes, that for any attack against the novel system, there exists an attack against the certified system that is no more difficult to perform, and with the same or worse consequences. We argue that such arguments should be accepted by certification bodies. Even if they are not directly accepted, the outcomes of the comparison should guide the vendor in changing the evidence documentation that supports the claims of ToE satisfying the ST.

Our proposal is similar to *game-based* security proofs of cryptographic primitives [1]. In cryptography, security definitions define an interactive attack game that is played between the adversary and the environment; the latter provides an interface to the adversary through which the primitive may be invoked in well-defined ways. The primitive is deemed secure if no adversary's probability of *winning* the game exceeds a certain value. A typical security proof presents a different game where the same adversary obviously has only a low probability of winning, and then argues that in the original game, the adversary's win is not much more likely.

In these security proofs, it is common to introduce *intermediate* games, such that any two neighbouring games differ only slightly from each other, simplifying the argument that the adversary's winning probability in i-th game is at most negligibly larger than in the $(i+1)$-st game. Here the zeroth game comes from the security definition, and the final game is the one where the adversary's win is obviously unlikely. We advocate that we can do something similar for certification: the zeroth ToE is the novel one that we want to certify, the final ToE has already been certified, and we may introduce "intermediate" ToE-s that are easier to compare against the previous and next ones.

Related work. The certification of evolving systems, the simplification of updating the evidence, and the reuse of evaluation results have been the topic of a statement from the CC Recognition Arrangement Management Committee [16]. Methods for using CC together with agile product development [18,19], and for continued certification of cyber security products [5] have been proposed. But we are aware of only a few attempts to show the security of one system by compar-

ing it to another one, where the latter one has already been deemed secure [3]. The other example is Trusted Computer System Evaluation Criteria (TCSEC), where the system evaluation process included the Rating Maintenance Phase (RAMP). The goal of RAMP was to provide an instrument to extend evaluation of a certified system to a new version of that system by analysing changes that were introduced [10].

In the rest of this paper, we describe our proposal for comparing two systems in more detail. We will then give an example of comparing two systems, where one of them is based on well-established hardware security technologies, while the other one makes use of threshold cryptography. In order to compare them against each other, we come up with a couple of intermediate systems, and argue that each of them is at least as secure as the next one.

2 Proposed methodology

Suppose we have a ToE T', for which we want to provide evidence that it matches a security target. We also have another system T, which we have already shown to match the same or a similar security target. We have the detailed specifications of both T and T', such that we can make a list of all potential weaknesses that an adversary may exploit in T or in T'. We may already have used these lists to create the security target [7]. For describing the processes of T and T', BPMN [11] may be a good choice, perhaps annotated further [14,17] to show the protection mechanisms in use. We create lists of weaknesses with high granularity, considering each data storage, movement, or processing step as something that the adversary may obtain information from, or tamper with the inputs and outputs of. The high granularity also means that an adversary may exploit one weakness either fully, or not exploit it at all, but there are no partial exploitations. Let W be the set of all weaknesses of T, and W' the set of all weaknesses of T'. Also, let $\mathcal{W}$ [resp. $\mathcal{W}'$] be the set of subsets of W [resp. W']. When an adversary performs an attack against T or T', then the set of weaknesses it exploits is an element of $\mathcal{W}$ or $\mathcal{W}'$.

The security target specifies the operational environment of the ToE, including the threats, organisational policies, and security assumptions. Hence not any set of weaknesses is considered exploitable against the system T. Rather, the security target specifies the set of *considered sets of weaknesses* $\mathcal{W}_{\mathrm{C}} \subseteq \mathcal{W}$ that contains only such sets of weaknesses that an adversary may *attempt* to exploit according to the security target. Here the attempts by the adversary do not correspond to his success; rather, the ToE is meant to employ measures that make the adversary unsuccessful. It is natural to assume that $\mathcal{W}_{\mathrm{C}}$ is downwards closed: if $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{W}$, $\mathbf{w}_1 \subseteq \mathbf{w}_2$, and $\mathbf{w}_2 \in \mathcal{W}_{\mathrm{C}}$, then also $\mathbf{w}_1 \in \mathcal{W}_{\mathrm{C}}$.

The security target for T' similarly specifies $\mathcal{W}'_{\mathrm{C}} \subseteq \mathcal{W}'$. As T' has not been certified yet, our methodology may bring adjustments to the precise definition of $\mathcal{W}'_{\mathrm{C}}$; hopefully not reducing it by too much.

To each set of weaknesses of T or T' we can attach two (abstract) quantities: the *difficulty* (for the adversary) of exploiting this set of weaknesses, and the

seriousness of the effects of successful exploitation by the adversary. We do not define the structure of these quantities, but we require that there is a (partial) order on them. Hence we get two preorders on $\mathcal{W} \cup \mathcal{W}'$: for two sets of weaknesses $\mathbf{w}_1$ and $\mathbf{w}_2$ we write $\mathbf{w}_1 \prec_\mathrm{D} \mathbf{w}_2$, if exploiting all weaknesses in $\mathbf{w}_1$ is no more difficult for the adversary than exploiting all weaknesses in $\mathbf{w}_2$. We also write $\mathbf{w}_1 \prec_\mathrm{S} \mathbf{w}_2$ if the effects of exploiting $\mathbf{w}_1$ are no worse than the effects of exploiting $\mathbf{w}_2$. A natural corollary of these definitions is, that if $\mathbf{w}_1 \subseteq \mathbf{w}_2$, then also $\mathbf{w}_1 \prec_\mathrm{D} \mathbf{w}_2$ and $\mathbf{w}_1 \prec_\mathrm{S} \mathbf{w}_2$.

It is possible that for some $\mathbf{w}_1 \neq \mathbf{w}_2$ we have both $\mathbf{w}_1 \prec_\mathrm{D} \mathbf{w}_2$ and $\mathbf{w}_2 \prec_\mathrm{D} \mathbf{w}_1$ (denote this $\mathbf{w}_1 \sim_\mathrm{D} \mathbf{w}_2$), hence $\prec_\mathrm{D}$ is not an order, but only a preorder. The same applies to $\prec_\mathrm{S}$. It may be difficult to describe all refinements that $\prec_\mathrm{D}$ and $\prec_\mathrm{S}$ have with respect to the subset inclusion. However, an imprecise description does not invalidate the outcomes of our proposed analysis.

We want to show that for each $\mathbf{w}' \in \mathcal{W}'_\mathrm{C}$, there exists some $\mathbf{w} \in \mathcal{W}_\mathrm{C}$, such that $\mathbf{w} \prec_\mathrm{D} \mathbf{w}'$ and $\mathbf{w}' \prec_\mathrm{S} \mathbf{w}$. We thus have to present a function $f : \mathcal{W}'_\mathrm{C} \to \mathcal{W}_\mathrm{C}$, such that $f(\mathbf{w}') \prec_\mathrm{D} \mathbf{w}'$ and $\mathbf{w}' \prec_\mathrm{S} f(\mathbf{w}')$ for all $\mathbf{w}' \in \mathcal{W}'_\mathrm{C}$. We have found it easier to present not a function f, but a relation $F \subseteq \mathcal{W}' \times \mathcal{W}$, requiring that each $\mathbf{w}' \in \mathcal{W}'_\mathrm{C}$ is related to at least one element of $\mathcal{W}_\mathrm{C}$.

Each element of $\mathcal{W}'$ [resp. $\mathcal{W}$] is a subset of W' [resp. W]; hence it has the natural representation as a characteristic vector of type $W' \to \{0,1\}$ [resp. $W \to \{0,1\}$]. The entire relation F is thus represented as a boolean function $\phi : ((W' \dot\cup W) \to \{0,1\}) \to \{0,1\}$, which in turn is conveniently represented as a boolean formula. The variables of this formula are exactly the weaknesses of T and T'.

We say that two systems T and T' are *similar* if there is enough relationships $\prec_\mathrm{D}$ and $\prec_\mathrm{S}$ between the corresponding attack sets. If the systems T and T' are sufficiently dissimilar, then the construction of ϕ may fail, because of the lack of relationships $\prec_\mathrm{D}$ and $\prec_\mathrm{S}$ between attack sets $\mathbf{w} \in \mathcal{W}$ and $\mathbf{w}' \in \mathcal{W}'$. Such lack of (obvious) relationships may inform us of the security mechanisms that we need to use in T' in order to follow our methodology — the security mechanisms used by T' have to be strong enough to justify the addition of $\prec_\mathrm{D}$- or $\prec_\mathrm{S}$-relationships between attack sets against T and T'. These mechanisms may also be necessary for T' to match the security target. If intermediate systems have been introduced, then the requirements to use stronger security mechanisms may propagate from T through these systems all the way back to the system T'.

3 Example

In this section, we show how to use the proposed methodology to compare two authentication systems, where the user's browser establishes a secure connection with a relying party's (RP) server, and the user is authenticated during this process. In both systems, the RP server, knowing the public key of the user, generates a challenge, which has to be signed with the corresponding private key. The systems differ in how the private key is stored, and how the signature is generated.

An authentication system has to provide processes for the full lifecycle of a user's keypair: how it is generated, how a certificate is issued for it, how the RP finds the public key, how the challenges are signed, how the key is revoked. A full comparison of the two systems has to consider all of them. For concreteness, in this paper we will only focus on the authentication process itself, which includes the signing, and also some details of the storage of the private key.

The first system T has a smartcard as its centerpiece, containing the user's private key, and issuing signatures with it when activated. There exist protection profiles for such devices [13,12], and a number of cards or chips on the card have been certified to satisfy them. A typical use of a smartcard is as part of the Client Certificate Authentication (CCA) in the Transport Layer Security (TLS) protocol [15].

The second system T' uses threshold cryptography [4], sharing the private key between the user's phone and a central server. It can be used to authenticate the client end in an established TLS session. When attempting to directly certify such a system, it may be difficult to argue that the user has sufficient control over the private key — neither the phone nor the remote server alone provides such control.

3.1 Authentication with a smartcard

When a relying party wants to make sure it is talking to a device of a user with a given public key, it will create a challenge bit-string and ask the device to sign it. If the corresponding private key is stored in a smartcard, used with a card reader, then the challenge goes from the client application to the card reader, then together with the PIN (entered by the user on the PINpad if the card reader has one, and on the keyboard otherwise) to the smartcard, which checks the PIN and creates the signature, which then moves back the same way. If there have been too many wrong entries of the PIN, then the card locks up. The whole process is displayed as a BPMN diagram in Figure 1, split into relatively atomic pieces.

The card stores the private key, and the PIN. The user also knows the PIN. Inside the card, the PIN entered by the user is compared against the PIN that the card is storing. The environment of the card is protected, meaning that the PIN and the private key (or any details about it) cannot be read out, and the logic that it executes cannot be thwarted. The card reader is also protected, in that the PIN entered on its pad will only be sent to the card and not anywhere else.

This process, and the protection offered by it is considered a good example of two-factor authentication, showing that the user knows the PIN (knowledge factor), and the user has the card (possession factor). It is considered a good example even if the PIN entry is less isolated from the outside world, i.e. when the user enters the PIN on the computer.

Weaknesses of a system with a smartcard. In this process, the following weaknesses can be identified. These are the elements of the set W — the pos-

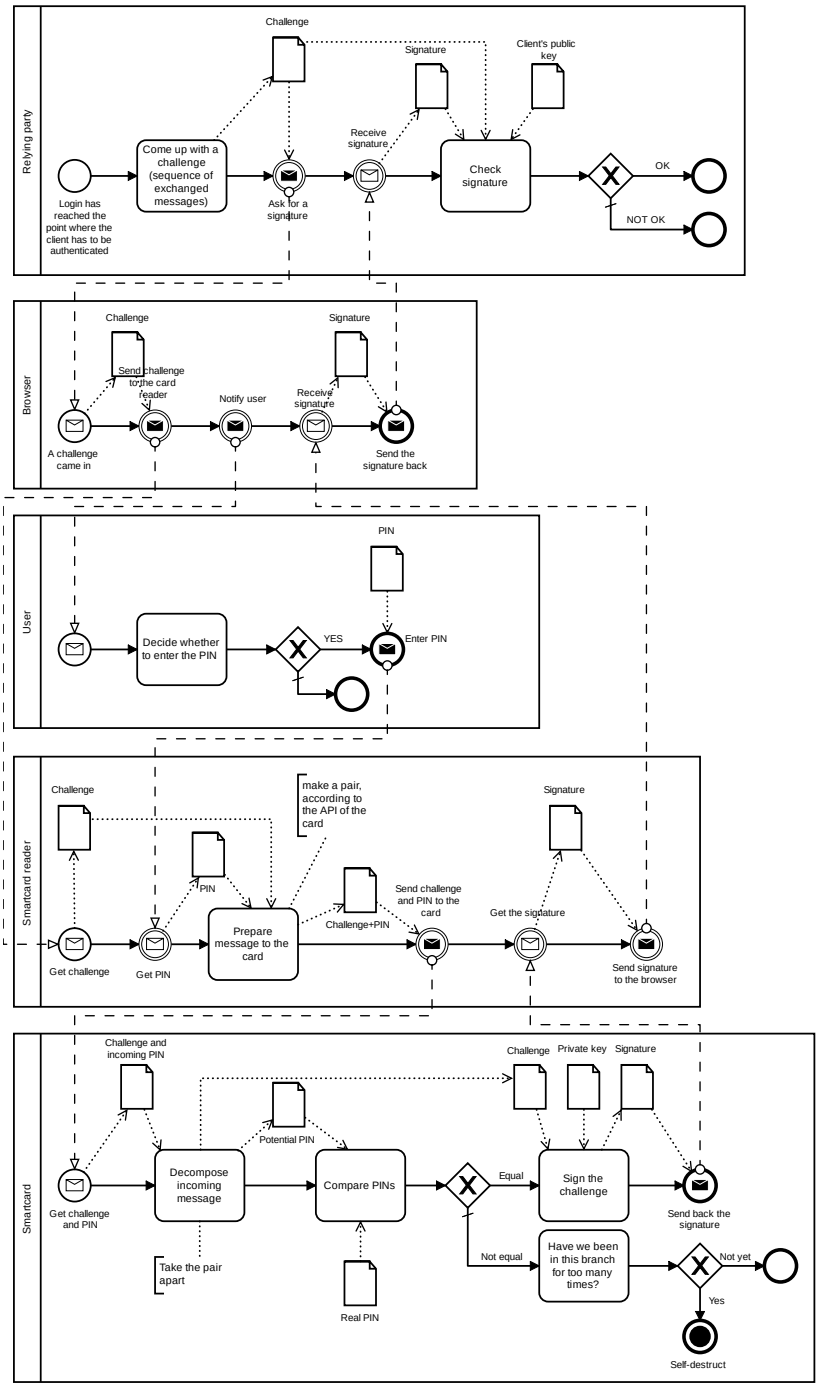

Fig. 1. The smartcard system

sible locations where an adversary could mount an attack. As the system T is considered secure, none of these locations, or their combinations are actually an exploitable weakness, i.e. $\mathcal{W}_C = \mathcal{W}$.

Possible attacks against the relying party

(RP1) Affect the computation of the challenge
(RP2) Learn the challenge
(RP3) Modify the challenge while it is sent to the browser
(RP4) Change the outcome of the signature check
(RP5) Accept the log-in, even if the signature does not check

Possible attacks against the browser

(B1) Learn the challenge
(B2) Modify the challenge while it is sent to the card reader
(B3) Modify the signature while it is sent to the relying party

Possible attacks against the user

(U1) Learn the PIN from the user
(U2) Change the PIN

Possible attacks against the smartcard reader

(SCR1) Learn the PIN that the user entered
(SCR2) Change the challenge that is sent to the smartcard
(SCR3) Change the PIN that is sent to the smartcard
(SCR4) Change the signature while it is sent to the browser

Possible attacks against the smartcard

(SC1) Learn the PIN
(SC2) Interfere with the PIN comparison procedure
(SC3) Make the decision of the PIN check take the other path
(SC4) Make the decision about counts of incorrect PINs take the other path
(SC5) Learn the private key
(SC6) Change the private key
(SC7) Change the challenge that enters the computation of the signature
(SC8) Learn the signature
(SC9) Change the signature sent back to the smartcard reader

3.2 Authentication with SplitKey

In systems relying on threshold cryptography, where the private key is stored in a secret-shared manner, several storage devices with the shares of the key have to be present in order to make use of that private key. If all the storage devices offer strong properties of unclonability, tamper-resistance and similar, then we can have even stronger security properties for the authentication process, compared to using just a single device.

However, we typically want to use threshold cryptography in order to reduce the requirements on the devices storing, and on the procedures for accessing individual shares of the private key. If we still want to argue that our authentication procedure has strong security properties similar to logging in with a smartcard, then we have to involve the properties of several devices in our arguments.

In particular, the SplitKey technology [2] for signature creation shares the private key among two devices, such that one of the devices – the phone – does not offer a strong protection for its key share. Hence, in an authentication system based on SplitKey, when arguing about the possession of a private key, we have to involve the second device in our arguments. This involvement may touch the physical properties of that device, as well as technical and organisational properties controlling the access to it.

The authentication procedure is depicted in Figure 2, in similar detail to the previous figure. Let us recall that in SplitKey, an RSA private key has been shared between the user's phone and the server, and the keyshare on the phone is encrypted with a low-entropy key (derived from the PIN that the user enters). If the adversary has obtained just the encrypted keyshare, then it cannot recognize, which PIN is the correct one. However, the server can recognize if it has received a signature share from the phone that has been created with a wrong keyshare.

When the phone has received the challenge, it asks the user to input the PIN. In order to help the user understand the context, in which the signing of the challenge is about to occur, the phone shows the user a short *control code*, derived from the challenge. The same control code is shown to the user by the relying party's website. The user is supposed to enter the PIN only if the control codes match.

Weaknesses of a SplitKey-based system. Considering the steps in Fig. 2, we can list the following possible weaknesses.

Attacks against the relying party:

(RP1) Affect the computation of the challenge
(RP2) Learn the challenge
(RP3) Affect the computation of the control code
(RP4) Learn the control code
(RP5) Modify the challenge while it is sent to the phone
(RP6) Change the outcome of the signature check
(RP7) Accept the log-in, even if the signature does not check

Attacks against the browser

(B1) Learn the control code
(B2) Change the control code, when it is shown to the user

Attacks affecting the user

(U1) Learn the PIN from the user
(U2) Change the PIN

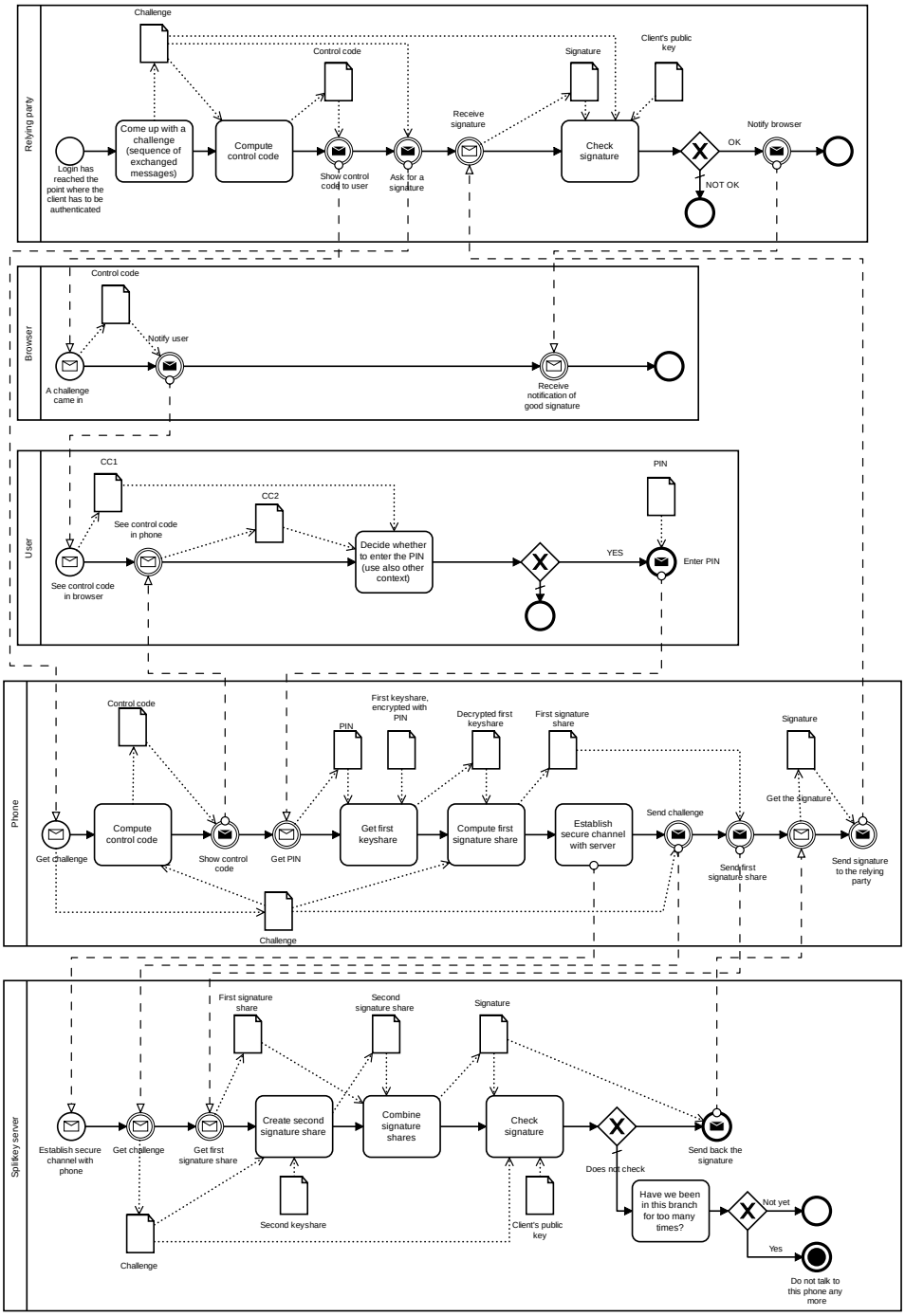

Fig. 2. SplitKey authentication system

(U3) Cause the user to incorrectly compare the control codes

Attacks against the phone

(P1) Affect the computation of the control code
(P2) Change the control code on the phone screen, where it is shown to the user
(P3) Learn the PIN that the user enters
(P4) Learn the encrypted first keyshare (encrypted with PIN)
(P5) Learn the plain first keyshare
(P6) Change the encrypted first keyshare, before it has been decrypted with the PIN
(P7) Change the plain first keyshare
(P8) Change the challenge that enters the computation of the first signature share
(P9) Learn the first signature share
(P10) Change the first signature share
(P11) Interfere with the establishment of the secure channel with the SplitKey server, thereby obtaining the capability to read and/or change messages sent over it
(P12) Change the challenge sent to the SplitKey server
(P13) Change the first signature share sent to the SplitKey server
(P14) Change the signature sent to the relying party

Attacks against the Splitkey server

(SS1) Interfere with the establishment of the secure channel with the phone, thereby obtaining the capability to read and/or change messages sent over it
(SS2) Interfere with the establishment of the secure channel with the phone, thereby confusing the server on the identity of the phone
(SS3) Learn the second keyshare
(SS4) Change the second keyshare
(SS5) Change the challenge that goes into the second signature share creation process
(SS6) Learn the second signature share
(SS7) Interfere with the signature combination process
(SS8) Learn the signature
(SS9) Change the client's public key, against which the signature is checked
(SS10) Interfere with the signature checking procedure
(SS11) Make the decision of the signature check go otherwise
(SS12) Change the signature sent back to the phone
(SS13) Make the decision about counts of unsuccessful signature creations take the other path

Among the attacks against the SplitKey system, we can identify the following relationships for $\prec_{\mathrm{D}}$ and $\prec_{\mathrm{S}}$. If the adversary learns both the PIN and the encrypted first keyshare, then he also has the plain first keyshare: $\{\mathrm{P5}\} \prec_{\mathrm{X}} \{\mathrm{P3}, \mathrm{P4}\}$ and $\{\mathrm{P5}\} \prec_{\mathrm{X}} \{\mathrm{U1}, \mathrm{P4}\}$, where X may be both D and S. Changing

a value has no greater effect than changing everything that is computed from this value. In particular, $\{\text{SS4}\} \prec_{\text{S}} \{\text{SS7}\} \prec_{\text{S}} \{\text{SS10}\} \prec_{\text{S}} \{\text{SS11}, \text{SS12}\}$ and $\{\text{P6}\} \prec_{\text{S}} \{\text{P7}\} \prec_{\text{S}} \{\text{P12}\} \prec_{\text{S}} \{\text{SS7}\}$. The same relationships generally continue to hold when we add the same additional attacks to both sides of $\prec_{\text{X}}$.

We are using threshold cryptography, where the adversary learning just a single share of a secret should not yet affect the seriousness of attacks that he can perform. In particular, we would like to state that $\{\text{P5}\} \cup \mathbf{w} \prec_{\text{S}} \mathbf{w}$ for a significant class of attack sets $\mathbf{w}$ that do not contain attacks against the second keyshare. We state this for all $\mathbf{w}$ that do not contain SS3 or SS6. Instead of the set $\{\text{P5}\}$, we can also consider other sets that imply the knowledge of the first keyshare or signature share: $\{\text{P3}, \text{P4}\}$ and $\{\text{P9}\}$.

3.3 First intermediate system

The systems T and T' are too different for the direct application of the comparison methodology we gave in Sec. 2. As we see below, we can cross this gap by proposing a couple of intermediate systems.

We change the system T' into the system T_1, where instead of computing the first signature share on the phone, we move all the computations into the Splitkey server. We let the Splitkey server store the user's private key share encrypted with the user's PIN and we send the PIN together with the challenge from the phone to the SplitKey server. The authentication process in the first intermediate system is depicted in Figure 3.

Weaknesses of the first intermediate system. Obviously the possible attacks against the relying party, browser, and user are the same as in the SplitKey-based system, as there have been no changes to this part of the system. Attacks against the phone and the server have changed; quite often, the change is simply in the target device.

Attacks against the phone

(P1) Affect the computation of the control code
(P2) Change the control code on the phone screen, where it is shown to the user
(P3) Learn the PIN that the user enters
(P4) Interfere with the establishment of the secure channel with the SplitKey server, thereby obtaining the capability to read and/or change messages sent over it
(P5) Change the challenge sent to the SplitKey server
(P6) Change the signature sent to the relying party
(P7) (**new**) Change the PIN sent to the SplitKey server

Attacks against the SplitKey server

(SS1) Interfere with the establishment of the secure channel with the phone, thereby obtaining the capability to read and/or change messages sent over it

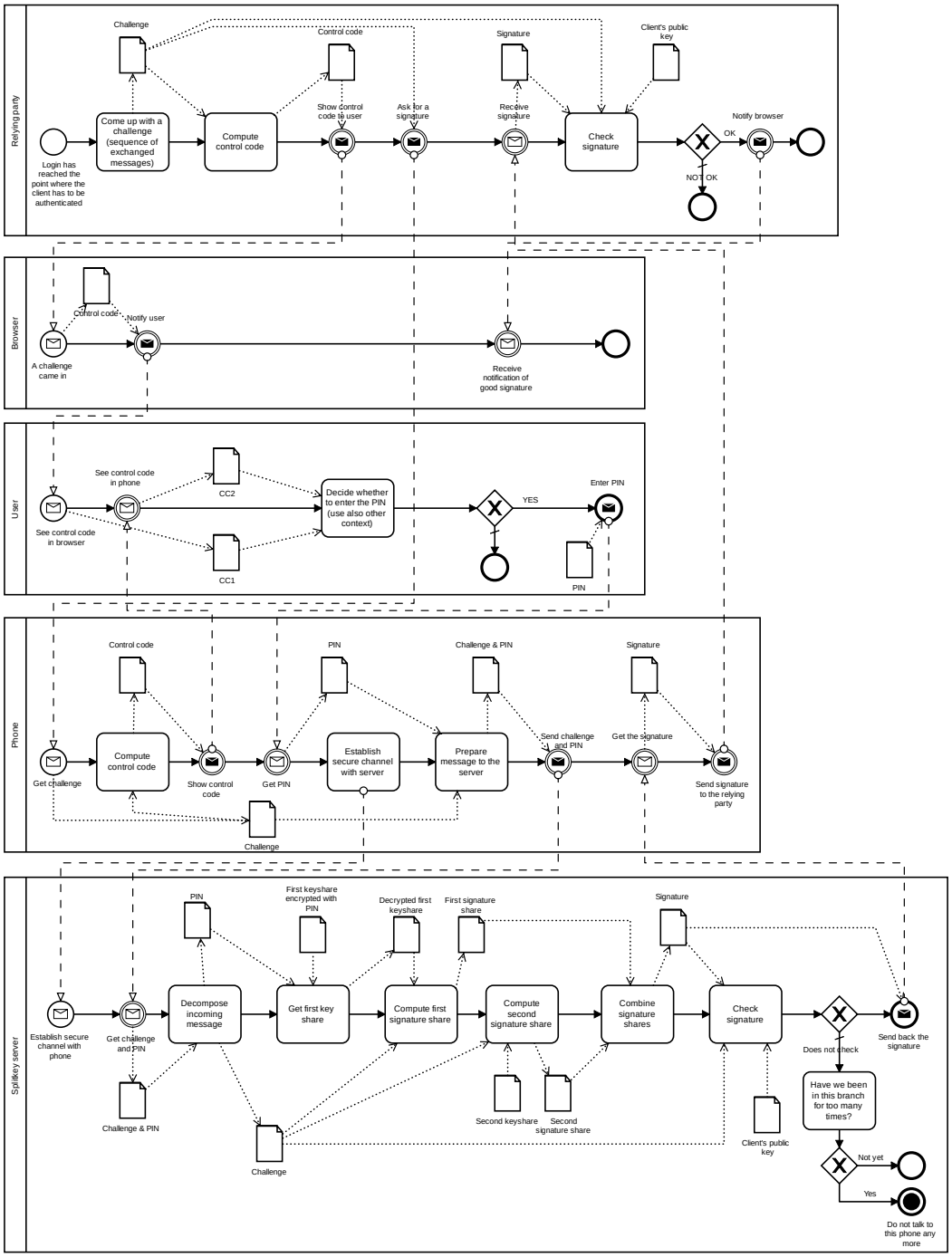

Fig. 3. The first intermediate system

(SS2) Interfere with the establishment of the secure channel with the phone, thereby confusing the server on the identity of the phone
(SS3) Learn the second keyshare
(SS4) Change the second keyshare
(SS5) Change the challenge that goes into the second signature share creation process
(SS6) Learn the second signature share
(SS7) Interfere with the signature combination process
(SS8) Learn the signature
(SS9) Change the client's public key, against which the signature is checked
(SS10) Interfere with the signature checking procedure
(SS11) Make the decision of signature check go otherwise
(SS12) Change the signature sent back to the phone
(SS13) Make the decision about counts of unsuccessful signature creations take the other path
(SS14) (**moved**) Learn the PIN received from the phone
(SS15) (**moved**) Learn the encrypted first keyshare (encrypted with PIN)
(SS16) (**moved**) Learn the plain first keyshare
(SS17) (**moved**) Change the encrypted first keyshare, before it has been decrypted with the PIN
(SS18) (**moved**) Change the plain first keyshare
(SS19) (**moved**) Change the challenge that enters the computation of the first signature share
(SS20) (**moved**) Learn the first signature share
(SS21) (**moved**) Change the first signature share

For the system T_1, we can again identify certain relationships for $\prec_{\mathrm{D}}$ and $\prec_{\mathrm{S}}$. All the relationships from T' are present, except that some of them have changed their numbers (and also moved their target from the phone to the server). We also assume that the different values the server computes during the signature creation process are equally difficult for an adversary to learn or change; thus $\{\mathrm{SS16}\} \prec_{\mathrm{D}} \{\mathrm{SS3}\}$, $\{\mathrm{SS6}\} \sim_{\mathrm{D}} \{\mathrm{SS20}\} \sim_{\mathrm{D}} \{\mathrm{SS8}\}$ and $\{\mathrm{SS5}\} \sim_{\mathrm{D}} \{\mathrm{SS19}\}$.

We also have relationships between the weaknesses of T' and T_1. We may assume that if $\mathbf{w}$ is a set of weaknesses for both of them (i.e. the names are the same), then $T'.\mathbf{w} \sim_{\mathrm{S}} T_1.\mathbf{w}$. If the targets of the weaknesses in $\mathbf{w}$ are also the same, then also $T'.\mathbf{w} \sim_{\mathrm{D}} T_1.\mathbf{w}$. For example: $\{T'.\mathrm{P4}\} \sim_{\mathrm{S}} \{T_1.\mathrm{SS15}\}$, but $\{T'.\mathrm{P4}\} \not\sim_{\mathrm{D}} \{T_1.\mathrm{SS15}\}$.

Comparison between T' and T_1. We can now write down the formula ϕ that matches each set of attacks against T' with a set of attacks against T_1. Denote $T'.\mathrm{K} \equiv (T'.\mathrm{P3} \wedge T'.\mathrm{P4}) \vee T'.\mathrm{P5}$ and $T_1.\mathrm{K} \equiv (T_1.\mathrm{SS14} \wedge T_1.\mathrm{SS15}) \vee T_1.\mathrm{SS16}$, i.e. "K" denotes the adversary's ability to learn the first keyshare. The formula ϕ is the conjunction of the following statements:

- $T'.\mathrm{RP}i \Leftrightarrow T_1.\mathrm{RP}i$ for $i \in \{1, \ldots, 7\}$
- $T'.\mathrm{B}i \Leftrightarrow T_1.\mathrm{B}i$ for $i \in \{1, 2\}$

– $T'.\mathrm{U}i \Leftrightarrow T_1.\mathrm{U}i$ for $i \in \{1,2,3\}$
– $T'.\mathrm{P}i \Leftrightarrow T_1.\mathrm{P}i$ for $i \in \{1,2,3\}$
– $T'.\mathrm{P}i \Leftrightarrow T_1.\mathrm{P}(i-7)$ for $i \in \{11,12\}$
– $T'.\mathrm{P14} \Leftrightarrow T_1.\mathrm{P6}$
– $T'.\mathrm{SS}i \Leftrightarrow T_1.\mathrm{SS}i$ for $i \in \{1,\ldots,13\}$
– (T'.K and T'.SS3) $\Leftrightarrow$ (T_1.K and T_1.SS3)
– (T'.K and T'.SS6) $\Leftrightarrow$ (T_1.K and T_1.SS6)
– (T'.P9 and T'.SS3) $\Leftrightarrow$ (T_1.SS20 and T_1.SS3)
– (T'.P9 and T'.SS6) $\Leftrightarrow$ (T_1.SS20 and T_1.SS6)

Sets of attacks against the relying party, browser and user have not changed, therefore the sets from the SplitKey system are equivalent to the corresponding sets in the first intermediate system. Also, certain attacks against the phone and the SplitKey server are the same in both systems as these are not connected to the changes we introduced in the first intermediate system.

Attacks involving the adversary learning one or both shares of the key and the signature are the most interesting ones. As the discussion of $\prec_{\mathrm{S}}$ above shows, we can disregard attacks where the adversary learns only one of the shares, as they are no more powerful than learning no shares at all. The last four equivalences above state that if an attack set $\mathbf{w}$ against T' allows the adversary to learn a first share (of the private key, or the signature) and a second share, then we match it with an attack set $\mathbf{w}'$ against T_1 where the adversary also learns the same first share and the same second share. The seriousness of these attacks is the same — $T'.\mathbf{w} \sim_{\mathrm{S}} T_1.\mathbf{w}'$, because the adversary obtains the same knowledge and interferes with the same operations. The difficulty is also the same, although the phone should be easier to attack than the Splitkey server — the set $\mathbf{w}$ contains either T'.SS3 or T'.SS6, hence $\mathbf{w}'$ contains either T_1.SS3 or T_1.SS6, but these are at least as difficult to perform as T_1.SS16 or T_1.SS20.

3.4 Second intermediate system

We change the system T_1 into the system T_2, where instead of computing a combined signature from two signature shares, the Splitkey server generates a single signature using a single private key. The private key is not encrypted with the user's PIN. Instead, the server verifies the correctness of the user's PIN received together with the challenge, by comparing the PIN received from the phone with one stored on the server side. The authentication process in the second intermediate system is depicted in Figure 4.

Weaknesses of the second intermediate system. Attacks against the relying party, browser, user, and phone are the same as for the first intermediate system.

Attacks against the Splitkey server

(SS1) Interfere with the establishment of the secure channel with the phone, thereby obtaining the capability to read and/or change messages sent over it

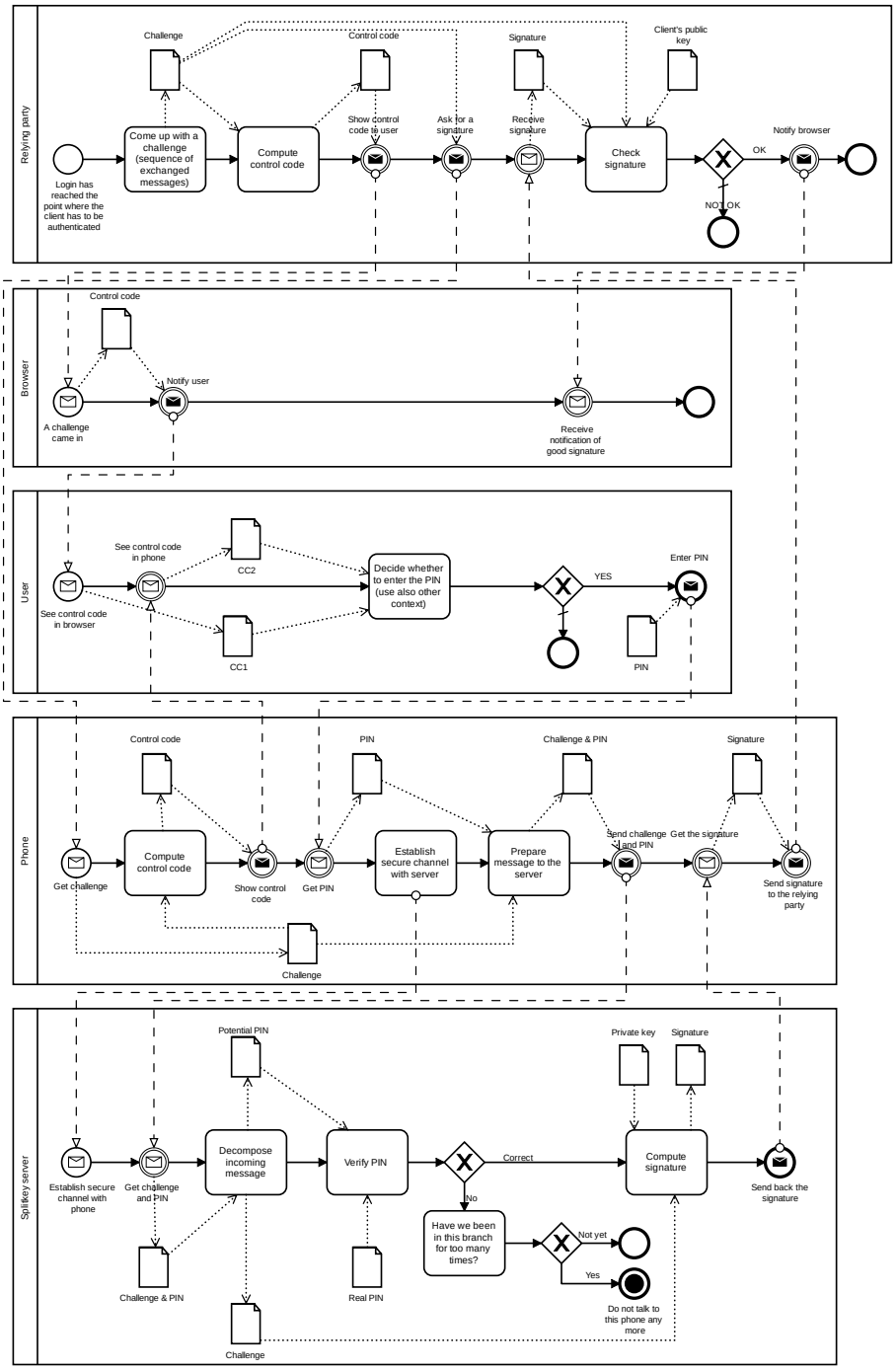

Fig. 4. The second intermediate system

(SS2) Interfere with the establishment of the secure channel with the phone, thereby confusing the server on the identity of the phone
(SS3) Learn the PIN received from the phone
(SS4) Interfere with the PIN verification process
(SS5) Make the decision of PIN check take the other path
(SS6) Make the decision about counts of incorrect PINs take the other path
(SS7) Learn private key
(SS8) Change private key
(SS9) Change the challenge that goes into the signature creation process
(SS10) Learn the signature
(SS11) Change the signature sent back to the phone

The second intermediate system is simpler than the first one because threshold cryptography is no longer used. We again have relationships between weaknesses of T_1 and T_2: we assume that learning the private key from the Splitkey server of T_2 is not harder than learning the second keyshare from the Splitkey server of T_1. We hence have $\{T_2.\text{SS7}\} \prec_\text{D} \{T_1.\text{SS3}\}$. We also have $\{T_1.\text{SS3}\} \prec_\text{S} \{T_2.\text{SS7}\}$: learning the whole private key definitely has at least as serious consequences as learning only one keyshare.

Comparison between T_1 and T_2. We can now write down the formula ϕ that matches each set of attacks against T_1 with a set of attacks against T_2. It is the conjunction of the following statements:

- $T_1.\text{RP}i \Leftrightarrow T_2.\text{RP}i$ for $i \in \{1, \ldots, 7\}$
- $T_1.\text{B}i \Leftrightarrow T_2.\text{B}i$ for $i \in \{1, 2\}$
- $T_1.\text{U}i \Leftrightarrow T_2.\text{U}i$ for $i \in \{1, 2, 3\}$
- $T_1.\text{P}i \Leftrightarrow T_2.\text{P}i$ for $i \in \{1, \ldots, 7\}$
- $T_1.\text{SS}i \Leftrightarrow T_2.\text{SS}j$ for $(i, j) \in \{(1, 1), (2, 2), (14, 3), (10, 4), (11, 5), (13, 6), (8, 10), (12, 11)\}$
- ((((T_1.SS14 and T_1.SS15) or T_1.SS16 or T_1.SS20) and (T_1.SS3 or T_1.SS6)) $\Leftrightarrow$ T_2.SS7
- (T_1.SS4 or T_1.SS17 or T_1.SS18) $\Leftrightarrow$ T_2.SS8
- (T_1.SS5 or T_1.SS19 or T_1.SS7) $\Leftrightarrow$ T_2.SS9

Sets of attacks against the relying party, browser, user and phone have not changed as we have not introduced changes to these parts of the system. Additionally, certain attacks against the SplitKey server are the same in both systems as these are not related to the changes we introduced in T_2. In T_2, we have a single private key that is stored on the SplitKey server. It means that we can match a combination of attacks that retrieve keyshares from T_1 against a single attack in T_2, where the adversary extracts the private key from the SplitKey server. Similarly, attacks changing challenges, keyshares, or signature shares, or interfering with the combination of signatures in T_1 are no worse than the attack changing the challenge, key, or signature in T_2.

In T_1, the PIN is used to decrypt the first keyshare at the server, and only a single PIN can make the following signature check succeed. Hence it makes sense

to state that the attacks T_1.SS10 on interfering with the signature check, and T_1.SS11 on subsequently changing the control flow, are the same as the attacks T_2.SS4 and T_2.SS5 that interfere with the comparison of PINs and subsequent branching.

3.5 Comparing the second intermediate system and the smartcard system

The attacks against the smartcard system T have been described in Sec. 3.1. There is still a significant difference between T_2 and T, due to the use of different hardware components, but as we see below, we can overcome this by introducing and justifying $\prec_{\mathrm{D}}$-relationships among the attacks against T_2 and T.

Comparison between T_2 and T. The formula ϕ is the conjunction of the following statements:

- $T_2.\mathrm{RP}i \Leftrightarrow T.\mathrm{RP}i$ for $i \in \{1, 2\}$
- $T_2.\mathrm{RP}i \Leftrightarrow T.\mathrm{RP}(i-2)$ for $i \in \{5, 6, 7\}$
- $T_2.\mathrm{U}i \Leftrightarrow T.\mathrm{U}i$ for $i \in \{1, 2\}$
- $T_2.\mathrm{P}i \Leftrightarrow T.\mathrm{SCR}j$ for $(i, j) \in \{(3, 1), (5, 2), (7, 3), (6, 4)\}$
- $T_2.\mathrm{SS}i \Leftrightarrow T.\mathrm{SC}(i-2)$ for $i \in \{3, \ldots, 11\}$
- $T_2.\mathrm{P4}$ or $T_2.\mathrm{SS1} \Leftrightarrow T.\mathrm{SCR1}$ and $T.\mathrm{SCR2}$ and $T.\mathrm{SCR3}$ and $T.\mathrm{SC9}$

Most of the attacks against the relying party and user are the same in both systems. One difference comes from the fact that smartcard system does not have Control Codes, thus, the attacks against them in T_2 are vacuously successful in T. We matched a set of attacks against the phone in T_2 against similar attacks targeting smartcard reader in the smartcard system. We have not matched T_2.RP3, T_2.RP4, T_2.B1, T_2.B2, T_2.P1, and T_2.P2 with anything in T as all of these attacks are related to the Control Codes and the smartcard system does not have this protection mechanism employed.

The set of attacks against the SplitKey server in T_2 is almost identical to the set of attacks against the smartcard in the smartcard system, except attacks related to the communication channel. However, the SplitKey server and smartcard have different hardware, hence we need to introduce the relationships $\{T.\mathrm{SC}i\} \prec_{\mathrm{D}} \{T_2.\mathrm{SS}(i+2)\}$ for the analysis to go through. Also, the attacks related to the communication channel in T_2 are matched with the set of attacks where the adversary learns and changes the values sent over that channel (e.g. PIN, challenge), and this again requires extra $\prec_{\mathrm{D}}$-relationships. These requirements on $\prec_{\mathrm{D}}$ are discussed below.

3.6 Propagation of the security requirements

We described which attacks are equivalent to each other in the neighbouring systems. Now, we analyse these to identify security requirements that should be

in place in the SplitKey system for the above listed attacks to be equivalent to each other.

The communication channel between the phone and SplitKey server must offer the same level of protection as smartcard system offers against SCR1, SCR2, SCR3, SC9 attacks. This requirement arises when we move from the second intermediate system to the smartcard system, as attacks against the communication channel are removed at this point. However, we identified that attacks against the communication channel are equivalent to the attacks against the smartcard system that involve learning and modifying information sent to the smartcard. The requirement of protecting the communication channel propagates through T_2 and T_1 back to T'.

We introduced the relationship $\{T.\text{SC5}\} \prec_{\text{D}} \{T_2.\text{SS7}\}$, requiring that the Splitkey server protects the private key at least as strongly as the smartcard. The same requirement holds for T_1, except that there is no private key in T_1. We can verify that it will be sufficient to strongly protect only the second keyshare: $\{T_2.\text{SS7}\} \prec_{\text{D}} \{T_1.\text{SS3}\}$. The same requirement propagates to T'.

It should be as hard to interfere with the signature verification process in the SplitKey server as it is to interfere with the PIN verification in the smartcard. There are attacks targeting the PIN verification process in the smartcard system and in T_2. However, T_1 and T' have attacks against signature verification. Signature verification in the SplitKey system serves to verify whether the user correctly decrypted their part of the private key to create their signature share, meaning that the user entered the correct PIN code. In the smartcard system, the PIN is verified straightforwardly by comparing the code stored in the card with the one received from the user.

It should be as hard to interfere with the signature creation/combination process in the SplitKey server as it is in the smartcard system. In the smartcard system and in T_2, the adversary can modify the challenge that enters the computation of the signature. In T_1 and T', the adversary can modify challenges for two signature shares — on the phone side and on the server side. If the SplitKey server receives an incorrect signature share from the phone and decides not to discard it but use it further, it means that it can successfully execute attacks against the signature verification process, that were covered in the previous requirement. Therefore, in this requirement, we focus on the server side attacks, where the adversary can not only change the challenge for the second signature share but also interfere with the signature combination process.

It should be as hard to change the server's private key share that is used to create the signature in the Splitkey server as it is in the smartcard system. In the smartcard system and T_2, adversary can attempt to change the single private key that will be used to create signature. In T_1 and T', the adversary may target two shares of the private key. With the modified user's share of the private key, adversary will create a signature share that will not pass the verification procedure. If the server decides to use incorrect signature share, it means that one can execute attacks against signature verification that were covered before. Therefore, this requirement again focuses on the modifications on the server side.

It should be as hard to capture the PIN entered by the user from the phone as it is to capture the PIN from the smartcard reader. Starting from the smartcard system, where the PIN is sent to the smartcard through the smartcard reader this requirement propagates all the way back to the SplitKey system, where the user enters the PIN in the phone. It may be difficult to argue that this requirement is satisfied, if a smartcard reader with PINpad is used in T. It will be easier to argue for it, if computer's keyboard is used in T.

4 Conclusion

We have presented the case that comparison-based arguments can be useful for providing the evidence that a system satisfies certain security properties, and, in particular, that a ToE satisfies a Security Target. It remains to be determined, how scalable the proposed methodology is – how much effort this methodology will require in different practical applications for evolving and novel ToEs, and what kind of tool support will be useful. The proposed methodology also needs evaluation by certification laboratories.

In our example, we have compared authentication systems. We hope that it is possible to similarly show that a threshold cryptography based system may be a qualified electronic signature creation device (QSCD) [6], by again comparing it to systems making use of smartcards.

References

1. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer (2006)
2. Buldas, A., Kalu, A., Laud, P., Oruaas, M.: Server-Supported RSA Signatures for Mobile Devices. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10492, pp. 315–333. Springer (2017)
3. Buldas, A., Saarepera, M.: Electronic Signature System with Small Number of Private Keys. In: Ellison, C.M., Polk, W.T., Hastings, N.E., Smith, S.W. (eds.) NISTIR 7085: 2nd Annual PKI Research Workshop Proceedings, pp. 110–122. National Institute of Standards and Technology (NIST) (2004)
4. De Santis, A., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely. In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC '94). p. 522–533. Association for Computing Machinery, New York, NY, USA (1994)
5. Dupont, S., Ginis, G., Malacario, M., Porretti, C., Maunero, N., Ponsard, C., Massonet, P.: Incremental Common Criteria Certification Processes using DevSecOps Practices. In: IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Vienna, Austria, September 6-10, 2021. pp. 12–23. IEEE (2021)

6. European Parliament and Council of European Union: Regulation (EU) no 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. OJ L 257, 28.8.2014, p. 73-114 (2014)
7. Hernandez-Ardieta, J.L., Blanco, P., Vara, D.: A methodology to construct Common Criteria security targets through formal risk analysis. In: Proceedings of XII Spanish Meeting on Cryptology and Information Security (RECSI 2012) (2012)
8. ISO/IEC 15408-1/2/3:2005 - Information technology — Security techniques — Evaluation criteria for IT security
9. Keblawi, F., Sullivan, D.: Applying the Common Criteria in Systems Engineering. IEEE Security and Privacy **4**(2), 50–55 (2006)
10. National Computer Security Center: Rating Maintenance Phase Program Document Version 2. Rainbow Series, NCSC-TG-013 V2 (1995), https://web.archive.org/web/20110720184904/http://iaarchive.fi/Rainbow/NCSC-TG-013%20PINK%20version%202.pdf
11. OMG: Business Process Model and Notation (BPMN), http://www.omg.org/spec/BPMN/2.0/
12. PP-Module for User Authentication Devices, Version 1.0. National Information Assurance Partnership (Jul 2019)
13. prEN 14169-1:2009: Protection profiles for Secure signature creation device — Part 2: Device with key generation. Technical Committee CEN/TC 224 (Dec 2009)
14. Pullonen, P., Matulevičius, R., Bogdanov, D.: PE-BPMN: privacy-enhanced business process model and notation. In: Carmona, J., Engels, G., Kumar, A. (eds.) Business Process Management - 15th International Conference, BPM 2017, Barcelona, Spain, September 10-15, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10445, pp. 40–56. Springer (2017)
15. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Aug 2018). https://doi.org/10.17487/RFC8446, https://www.rfc-editor.org/info/rfc8446
16. Reuse of Evaluation Results and Evidence (Oct 26th 2002), information Statement on behalf of the Common Criteria Recognition Arrangement Management Committee, Document no. 2002-08-009-002
17. Salnitri, M., Dalpiaz, F., Giorgini, P.: Designing secure business processes with SecBPMN. Softw. Syst. Model. **16**(3), 737–757 (2017)
18. Sinnhofer, A.D., Raschke, W., Steger, C., Kreiner, C.: Evaluation paradigm selection according to Common Criteria for an incremental product development. In: Tverdyshev, S. (ed.) International Workshop on MILS: Architecture and Assurance for Secure Systems, MILS@HiPEAC 2015, Amsterdam, The Netherlands, January 20, 2015. Zenodo (2015)
19. Sinnhofer, A.D., Raschke, W., Steger, C., Kreiner, C.: Patterns for Common Criteria Certification. In: Link, C., Eloranta, V. (eds.) Proceedings of the 20th European Conference on Pattern Languages of Programs, EuroPLoP 2015, Kaufbeuren, Germany, July 8-12, 2015. pp. 33:1–33:15. ACM (2015)
20. Sun, N., Li, C., Chan, H., Le, B.D., Islam, M.Z., Zhang, L.Y., Islam, M.R., Armstrong, W.: Defining security requirements with the common criteria: Applications, adoptions, and challenges. IEEE Access **10**, 44756–44777 (2022)

Comparing security levels of postal and Internet voting

Jelizaveta Vakarjuk[a], Nikita Snetkov[a], and Jan Willemson[a]

[a] Cybernetica, Mäealuse 2/1, 12618 Tallinn

ARTICLE HISTORY
Compiled May 22, 2024

ABSTRACT
This paper formalises the claim that security properties of Internet voting are stronger than those of paper-based postal voting. For that, we make use of a sequence-of-games approach common in the cryptographic research. We build BPMN models of both of the systems, presenting explicit reductions and intermediate games. Our comparison is based on the example of Estonian IVXV i-voting system and postal voting available for the expatriates.

1. Introduction

Voting is a well-established method of collective decision-making, with its roots being tracked back as far as ancient Greece Boegehold (1963). Even though the general problem setting of voting is simple, its implementation has turned out to be challenging. In case of political elections, the power to take impactful decisions is distributed. Hence, fairness of the whole process has to be ensured in order to guarantee that this power is vested in the representatives truly preferred by the society. The problem is, fairness is a notion having many dimensions.

On one hand, we want the elections to satisfy *generality* and *uniformity* requirements (every eligible voter can cast one vote, and only one). In order to guarantee these properties, we have to make sure that no votes are unduly discarded, but at the same time there should be no extra votes entering the tally. On the technical implementation level, such guarantees are provided by a number of security measures. In the polling stations, voter identities are verified, and several physical and organisational security mechanisms are put in place to prevent ballot box stuffing.

At the same time, we also want to guarantee voter's freedom in his/her decision. In physical polling stations, this property is typically achieved via secret voting, implemented by putting polling booths in place.

However, on-site polling station voting also has its drawbacks. It requires all the voters to come to a single geographical point in a short period of time. In the era of globalized world with highly mobile people, this is less and less of an option. Recent UN Migration report states that in 2020 there were 281 million migrants internationally, constituting 3.6 per cent of the world's population. This number has been growing steadily during the last decades *World Migration Report 2022* (n.d.). Also, during the COVID-19 pandemic, large-scale gatherings of people were discouraged, implying

CONTACT Jan Willemson. Email: jan.willemson@gmail.com

significant challenges for election organisers Cotti, Engelhardt, Foster, Nesson, and Niekamp (2021); Landman and Splendore (2020).

Thus, other voting methods allowing to cast votes remotely are required. Currently, there are two major alternatives – postal and Internet voting Krimmer, Duenas-Cid, and Krivonosova (2021). Even though some form of remote voting apparently took place already in the Roman Empire, systematic postal voting can be tracked back to late 19th century USA, New Zealand and Australia Abeels (2021).

Internet voting, also referred to as i-voting, is of course much more recent. However, in the last twenty years, several countries including Australia Brightwell, Cucurull, Galindo, and Guasch (2015); Conway and Teague (2022), Canada Goodman, Spycher-Krivonosova, Essex, and Brunet (2023), Estonia Ehin, Solvak, Willemson, and Vinkel (2022a), France Blanchard, Gallais, Leblond, Sidhoum-Rahal, and Walter (2022); Debant and Hirschi (2023), Norway Gebhardt Stenerud and Bull (2012); Gjøsteen (2012), Russia Gaudry and Golovnev (2020); Vakarjuk, Snetkov, and Willemson (2022) and Switzerland Haines, Pereira, and Teague (2022) have had elections with possibility to cast votes via Internet.

Both of these approaches have their own merits and shortcomings. Postal voting is easily accessible to anyone with basic literacy skills and access to a postal service. Internet voting, on the other hand, allows for better vote secrecy protection for the vote in transit, and enables stronger identity verification.

This paper aims at formalising the latter intuition and formally comparing the security properties of postal and Internet voting. Our first contribution is constructing detailed models using Business Process Modeling Notation (BPMN) of the vote casting phase for the i-voting and postal paper voting based on the case of Estonia. Estonia is the country with the longest continuous practice of i-voting for political elections, and this practice is very well documented, allowing for a fine-grained analysis.

Our main contribution is in constructing the reductions from the i-voting process to the postal voting through the sequence of intermediate systems, following the approach of game-based security proofs. Our reduction aims to show that the security properties of the i-voting system are stronger compared to those of the postal voting.

2. Related work

There is an extensive body of research on the security of remote electronic voting (see e.g. Bannet, Price, Rudys, Singer, and Wallach (2004); Moynihan (2004a, 2004b); Teague (2021); Vegas and Barrat (2016)), but also postal voting Abeels and Pereira (2021); Benaloh (2021); Luechinger, Rosinger, and Stutzer (2007). As Estonia is still the prime example of a country using Internet voting for political elections, its case has received extra treatment in the literature Heiberg, Laud, and Willemson (2011); Müller (2022); Pereira (2022); Springall et al. (2014); Sutopo, Haines, and Roenne (2024).

There are also a number of attempts to propose a methodology for the comparative analysis of voting systems. Krimmer and Volkamer Krimmer and Volkamer (2005) introduced a catalogue of criteria which could be used as a framework to compare different remote voting systems from legal, technology and security point of view. They applied this catalogue to compare postal and e-voting procedures used to elect chairman the chairman of the German Informatics Society in 2004. Willemson Willemson (2018) provided analysis and comparison of different security aspects of paper-based voting and e-voting. Musial-Karg Musial-Karg (2016) studied polling station, postal

and electronic voting focusing usability and their affect on democracy though omitting the security aspect. Li *et al.* developed a taxonomy to analyze the security requirements set for remote voting schemes Li, Kankanala, and Zou (2014). Neumann *et al.* constructed an evaluation framework for Internet voting schemes that enables to incorporate the expertise of system analysts and election officials Neumann, Noll, and Volkamer (2017). Juvonen composed a very thorough framework for comparing security levels of different voting schemes based on various dimensions of requirements Juvonen (2019). Krips *et al.* attempted to compare security levels of voting systems in terms of sets of trust assumptions Krips, Snetkov, Vakarjuk, and Willemson (2023).

As the number of requirement dimensions is very high for voting, all of the above approaches typically end up concluding that the security levels of specific voting systems are incomparable. Our aim is to develop a methodology allowing to establish a strict relationship between two modes of voting. Of course, this methodology can not be expected to be universal as the multi-dimensionality problem is inherent to voting as a general problem setting. However, we will show that for the voting modes of our choice (Internet and postal), such a formal comparison can be completed.

3. Methodology

For the comparison of voting systems we adapt the methodology from Laud *et al.* Laud and Vakarjuk (2022), which can be divided into following steps:

(1) construct models of the systems that need to be compared;
(2) identify the main data elements, processes and decision points that can be targeted by the adversary;
(3) construct lists of the system weaknesses by analysing the identified data elements, processes and decision points in the system;
(4) identify the relations between the listed weaknesses;
(5) compare the sets of weaknesses for both systems.

First, we construct the models of the systems we want to compare. In this work, we are using Business Process Model and Notation (BPMN) to create detailed models of the analysed voting systems. The BPMN diagrams allow modeling the steps of the voting process in details, depicting visually the sequence of activities performed by the protocol participants. Additionally, BPMN diagrams model the information flows that are needed to complete the process, while making it explicit which data elements are used for which processes. Based on the constructed models, we identify all the information components (data elements), processes and decision points that can be targeted by the adversary.

We denote i-voting system as $S_{\mathsf{i\text{-}voting}}$ and postal-voting system as $\mathcal{S}_{\mathsf{postal\text{-}voting}}$. Next, we create a list of weaknesses of the system $\mathcal{W}_{\mathsf{i\text{-}voting}}$ for the i-voting and $\mathcal{W}_{\mathsf{postal\text{-}voting}}$ for the postal voting by analysing each identified data element in the storage, transmission and processing steps. Additionally, we analyse during which processes and computations the adversary may get access to or tamper with the data elements. We also analyse all the decision points from the process model, and identify how the adversary can interfere with the decision making process. The adversary may either exploit the identified weaknesses fully or not exploit them at all. The weaknesses are grouped in the sets based on which party's process or data element is targeted. Table 1 presents our classification for the types of attacks adversary may perform targeting system weaknesses. It also illustrates how the identified attacks are related to the main security requirements for the voting systems.

Table 1. Attack classification

Attack type	Explanation	Targeted property
Read, learn	Reading/learning a data element that is not supposed to be learned	Vote secrecy, authenticity
Modify	Modifying a data element or interfering with the computations involving that data element, resulting in modifications of the data	Vote integrity, ballot box integrity
Delete, delay	Prevent a data element from being delivered to the destination	Ballot box integrity
Create, forge, replace	Introduce new adversary-provided data elements into the system	Ballot box integrity, authenticity

Next, we analyse the weakness sets and identify relations between them. There are two types of relations that we consider:

- $\prec_{\mathrm{S}}$ compares *seriousness* of a successful attack. For the weakness sets $\mathbf{w}_1$ and $\mathbf{w}_2$, we write $\mathbf{w}_1 \prec_{\mathrm{S}} \mathbf{w}_2$ to denote that the effects of exploiting all the weaknesses in $\mathbf{w}_1$ are no worse than the effects of exploiting $\mathbf{w}_2$.
- $\prec_{\mathrm{D}}$ compares *difficulty* of performing attack. For the weakness sets $\mathbf{w}_1$ and $\mathbf{w}_2$, we write $\mathbf{w}_1 \prec_{\mathrm{D}} \mathbf{w}_2$ to denote that exploiting all the weaknesses in $\mathbf{w}_1$ is no more difficult than exploiting all weaknesses in $\mathbf{w}_2$.

To find the relations between the weaknesses, we start by identifying what is the result of exploiting the weakness. The next step is to find another weakness or combination of weaknesses exploitation of which leads to the same results.

Our final goal is to compare sets of weaknesses for both systems $S_{\mathsf{i\text{-}voting}}$ and $S_{\mathsf{postal\text{-}voting}}$ to show that for each weakness $\mathbf{w}' \in \mathcal{W}_{\mathsf{postal\text{-}voting}}$, there exists some $\mathbf{w} \in \mathcal{W}_{\mathsf{i\text{-}voting}}$, such that $\mathbf{w} \prec_{\mathrm{S}} \mathbf{w}'$ and $\mathbf{w} \prec_{\mathrm{D}} \mathbf{w}'$. However, we are only able to identify relations $\prec_{\mathrm{D}}$ and $\prec_{\mathrm{S}}$ if the systems are similar enough in the processes and usage of data elements. If the systems are not sufficiently similar, we need to introduce intermediate systems, like it is done in the game-based security proofs for the cryptographic systems Shoup (2004). As is the case with its cryptographic counterpart, intermediate systems are not necessarily realistic or deployable in the real-life. The goal of introducing them is to arrive from one system to another through changes small enough to allow for immediate analysis. We also can identify security requirements and mechanisms that should be in place for the systems $S_{\mathsf{i\text{-}voting}}$ and $S_{\mathsf{postal\text{-}voting}}$ to be comparable.

4. Comparison

In this work, we focus on comparing *vote casting phase* of i-voting to the postal voting. Our comparison is based on the Estonian voting process as Estonia is a country with an established i-voting system in operation since 2005. We refer to Ehin *et al.* for some historic background and a general overview of the system Ehin, Solvak, Willemson, and Vinkel (2022b).

4.1. I-voting

In the Estonian i-voting system (code-named IVXV), the voter (V) uses a dedicated *voting application* (VA) to cast the vote. The voter is first required to authenticate

him/herself through the application. Then the voter makes the choice, which is first encrypted under the system's public key and then digitally signed with the voter's *signing device* (SD). IVXV currently supports signing with a smartcard-based eID, but in our model we decided to make digital signature creation more general, requiring the voter to have a *knowledge factor* (that can be a PIN code) and a *possession factor* (that can be a smartcard with the signing key). The voting application then generates and displays a QR code that the voter can use to verify the vote using the *verification application* (VRA) Heiberg and Willemson (2014).

The votes are sent to the *vote collector* (VC) server that is responsible for storing them in the Vote database (Vote DB) before the vote casting phase ends. Vote collector interacts with the Online Certificate Status Protocol (OCSP) service to get information about the voter's public key validity, and *registration service* (RS) to obtain timestamps on the received votes. Registration service stores generated timestamps in the Timestamp database (Timestamp DB). Estonian i-voting system allows voters to recast their vote as an anti-coercion measure, but in the current comparison we omitted this functionality as it is challenging to model the relation between attack sets if the adversary may get access to different data elements in different vote casting processes performed by the same voter. The general diagram of the i-voting processes is presented in Appendix A, Figure A1.

4.1.1. Weaknesses of the i-voting system

We now list sets of weaknesses that the adversary, corrupting the main participants, may exploit. We consider different levels of corruption while listing the weaknesses, including full corruption and being able to modify some data elements in the system. We note that some of the listed weaknesses may be impossible to exploit within a meaningful timeframe. *Interfering* with some process refers to the adversary changing the values that are involved in the process (both inputs and outputs) – encryption key, verification key, PIN code, etc. *To take another path* refers to the weakness in verification procedures, where the adversary makes the verification process accept even if it should not, and vice versa.

The following weaknesses (forming the set $\mathcal{W}_{\text{i-voting}}$) were identified using our methodology.

4.1.1.1. Possible attacks against the voter.

(V1) Learn the voter's choice;
(V2) Modify the voter's choice;
(V3) Stop the voter from casting their choice;
(V4) Learn the knowledge factor from the voter;
(V5) Modify the knowledge factor that is sent to the signing device;
(V6) Interrupt the voter during vote verification process.

4.1.1.2. Possible attacks against the signing device.

(SD1) Learn the knowledge factor;
(SD2) Get access to possession factor;
(SD3) Interfere with the knowledge factor verification procedure;
(SD4) Make the decision of the knowledge factor verification to take another path;
(SD5) Learn the private key;

(SD6) Modify the private key;
(SD7) Learn the challenge;
(SD8) Modify the challenge that enters the computation of the signature;
(SD9) Learn the signature;
(SD10) Modify the signature sent back to the smartcard reader.

4.1.1.3. Possible attacks against the voting application.

(VA1) Learn the voter's identity;
(VA2) Interfere with the ballot generation;
(VA3) Modify blank ballot that is displayed to the voter;
(VA4) Learn the voter's choice;
(VA5) Interfere with the processing of the voter's choice;
(VA6) Interfere with the encryption process (e.g. modify encryption randomness, encryption key, plaintext);
(VA7) Learn the encryption randomness;
(VA8) Learn the ciphertext (encrypted ballot);
(VA9) Learn the signature;
(VA10) Modify the signature received from the signing device;
(VA11) Modify the vote that is sent to the VC;
(VA12) Interfere with the OCSP response verification process;
(VA13) Make the decision of the OCSP response verification to take another path;
(VA14) Interfere with the RS's signature verification process;
(VA15) Make the decision of the RS's signature verification to take another path;
(VA16) Interfere with process of recomputing hash;
(VA17) Interfere with the hash correctness verification process;
(VA18) Make the decision of the hash correctness verification to take another path;
(VA19) Interfere with the VC's signature verification process;
(VA20) Make the decision of the VC's signature verification to take another path;
(VA21) Interfere with the QR code generation process;
(VA22) Display a modified QR code;
(VA23) Learn the QR code;
(VA24) Interfere with the establishment of secure channel with VC, obtaining the capability to read/modify/drop messages sent over the channel;
(VA25) Do not to send the vote to VC.

4.1.1.4. Possible attacks against the vote collector.

(VC1) Learn the voter's identity from the received vote;
(VC2) Interfere with the voter eligibility verification process;
(VC3) Make the decision of the voter eligibility verification to take another path;
(VC4) Interfere with the vote format verification process;
(VC5) Make the decision of the vote format verification to take another path;
(VC6) Interfere with the signature verification process;
(VC7) Make the decision of the signature verification to take another path;
(VC8) Interfere with the OCSP response verification process;
(VC9) Make the decision of the OCSP response verification to take another path;
(VC10) Delete the OCSP response;
(VC11) Interfere with the qualified signature combining process;
(VC12) Interfere with the vote ID generation;

(VC13) Learn the VC's private key;
(VC14) Modify the VC's private key;
(VC15) Interfere with the challenge generation;
(VC16) Modify the registration request that is sent to the RS;
(VC17) Decide not to send the registration request to the RS;
(VC18) Interfere with the establishment of secure channel with the RS, obtaining the capability to read/modify/drop messages sent over the channel;
(VC19) Modify the timestamp received from the RS;
(VC20) Delete the timestamp;
(VC21) Modify a record in the in Vote DB;
(VC22) Delete a record from the Vote DB;
(VC23) Add a forged record to the Vote DB;
(VC24) Decide not to notify the VA that the vote is received;
(VC25) Modify the confirmation information sent to the VA;
(VC26) Interfere with the process of finding the correct vote in the Vote DB;
(VC27) Modify the vote that is sent to the VA.

4.1.1.5. Possible attacks against the registration service.

(RS1) Modify the registration request received from the VC;
(RS2) Interfere with the VC's signature verification process;
(RS3) Make the decision of the VC's signature verification to take another path;
(RS4) Interfere with the vote registration process;
(RS5) Learn the RS's private key;
(RS6) Modify the RS's private key;
(RS7) Modify the challenge that enters signature computation;
(RS8) Interfere with the timestamp combination process;
(RS9) Modify a timestamp that is stored in the Timestamp DB;
(RS10) Modify a timestamp that is sent to the VC;
(RS11) Add a forged timestamp to the Timestamp DB;
(RS12) Delete a timestamp from the Timestamp DB.

4.1.1.6. Possible attacks against the verification application.

(VRA1) Interfere with the QR code scanning process;
(VRA2) Query a modified vote ID from the VC;
(VRA3) Interfere with the establishment of a secure channel with the VC, obtaining the capability to read/modify/drop messages sent over the channel;
(VRA4) Interfere with the vote format verification process;
(VRA5) Make the decision of the vote format verification to take another path;
(VRA6) Interfere with the signature verification process;
(VRA7) Make the decision of the signature verification to take another path;
(VRA8) Interfere with the OCSP response verification process;
(VRA9) Make the decision of the OCSP response verification to take another path;
(VRA10) Interfere with the VC's signature verification process;
(VRA11) Make the decision of the VC's signature verification to take another path;
(VRA12) Interfere with the RS's signature verification process;
(VRA13) Make the decision of the RS's signature verification to take another path;
(VRA14) Modify the vote received from the VC;
(VRA15) Interfere with the restoration of the plaintext;

(VRA16) Display a wrong plaintext to the voter;
(VRA17) Learn the voter's choice.

4.1.2. Relations between the attacks

Some of the identified attacks are tightly connected to each other. For example, an attack may not be serious on its own, but, when combined with other attacks, results in exploitation of a critical weakness. Another example of a relation between the attacks is that several different attacks, when successful, are leading to the same results. Therefore, we need to analyse which attacks are related to each other, easing the comparison of the intermediate systems.

First, we notice that changing a data element has no greater effect than changing the other data elements that are computed from this value. For example, interfering with the encryption process leads to a modified encrypted ballot. Therefore, exploiting the weakness "interfere with encryption process" (VA6) is at least as serious ($\prec_S$) as exploiting the weakness "modify vote" (VA11), because the vote contains the encrypted ballot. We denote this relation as VA6 $\prec_S$ VA11. Similarly, we identify the following relations:

- The attacks V2, VA5 result in a modified voter's choice and can be related by V2 $\prec_S$ VA5.
- The attacks VA21, VA22, VRA1 result in a modified QR code and can be related by VA21 $\prec_S$ VA22 $\prec_S$ VRA1.
- The attacks SD6, SD8, SD10, VC6, VC7 result in a modified (voter's) signature (that may not pass verification) and can be related by (SD6 or SD8) $\prec_S$ SD10 $\prec_S$ VC6 $\prec_S$ VC7.
- The attacks VC14, VC15, RS2, RS3 result in a modified VC's signature (that may not pass verification) and can be related by (VC14 or VC15) $\prec_S$ RS2 $\prec_S$ RS3.
- The attacks RS6, RS7, RS8, VRA12, VRA13 result in a modified RS's signature (that may not pass verification) and can be related by (RS6 or RS7) $\prec_S$ RS8 $\prec_S$ VRA12 $\prec_S$ VRA13.
- The attacks V5, SD3, SD4 result in a modified knowledge factor (that may not pass the verification process) and can be related by V5 $\prec_S$ SD3 $\prec_S$ SD4
- The attacks V3, VA25, VC22 result in not having the voter's choice included in the ballot box and can be related by V3 $\prec_S$ VA25 $\prec_S$ VC22.

However, modifying an encrypted ballot or a signature does not always result in a new valid ciphertext (that could be decrypted to a meaningful ballot) or a valid signature (that passes the verification process). The adversary may try to exploit those weaknesses not only to change the voter's choice, but also to invalidate the vote. If the signature verification on the modified encrypted ballot does not pass, the vote will not be counted in the final tally.

Now, let us look at different types of relations, where several attacks can be combined into a more serious one.

- The encrypted ballot can be decrypted using the encryption randomness and the public key. The resulting relation is (VA23 or VA7) and VA8 $\prec_S$ VA4.
- If the adversary is able to obtain a digital signature and the encrypted ballot (that are used to construct the vote), they are able to learn the voter's identity. The resulting relation is (VA9 and VA8) $\prec_S$ VA1.
- Learning the voter's knowledge factor and obtaining access to the possession factor results in having access to the private (signing) key. The resulting relation

is (V4 or SD1) and SD2 $\prec_S$ SD5.

4.2. Postal voting

If the voter wishes to cast their vote from abroad, they may do so by means of postal voting. In this case, the voter is required to submit a written application to the *foreign mission* (FM) expressing their wish to cast a vote *Riigikogu Election Act* (14.12.2020). If the application gets approved, the voter is sent a ballot, a candidate list and two envelopes (inner envelope and outer envelope). The voter fills in the ballot and seals it into the inner envelope. The inner envelope is then placed into the outer one. Next, the voter writes their name and identification code on the outer envelope and sends it to the foreign mission through the *postal service* (PS). The foreign mission sends the received votes to the State Electoral Office (EO), who later forwards those to the designated vote counting committees. A detailed postal voting diagram is presented in Appendix A, Figure A2.

4.2.1. Weaknesses of the postal voting system

The following weaknesses (forming the set $\mathcal{W}_{\text{postal-voting}}$) have been identified using our methodology.

4.2.1.1. Possible attacks against the voter.

(V1) Learn the voter's choice;
(V2) Modify the voter's choice;
(V3) Stop the voter from casting their choice;
(V4) Create a signature (voter identification data) on behalf of voter;
(V5) Remove a signature (voter identification data) from the outer envelope;
(V6) Modify a signature (voter identification data) on the outer envelope;
(V7) Modify a blank ballot;
(V8) Modify the voter's choice on the filled ballot;
(V9) Modify an envelope.
(V10) Modify the candidate list.

4.2.1.2. Possible attacks against the postal service.

(PS1) Modify
  (a) a blank ballot,
  (b) a candidate list,
  (c) an envelope
  before it is delivered to the voter;
(PS2) Delay delivery of a package with the documents for voting;
(PS3) Open a package and modify
  (a) an envelope,
  (b) a filled ballot
  before it is delivered to the FM;
(PS4) Learn the voter's choice;
(PS5) Delay delivery of a package to the FM;
(PS6) Open a package and modify
  (a) an envelope,

(b) a filled ballot
before it is delivered to the EO;
(PS7) Delay delivery of a package to the EO;

4.2.1.3. Possible attacks against the foreign mission.

(FM1) Learn the voter's identity from the received vote;
(FM2) Interfere with the voter eligibility verification process;
(FM3) Make the decision of the voter eligibility verification to take another path;
(FM4) Open envelopes and reveal the vote;
(FM5) Learn the voter's choice;
(FM6) Modify the voter's choice;
(FM7) Remove (destroy) a physical envelope;
(FM8) Modify a physical envelope (or voter data on it);
(FM9) Create and store a forged physical envelope;
(FM10) Interfere with the process marking that the vote is received;
(FM11) Do not send a vote to the EO.

4.2.1.4. Possible attacks against the state electoral office.

(EO1) Modify received envelopes;
(EO2) Modify a filled ballot;
(EO3) Learn the voter's choice;
(EO4) Remove the ballot of a specific voter;
(EO5) Delay forwarding ballots to the vote counting committees.

Similarly to how we identified relationships between the attacks for the i-voting system, we identify the following attack relationships among the attacks against the postal voting system:

- The attacks V3, FM7, FM11, EO4 result in the voter's choice not being recordered in the system. The resulting relation is V3 $\prec_S$ FM7 $\prec_S$ FM11 $\prec_S$ EO4.

It is not possible to directly compare the i-voting system to the postal voting system as the processes are too different from each other, and we cannot construct enough $\prec_S$ and $\prec_D$ relationships between the weakness sets. Therefore, we describe three *intermediate systems*. We start from i-voting and introduce small changes to the system until we receive one that is closer to the postal voting. For each of the introduced intermediate systems we provide a comparison with the previous system, constructing relationships for $\prec_S$ and $\prec_D$ and identifying differences between the systems.

4.3. The first intermediate system

We change the i-voting system by removing the usage of digital signatures and replacing them with handwritten ones. To enable the voter to provide a handwritten signature, we introduce the following modification – once the vote is encrypted, the ciphertext is printed out on paper by the voter. The voter is then required to provide a handwritten signature, scan the signed ciphertext and upload the scanned document with the help of the voting application to the vote collector server.

There are two more places in the system where the digital signatures are generated – on the vote collector side and on the registration service side. Those are also replaced with handwritten signatures. The vote collector is required to sign a printed

registration request that is then scanned and sent to the registration service. The registration service signs the printed response (containing a timestamp) that is scanned and sent back to the vote collector. Since there is no clear verification procedure for the handwritten signatures, we replace it with a verifying party visually checking that the signature is in place. The corresponding diagram is presented in Appendix A, Figure A3.

4.3.1. Weaknesses of the first intermediate system

Below, we describe the modifications to the weakness sets that are introduced due to the changes in the system. There are no modifications to the set of **possible attacks against the registration service**. Additionally, we remove the attack set related to the signing device, as this component is removed from the model.

We remove the knowledge factor-related attacks from the set of **possible attacks against the voter**, since those are not applicable anymore. We add the following attacks:

($\mathsf{V}'1$) Interfere with the encrypted ballot printing process;
($\mathsf{V}'2$) Modify the signature on the printed encrypted ballot;
($\mathsf{V}'3$) Remove the signature from the encrypted ballot;
($\mathsf{V}'4$) Create a handwritten signature on behalf of the voter;
($\mathsf{V}'5$) Interfere with the vote scanning procedure.

We remove the attacks related to digital signatures from the set of **possible attacks against the voting application** as those are not applicable. We add the following attacks:

($\mathsf{VA}'1$) Make the decision on RS's signature presence to take another path;
($\mathsf{VA}'2$) Make the decision on VC's signature presence to take another path;

We remove the attacks related to digital signatures from the set of **possible attacks against the vote collector** as those are not applicable. We add the following attacks:

($\mathsf{VC}'1$) Make the decision on signature presence to take another path;
($\mathsf{VC}'2$) Modify the vote received from the VA;
($\mathsf{VC}'3$) Interfere with the vote hash printing procedure;
($\mathsf{VC}'4$) Modify the signature on the printed vote hash;
($\mathsf{VC}'5$) Remove the signature from the vote hash;
($\mathsf{VC}'6$) Create a handwritten signature on behalf of the vote collector;
($\mathsf{VC}'7$) Interfere with the signed vote hash scanning procedure.

We remove the attacks related to digital signatures from the set of **possible attacks against the registration service** as those are not applicable. We add the following attacks:

($\mathsf{RS}'1$) Make the decision on VC's signature presence to take another path;
($\mathsf{RS}'2$) Interfere with the request printing procedure;
($\mathsf{RS}'3$) Modify the signature on the printed request;
($\mathsf{RS}'4$) Remove the signature from the request;
($\mathsf{RS}'5$) Create a handwritten signature on behalf of the RS;
($\mathsf{RS}'6$) Interfere with the signed request scanning procedure.

We remove the attacks related to digital signatures from the set of **possible attacks against the verification application** as those are not applicable. We add

the following attacks:

(VRA′1) Make the decision on the voter's signature presence to take another path;
(VRA′2) Make the decision on the RS's signature presence to take another path;
(VRA′3) Make the decision on the VC's signature presence to take another path;

Similarly to the analysis that we performed on the attacks for the i-voting system, we identify the following relations between the attacks on the first intermediate system.

- The attacks $\mathsf{VA6}, \mathsf{V}'1, \mathsf{V}'5$ result in a modified ciphertext and can be related by $\mathsf{VA6} \prec_S \mathsf{V}'1 \prec_S \mathsf{V}'5$.
- The attacks $\mathsf{V}'2, \mathsf{V}'5, \mathsf{VC}'2$ result in a modified voter's signature and can be related by $\mathsf{V}'2 \prec_S \mathsf{V}'5 \prec_S \mathsf{VC}'2$.

4.3.2. Comparison between i-voting and the first intermediate system

We have two sufficiently similar systems – the i-voting system S and the first intermediate system S'. We now show that for each weakness for S there exists weakness in S' that is no more difficult to exploit and that has the same or worse consequences. We do it by constructing relations in a similar way as we constructed relations for the weaknesses within one system. Some of the weakness sets remain the same in both systems. We concentrate on the removed weaknesses that do not carry from S to S'. For those we either find a relation with some other weaknesses in S' or we argue that removing this weakness does not make the system S' more difficult to attack.

When constructing relations between the weaknesses in S and S', we use the relations that have been already identified in Section 4.1.2. If there are several weaknesses $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$ that lead to the same result in S, we take one of them $\mathbf{w}_1$ and map to the corresponding weakness $\mathbf{w}'$ of S' in the relations below. However, it means that the weaknesses $\mathbf{w}_2, \mathbf{w}_3$ are also mapped to $\mathbf{w}'$.

(1) $S.\mathsf{SD}5 \Leftrightarrow S'.\mathsf{V}'4$
(2) $S.\mathsf{SD}10, S.\mathsf{VA}10 \Leftrightarrow S'.\mathsf{V}'2$
(3) $S.\mathsf{V}5 \Leftrightarrow S'.\mathsf{V}'3$
(4) $S.\mathsf{VC}13 \Leftrightarrow S'.\mathsf{VC}'6$
(5) ($S.\mathsf{VC}14$ or $S.\mathsf{VC}15$) $\Leftrightarrow S'.\mathsf{VC}'4$
(6) $S.\mathsf{RS}5 \Leftrightarrow S'.\mathsf{RS}'5$
(7) ($S.\mathsf{RS}6$ or $S.\mathsf{RS}7$) $\Leftrightarrow S'.\mathsf{RS}'3$
(8) ($S.\mathsf{VA}14$ or $S.\mathsf{VA}15$) $\Leftrightarrow S'.\mathsf{VA}'1$
(9) ($S.\mathsf{VA}19$ or $S.\mathsf{VA}20$) $\Leftrightarrow S'.\mathsf{VA}'2$
(10) ($S.\mathsf{VRA}6$ or $S.\mathsf{VRA}7$) $\Leftrightarrow S'.\mathsf{VRA}'1$
(11) ($S.\mathsf{VRA}12$ or $S.\mathsf{VRA}13$) $\Leftrightarrow S'.\mathsf{VRA}'2$
(12) ($S.\mathsf{VRA}10$ or $S.\mathsf{VRA}11$) $\Leftrightarrow S'.\mathsf{VRA}'3$

The relations (1) and (2) are constructed from the observation that the attacks ($S.\mathsf{SD}5, S.\mathsf{SD}10$) that were targeting the signing device in S are targeting the voter in S' ($S'.\mathsf{V}'4, S'.\mathsf{V}'2$). When changing from digital signatures to the handwritten signatures, we have made it simpler to perform certain attacks. If the i-voting system is instantiated with a (UF-CMA) secure digital signature scheme, the adversary cannot create valid signature forgeries in feasible time. However, forging the handwritten signatures is an easier process – the attacker just needs to have an access to a handwritten signature example created by the voter, and can then clone it.

The relations (1), (4) and (6) are connected to forging signatures. Attacks targeting the private keys ($S.\mathsf{SD}5, S.\mathsf{VC}13, S.\mathsf{RS}5$) in S are transformed into the attacks where the adversary creates a handwritten signature on behalf of that party

($S'.\mathsf{V}'4, S'.\mathsf{VC}'6, S'.\mathsf{RS}'5$) in S'. Those attacks are equivalent since the adversary who has an access to the party's private key can create valid digital signatures on behalf of that party.

Modifying the private key or the challenge that enters the computation of the digital signature ($S.\mathsf{VC}14, S.\mathsf{VC}15, S.\mathsf{RS}6, S.\mathsf{RS}7$) leads to the modified digital signature. Relations (5) and (7) present the fact that those attacks from S are equivalent to the adversary directly modifying a handwritten signature in S' ($S'.\mathsf{VC}'4, S'.\mathsf{RS}'3$).

The relation (3) is connected to the attack targeting the knowledge factor sent to the signing device $S.\mathsf{V}5$. This attack is transformed into an attack that aims to remove the voter's signature from the encrypted ballot $S'.\mathsf{V}'3$. We consider these attacks equivalent as an incorrect knowledge factor will not pass the verification process inside the signing device and the signature will not be created, which is equivalent to not having a handwritten signature on the ballot.

The relations (8), (9), (10), (11) and (12) express the observation that the attacks that were targeting the signature verification procedure in S ($S.\mathsf{VA}14$, $S.\mathsf{VA}15$, $S.\mathsf{VA}19$, $S.\mathsf{VA}20$, $S.\mathsf{VRA}6$, $S.\mathsf{VRA}7$, $S.\mathsf{VRA}10$, $S.\mathsf{VRA}11$, $S.\mathsf{VRA}12$, $S.\mathsf{VRA}13$) are transformed into equivalent attacks against the handwritten signature, where the verifier just checks whether the signature is present or not ($S'.\mathsf{VA}'1, S'.\mathsf{VA}'2, S'.\mathsf{VRA}'1, S'.\mathsf{VRA}'3, S'.\mathsf{VRA}'2$). Therefore, if the adversary attempts to forge a handwritten signature, it will be harder to catch him with the new verification procedure.

The attacks targeting OCSP response and its verification in S ($S.\mathsf{VA}12$, $S.\mathsf{VA}13$, $S.\mathsf{VC}8$, $S.\mathsf{VC}9$, $S.\mathsf{VC}10$, $S.\mathsf{VRA}8$, $S.\mathsf{VRA}9$) are not applicable in S', since there is no service for the handwritten signatures that provides information about the signature validity. Removed attacks related to the OCSP service are not targeting the main security requirements of the voting system, such as vote secrecy and vote/ballot box integrity. Therefore the system S does not have any critical weaknesses that are not present in the system S'.

4.4. The second intermediate system

We change the first intermediate system S' by removing the usage of ballot encryption and verification processes. Encryption is replaced by using two envelopes, as it is done in the postal voting systems. There is an inner envelope that does not contain any identification data, and the outer envelope with the voter's handwritten signature. Since the system does not use encryption now, we also remove the verification mechanism as it was built around the ballot being encrypted. The vote is now delivered in physical form using postal services. Additionally, instead of the encrypted ballot, Vote DB contains a record with vote ID, Timestamp and scanned VC's signature on the outer envelope. Since we want to preserve the property that the voter gets some assurance about their vote being delivered, we do not remove the confirmation message that is sent to the voting application once the vote is recorded. The resulting system diagram is presented in Appendix A, Figure A4.

4.4.1. Weaknesses of the second intermediate system

We remove the attacks related to the modification of encrypted ballot from the set of **possible attacks against the voter** and add the following attacks:

($\mathsf{V}''1$) Interfere with the blank ballot printing;

($\mathsf{V}''2$) Modify the voter's choice on the filled ballot;
($\mathsf{V}''3$) Modify the signature on the outer envelope;
($\mathsf{V}''4$) Remove the signature from the outer envelope;
($\mathsf{V}''5$) Create a signature on the outer envelope on behalf of the voter;
($\mathsf{V}''6$) Modify an envelope.

We remove the attacks related to learning the voter's choice, encryption and verification mechanisms from the set of **possible attacks against the voting application**.

We add the following attacks to the set of **possible attacks against the vote collector**:

($\mathsf{VC}''1$) Open envelopes and reveal the vote;
($\mathsf{VC}''2$) Learn the voter's choice;
($\mathsf{VC}''3$) Modify the voter's choice;
($\mathsf{VC}''4$) Remove (destroy) an envelope;
($\mathsf{VC}''5$) Modify an envelope (or any information contained on it);
($\mathsf{VC}''6$) Modify vote ID that is stored in the Vote DB; ´
($\mathsf{VC}''7$) Create and store a forged envelope with the filled ballot;
($\mathsf{VC}''8$) Modify VC's signature on the outer envelope;
($\mathsf{VC}''9$) Remove VC's signature from the outer envelope;
($\mathsf{VC}''10$) Interfere with the signed outer envelope scanning process;

Additionally, we remove the attacks related to the encrypted ballot and interaction with the verification application.

There are no modifications in the set of **possible attacks against the registration service**. Additionally, we remove attack sets related to the verification application, as this component is removed from the model.

We can identify the following additional relations between the attacks on the second intermediate system:

- The attacks $\mathsf{VA}2, \mathsf{V}''2$ result in modified blank ballot and can be combined into a relation $\mathsf{VA}2 \prec_{\mathrm{S}} \mathsf{V}''2$.

4.4.2. Comparison between the first and the second intermediate systems

Let us denote the second intermediate system as S'', then there are the following relations between attack sets in S' and S'':

(1) $S'.\mathsf{VA}4 \Leftrightarrow S''.\mathsf{VC}''2$
(2) $S'.\mathsf{VA}5 \Leftrightarrow S''.\mathsf{VC}''3$
(3) $S'.\mathsf{V}'i \Leftrightarrow S''.\mathsf{V}''(i+1)$ for $i \in \{2, 3, 4\}$
(4) $S'.\mathsf{VC}'2 \Leftrightarrow S''.\mathsf{VC}''3$
(5) $S'.\mathsf{VC}'4 \Leftrightarrow S''.\mathsf{VC}''8$
(6) $S'.\mathsf{VC}'5 \Leftrightarrow S''.\mathsf{VC}''9$

The relations (1) and (2) express the observation that attacks against the voting application $S'.\mathsf{VA}4, S'.\mathsf{VA}5$ from S' are now targeting the vote collector in S'': $S''.\mathsf{VC}''2, S''.\mathsf{VC}''3$. The primary difference between S' and S'' is in replacing encryption with using two envelopes for vote transmission and storage. Therefore, it becomes easier for the adversary against the system S'' to perform the attacks targeting the filled ballot. When in S' an encrypted ballot was modified by the adversary, it was not guaranteed that the modified ciphertext decrypts to *a valid vote*. In the system S'', however, the adversary may modify the voter's choice directly by opening the envelopes and performing the modifications that will result in a valid vote with much higher probability. Therefore, in the system S'', attacks targeting the voter's choice

are much stronger.

The relation (3) is valid since the voter in the system S'' signs the outer envelope, not the encrypted ballot. Attacks targeting the voter's signature on the encrypted ballot ($S'.\mathsf{V}'2, S'.\mathsf{V}'3, S'.\mathsf{V}'4$) in the system S'' are equivalent to the attacks targeting the voter's signature on the outer envelope ($S''.\mathsf{V}''3, S''.\mathsf{V}''4, S''.\mathsf{V}''5$) in the system S''.

The relation (4) corresponds to the fact that there is no more encryption functionality in S'' and the attack $S.\mathsf{VC}'2$ targeting the vote received from the VA is transformed into a more powerful attack $S''.\mathsf{VC}''3$ directly targeting the voter's choice in S''.

Additionally, since the verification functionality is removed from the system S'', the voter has no chance to notice if there were any attacks targeting their vote, making all those attacks successful. All the attacks related to the verification application and its functionality are not applicable in S''. With those attacks we remove the voter's chance to detect malicious behaviour of the system and notice if their vote was changed. Among the removed attacks from S' that are not mapped to any attacks in S'' there are no attacks that directly threaten vote secrecy, vote/ballot box integrity, or vote authenticity. Removed unmapped attacks only target verification process of the system.

The mechanism of signing the vote hash by the VC in S' is replaced by requiring the vote collector to sign the outer envelope. Therefore, the relations (5) and (6) express the observation that the attacks targeting VC signature on the vote hash ($S'.\mathsf{VC}'4, S'.\mathsf{VC}'5$) in the system S' are transformed into attacks targeting VC signature on the outer envelope in S'' ($S''.\mathsf{VC}''8, S''.\mathsf{VC}''9$). All the remaining attacks related to the hashing mechanism and its verification are no longer applicable. By removing this mechanism, we also remove the voter's confidence that their vote is indeed registered by the vote collector, since in the system S'' the adversary may sign an arbitrary envelope, add signature on behalf of the voter and present the scanned version as a confirmation of the vote being recorded.

4.5. The third intermediate system

To construct the third intermediate system, we change the model S'' by requiring the EO to prepare and print out blank ballots and candidate lists, and deliver them to the voter together with the inner and outer envelopes. Additionally, we replace the RS by requiring the vote collector to mark in the Voter Listthat the vote has been received. The corresponding system diagram is presented in Appendix A, Figure A5. This model is already close to the postal voting model, the only missing part is a separate postal service party that is responsible for delivering the blank ballots and filled ballots.

4.5.1. Weaknesses of the third intermediate system

We remove the attacks related to the ballot printing from the set of the **possible attacks against the voter** and add:

($\mathsf{V}'''1$) Modify a blank ballot;
($\mathsf{V}'''2$) Modify the candidate list.

We introduce a new set of **possible attacks against the State Electoral Office** that is similar to the removed set of the **possible attacks against the voting application**.

(EO1) Interfere with the printing process;

(EO2) Modify envelopes;
(EO3) Interfere with the ballot printing process;
(EO4) Delay delivery of the voting material to the voter.

We remove the attacks related to storing digital information about the vote in the Vote DB from the set of **possible attacks against the vote collector**. Additionally, we remove the attacks related to timestamping and notifying the voter of receiving their vote. We add the following attack:

(VC‴1) Interfere with the process of marking that the vote has been received.

We can identify the following additional relations between the attacks on the third intermediate system:

- The attacks $\mathsf{V3}, \mathsf{VC}''4, \mathsf{VC}'''1, \mathsf{EO}4$ result in the voter's choice not registered in the system and can be combined in the relation $\mathsf{V3} \prec_S \mathsf{VC}''4 \prec_S \mathsf{VC}'''1 \prec_S \mathsf{EO}4$.

4.5.2. Comparison between the second and the third intermediate systems

Let us denote the third intermediate system as S''', then there are the following relations between the attack set in S'' and S''':

(1) $S''.\mathsf{V}''1 \Leftrightarrow S'''.\mathsf{EO}1$
(2) $S''.\mathsf{VC}16$ or $S''.\mathsf{VC}19 \Leftrightarrow S'''.\mathsf{VC}'''1$
(3) $S''.\mathsf{VC}17$ or $S''.\mathsf{VC}20 \Leftrightarrow S'''.\mathsf{VC}'''1$
(4) $S''.\mathsf{VC}21$ or $S''.\mathsf{VC}22$ or $S''.\mathsf{VC}23 \Leftrightarrow S'''.\mathsf{VC}'''1$

Since the blank ballot is now generated and printed out by the EO, the attack related to the ballot printing process on the voter's side in system S'' is equivalent to the attack related to the ballot printing process on the EO's side in S'''. This is represented by the relation (1).

The main difference between the systems S'' and S''' is in the removed vote registration process that consists of the signing performed by the VC and the RS. Additionally, there is no longer Vote DB in S''' that stores information about the received and recorded votes. It is replaced with maintaining a Voter Listthat contains information about whether the vote has been received.

The relations (2) and (3) express the observation that the attacks related to the vote registration ($S''.\mathsf{VC}16, S''.\mathsf{VC}19, S''.\mathsf{VC}17, S''.\mathsf{VC}20$) in S'' are equivalent to the attack against the process of marking that the vote has been received $S'''.\mathsf{VC}'''1$. Those attacks are considered to be equivalent since the registration of the vote in the system S''' consists of marking that the vote has been received in the Voter List.

The attacks related to the modifications in the Vote DB ($S''.\mathsf{VC}21$, $S''.\mathsf{VC}22$, $S''.\mathsf{VC}23$) in S'' are transformed into the attack related to the modifications on the Voter List $S'''.\mathsf{VC}'''1$. This observation is expressed as the relation (4).

An additional difference comes from removing all the verification steps that are related to the vote registration. Therefore, the voter in S''' has no confirmation of whether their vote was received and recorded by the VC. As a result, all the attacks in S''' related to deleting the vote are more successful, since those cannot be detected by the verification mechanisms.

Attacks related to the signatures of the VC and the RS (and their verification) are no longer applicable in S'''. Since none of the removed attacks were directly related to the vote secrecy or ballot box integrity, the system S'' does not have critical weaknesses that are not present in system S'''.

4.5.3. Comparison between the third intermediate system and postal voting

The third intermediate system is almost the same as the postal voting system $\hat{S}$. Therefore, the attack sets of system S''' and $\hat{S}$ are almost the same. The difference is that the postal voting system has an additional party introduced – a postal service PS that is responsible for delivering all the voting materials between the parties. The attack sets for this party are analogous to the attacks connected with *establishing secure communication channel between the parties* in the i-voting system and intermediate systems which rely on data being transferred over the Internet. Executing attacks in $\hat{S}$ against physical documents, while those are transferred between the parties, became easier for the adversary. The reason is that the modifications are harder to notice, and learning the voter's choice requires just opening the envelope. Deleting the data elements from the system is also easier since it could remain unnoticed or be done just by mistake (e.g. having an incorrect delivery address). Additionally, the attacks targeting the PS are related to the main security requirements of the voting systems as vote secrecy, vote and ballot box integrity.

5. Conclusions

Our reduction from the i-voting system S to the postal voting system $\hat{S}$ identified the following major differences.

- When going from S to S', we see that certain digital signature related attacks are no longer applicable. At the same time, the system S' does not have any mechanism to verify whether any of the signatures given during the protocol are given by the voters, and not altered or forged by the adversary.
- When going from S' to S'' we see that the attacks related to the verification of the vote are no longer applicable. However, by removing the verification mechanism we also remove the ability of the voter to detect vote modification and dropping. Additionally, the system S'' does not rely on encryption. Thereby, the attacks related to learning and modifying the voter's choice have become easier to execute.
- When going from S'' to S''' we remove the attacks related to the registration service. Therefore, the voter in S''' has no confirmation that their vote was successfully received and registered by the system. All the attacks related to dropping the vote in transit or at rest cannot be detected by the voter in the system S''', making the adversary successful in executing them.
- When going from S''' to $\hat{S}$, we add attacks targeting the postal service. All the added attacks are strong since those are directly related to breaking the vote secrecy and vote or ballot box integrity. These attacks do not require additional parties to be corrupt or additional information to be learnt from the other participants of the protocol.

In conclusion, our reduction shows that the Estonian i-voting system indeed has stronger security guarantees than the postal voting system. The reduction shows that in the i-voting system critical attacks that are targeting the main security properties (vote secrecy, vote integrity, ballot box integrity) are distributed between many system components and may require learning different information from different parties in order to execute. In the postal voting system, however, there are less participating parties and all of them are targets to the critical attacks.

Funding

This paper has been supported by the Estonian Research Council under the grant number PRG2177.

References

Abeels, T. (2021). *Postal Voting.* (MSc thesis, Ecole polytechnique de Louvain, Université catholique de Louvain, http://hdl.handle.net/2078.1/thesis:33143)

Abeels, T., & Pereira, O. (2021). *Postal voting* (Unpublished doctoral dissertation). Master's thesis, Ecole polytechnique de Louvain, Université catholique de ....

Bannet, J., Price, D., Rudys, A., Singer, J., & Wallach, D. (2004). Hack-a-vote: Security issues with electronic voting systems. *IEEE Security & Privacy*, *2*(1), 32-37.

Benaloh, J. (2021). STROBE-Voting: Send Two, Receive One Ballot Encoding. In *Electronic voting - 6th international joint conference, e-vote-id 2021, virtual event, october 5-8, 2021, proceedings* (Vol. 12900, pp. 33–46). Springer.

Blanchard, E., Gallais, A., Leblond, E., Sidhoum-Rahal, D., & Walter, J. (2022). An Analysis of the Security and Privacy Issues of the Neovote Online Voting System. In *International joint conference on electronic voting* (pp. 1–18).

Boegehold, A. L. (1963). Toward a study of Athenian voting procedure. *Hesperia: The Journal of the American School of Classical Studies at Athens*, *32*(4), 366–374.

Brightwell, I., Cucurull, J., Galindo, D., & Guasch, S. (2015). An overview of the iVote 2015 voting system. *New South Wales Electoral Commission, Australia, Scytl Secure Electronic Voting, Spain*, 1–25.

Conway, A., & Teague, V. (2022). iVote Issues. *E-Vote-ID 2022*, 42.

Cotti, C., Engelhardt, B., Foster, J., Nesson, E., & Niekamp, P. (2021). The relationship between in-person voting and COVID-19: Evidence from the Wisconsin primary. *Contemporary economic policy*, *39*(4), 760–777.

Debant, A., & Hirschi, L. (2023, August). Reversing, breaking, and fixing the french legislative election E-Voting protocol. In *32nd usenix security symposium (usenix security 23)* (pp. 6737–6752). Anaheim, CA: USENIX Association. Retrieved from https://www.usenix.org/conference/usenixsecurity23/presentation/debant

Ehin, P., Solvak, M., Willemson, J., & Vinkel, P. (2022a). Internet voting in Estonia 2005–2019: Evidence from eleven elections. *Government Information Quarterly*, *39*(4), 101718.

Ehin, P., Solvak, M., Willemson, J., & Vinkel, P. (2022b). Internet voting in Estonia 2005-2019: Evidence from eleven elections. *Gov. Inf. Q.*, *39*(4), 101718.

Gaudry, P., & Golovnev, A. (2020). Breaking the Encryption Scheme of the Moscow Internet Voting System. In J. Bonneau & N. Heninger (Eds.), *Financial cryptography and data security* (pp. 32–49). Cham: Springer International Publishing.

Gebhardt Stenerud, I. S., & Bull, C. (2012). *When reality comes knocking: Norwegian experiences with verifiable electronic voting.* Gesellschaft für Informatik eV.

Gjøsteen, K. (2012). The Norwegian Internet Voting Protocol. In A. Kiayias & H. Lipmaa (Eds.), *E-voting and identity* (pp. 1–18). Berlin, Heidelberg: Springer Berlin Heidelberg.

Goodman, N., Spycher-Krivonosova, I., Essex, A., & Brunet, J. (2023). Verifiability experiences in ontario's 2022 online elections. In M. Volkamer et al. (Eds.), *Electronic voting* (pp. 87–105). Cham: Springer Nature Switzerland.

Haines, T., Pereira, O., & Teague, V. (2022). Running the Race: A Swiss Voting Story. In *International joint conference on electronic voting* (pp. 53–69).

Heiberg, S., Laud, P., & Willemson, J. (2011). The application of i-voting for estonian parliamentary elections of 2011. In *International conference on e-voting and identity* (pp. 208–223).

Heiberg, S., & Willemson, J. (2014). Verifiable internet voting in Estonia. In R. Krimmer &

M. Volkamer (Eds.), *6th international conference on electronic voting: Verifying the vote, EVOTE 2014, lochau / bregenz, austria, october 29-31, 2014* (pp. 1–8). IEEE.

Juvonen, A. (2019). *A framework for comparing the security of voting schemes.* (Master's thesis, University of Helsinki, http://hdl.handle.net/10138/310011)

Krimmer, R., Duenas-Cid, D., & Krivonosova, I. (2021). Debate: safeguarding democracy during pandemics. Social distancing, postal, or internet voting—the good, the bad or the ugly? *Public Money & Management*, *41*(1), 8–10.

Krimmer, R., & Volkamer, M. (2005). Bits or paper? comparing remote electronic voting to postal voting. In *Egov (workshops and posters)* (pp. 225–232).

Krips, K., Snetkov, N., Vakarjuk, J., & Willemson, J. (2023). Trust assumptions in voting systems. *arXiv preprint arXiv:2309.10391*.

Landman, T., & Splendore, L. D. G. (2020). Pandemic democracy: elections and COVID-19. *Journal of risk research*, *23*(7-8), 1060–1066.

Laud, P., & Vakarjuk, J. (2022). A Comparison-Based Methodology for the Security Assurance of Novel Systems. In *Computer security. ESORICS 2022 international workshops - cybericps 2022, SECPRE 2022, SPOSE 2022, CPS4CIP 2022, cdt&secomane 2022, EIS 2022, and secassure 2022, copenhagen, denmark, september 26-30, 2022, revised selected papers* (Vol. 13785, pp. 625–644). Springer.

Li, H., Kankanala, A. R., & Zou, X. (2014). A taxonomy and comparison of remote voting schemes. In *2014 23rd international conference on computer communication and networks* (p. 1-8).

Luechinger, S., Rosinger, M., & Stutzer, A. (2007). The impact of postal voting on participation: Evidence for switzerland. *Swiss Political Science Review*, *13*(2), 167–202.

Moynihan, D. P. (2004a). Building secure elections: E-voting, security, and systems theory. *Public Administration Review*, *64*(5), 515-528. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6210.2004.00400.x

Moynihan, D. P. (2004b). Building secure elections: E-voting, security, and systems theory. *Public Administration Review*, *64*(5), 515-528. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6210.2004.00400.x

Müller, J. (2022). Breaking and fixing vote privacy of the estonian e-voting protocol ivxv. In *Workshop on advances in secure electronic voting 2022.*

Musial-Karg, M. (2016). Alternative voting methods through the example of postal voting and e-voting in switzerland. *Bialstockie Studia Prawnicze*, *20*, 13.

Neumann, S., Noll, M., & Volkamer, M. (2017). Election-dependent security evaluation of internet voting schemes. In *ICT systems security and privacy protection - 32nd IFIP TC 11 international conference, SEC 2017* (Vol. 502, pp. 371–382). Springer.

Pereira, O. (2022). Individual verifiability and revoting in the estonian internet voting system. In *International conference on financial cryptography and data security* (pp. 315–324).

Riigikogu Election Act. (14.12.2020). (https://www.riigiteataja.ee/en/eli/ee/Riigikogu/act/514122020002/consolide)

Shoup, V. (2004). *Sequences of games: a tool for taming complexity in security proofs.* Cryptology ePrint Archive, Paper 2004/332. Retrieved from https://eprint.iacr.org/2004/332 (https://eprint.iacr.org/2004/332)

Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M., & Halderman, J. A. (2014). Security analysis of the estonian internet voting system. In *Proceedings of the 2014 acm sigsac conference on computer and communications security* (p. 703–715). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2660267.2660315

Sutopo, A., Haines, T., & Roenne, P. (2024). On the auditability of the estonian ivxv system. In A. Essex et al. (Eds.), *Financial cryptography and data security. fc 2023 international workshops* (pp. 19–33). Cham: Springer Nature Switzerland.

Teague, V. (2021). Which e-voting problems do we need to solve? In T. Malkin & C. Peikert (Eds.), *Advances in cryptology – crypto 2021* (pp. 3–7). Cham: Springer International Publishing.

Vakarjuk, J., Snetkov, N., & Willemson, J. (2022). Russian federal remote E-voting scheme of 2021–protocol description and analysis. In *Proceedings of the 2022 european interdisciplinary cybersecurity conference* (pp. 29–35).

Vegas, C., & Barrat, J. (2016). Overview of current state of e-voting worldwide. In *Real-world electronic voting* (pp. 67–92). 6000 Broken Sound Parkway, NW, (Suite 300): Auerbach Publications.

Willemson, J. (2018). Bits or paper: Which should get to carry your vote? *Journal of Information Security and Applications*, *38*, 124-131. Retrieved from https://www.sciencedirect.com/science/article/pii/S2214212617304933

World Migration Report 2022. (n.d.). Retrieved from https://publications.iom.int/books/world-migration-report-2022

Appendix A. Process diagrams

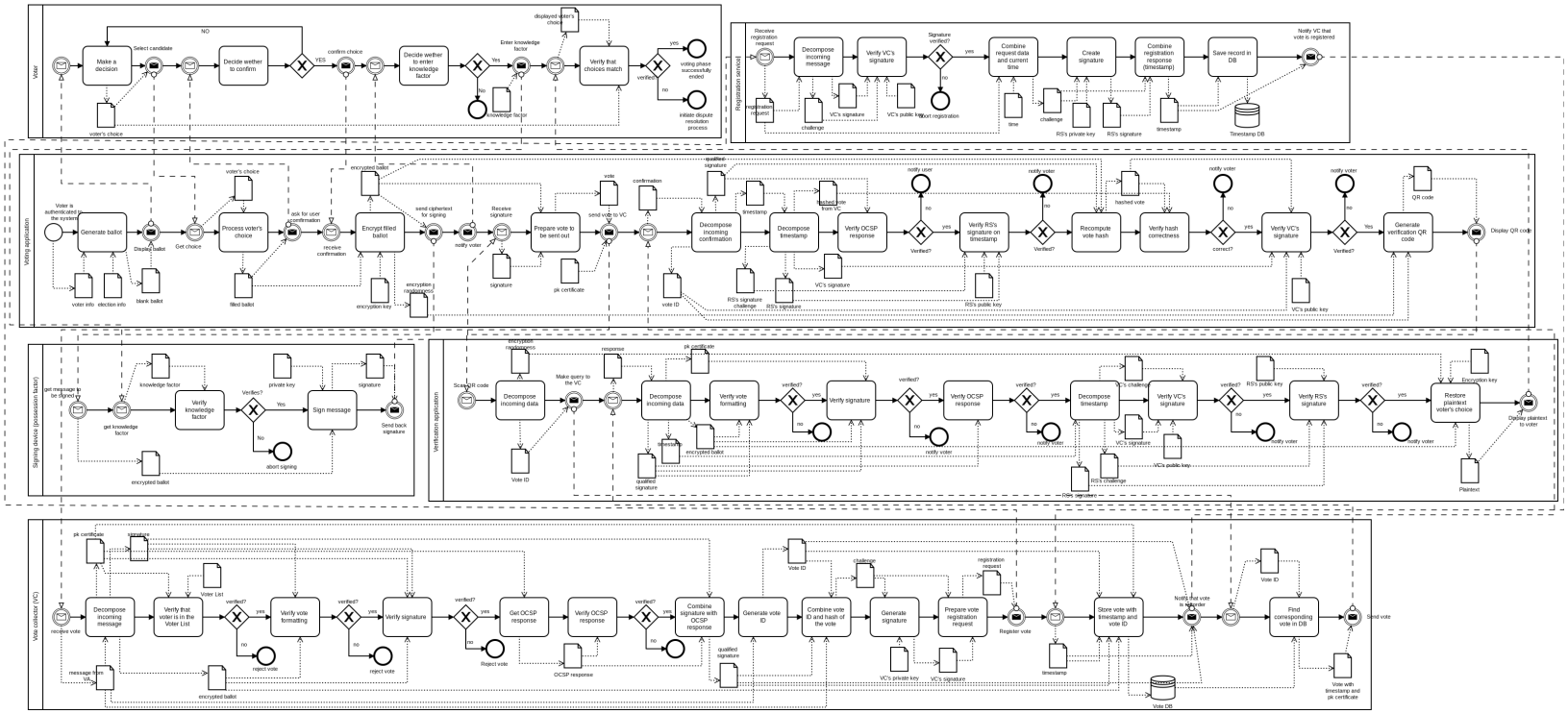

Figure A1. Internet voting system S

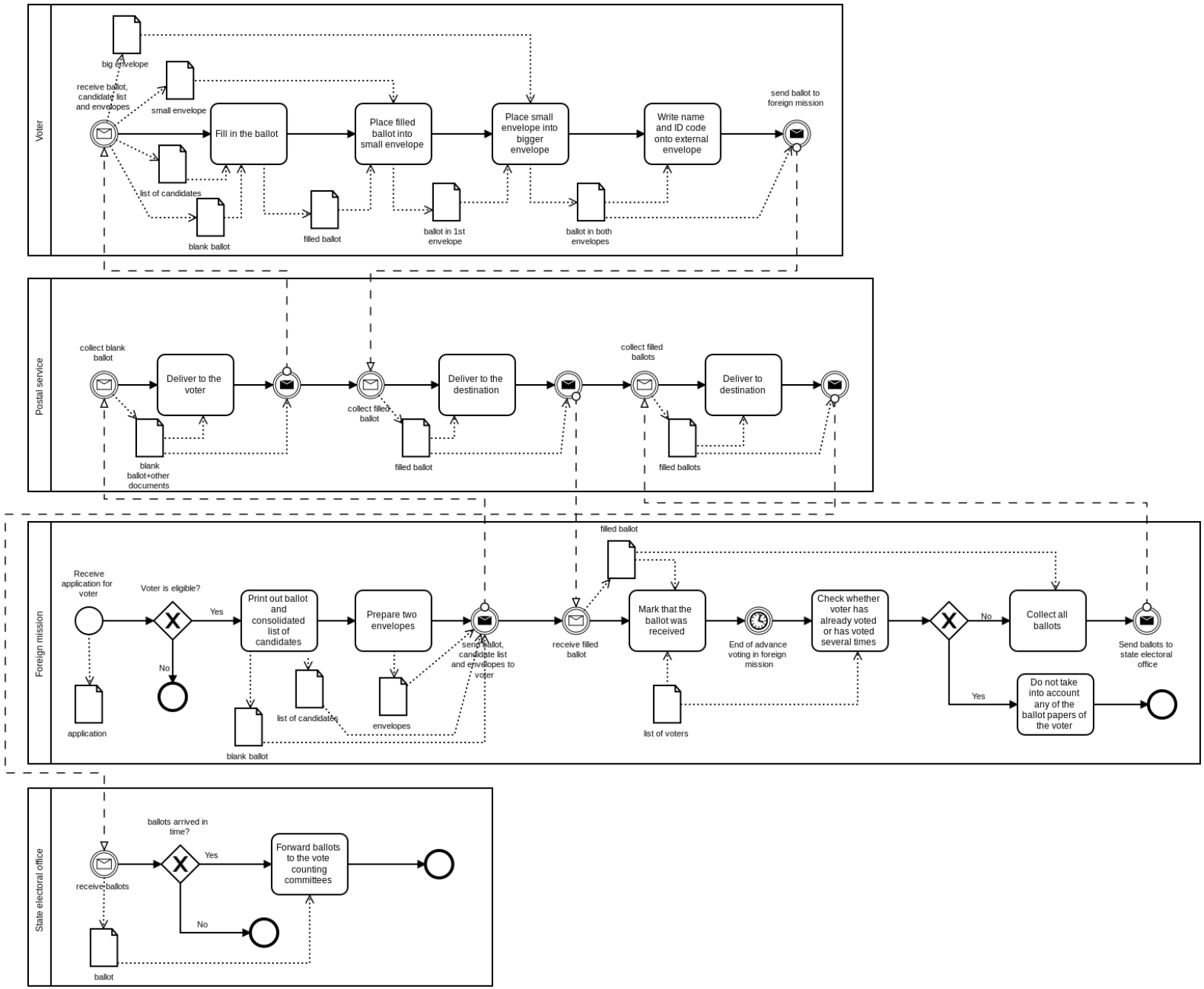

Figure A2. Postal voting system $\hat{S}$

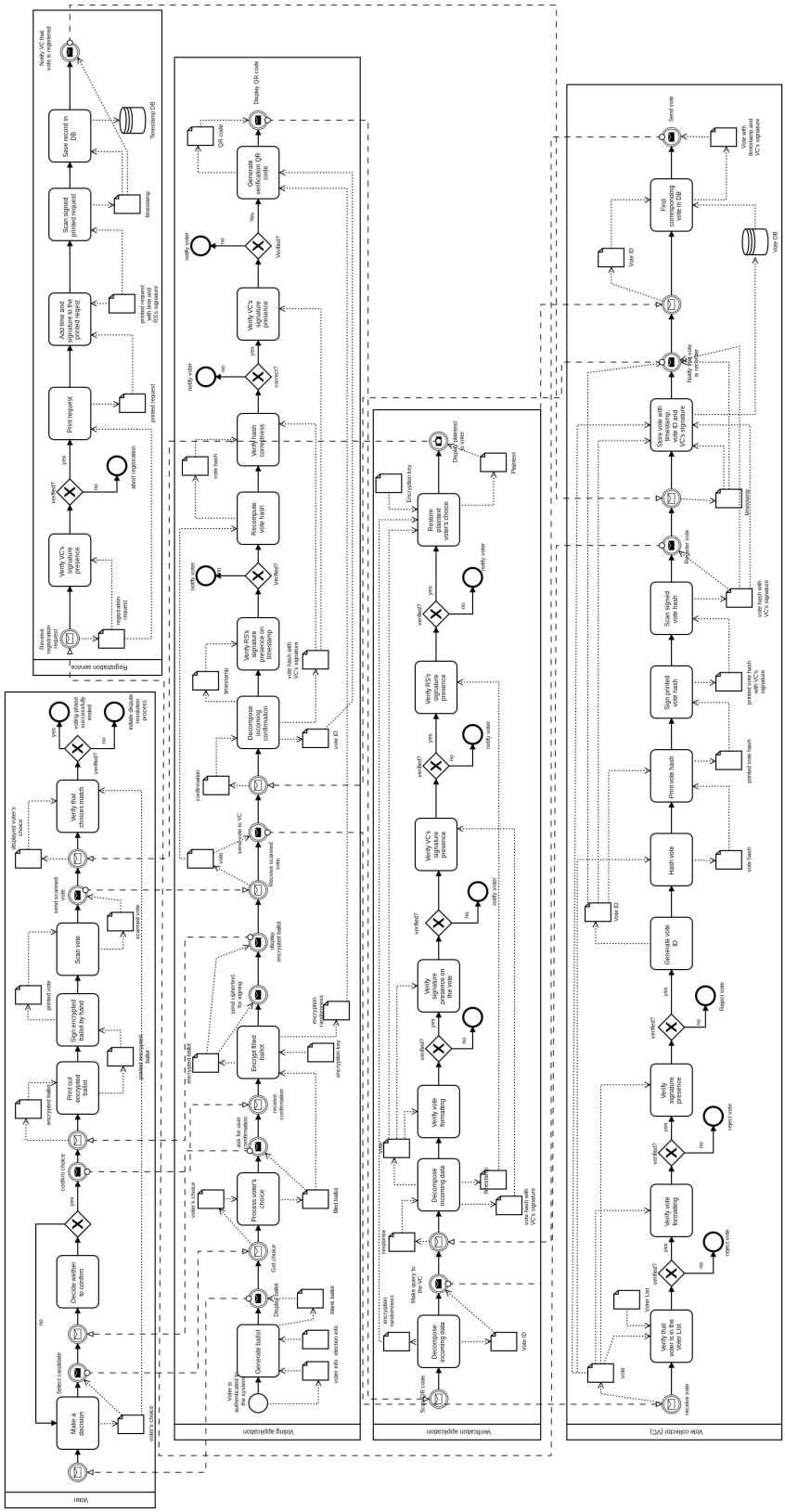

Figure A3. The first intermediate system S'

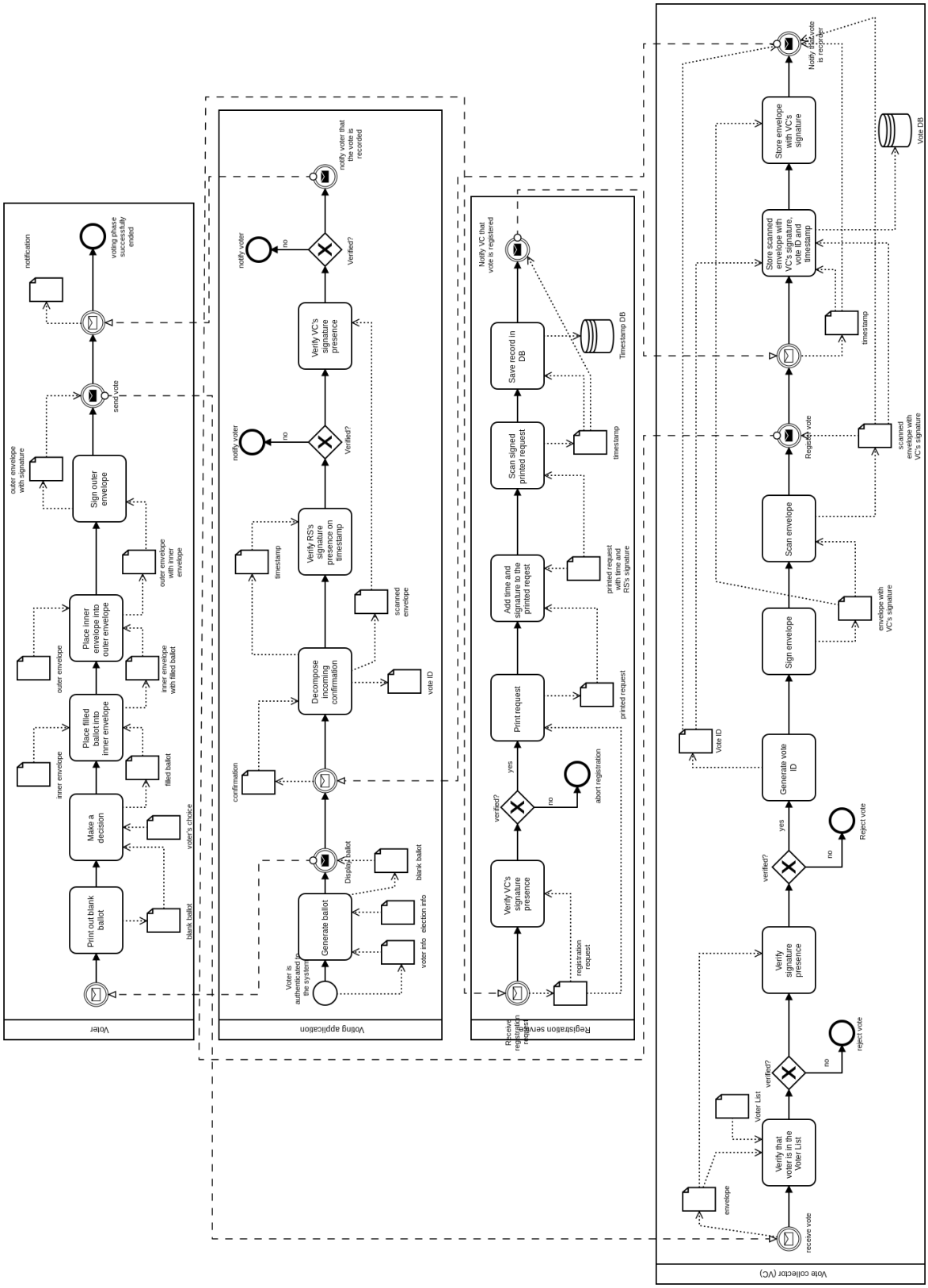

Figure A4. The second intermediate system S''

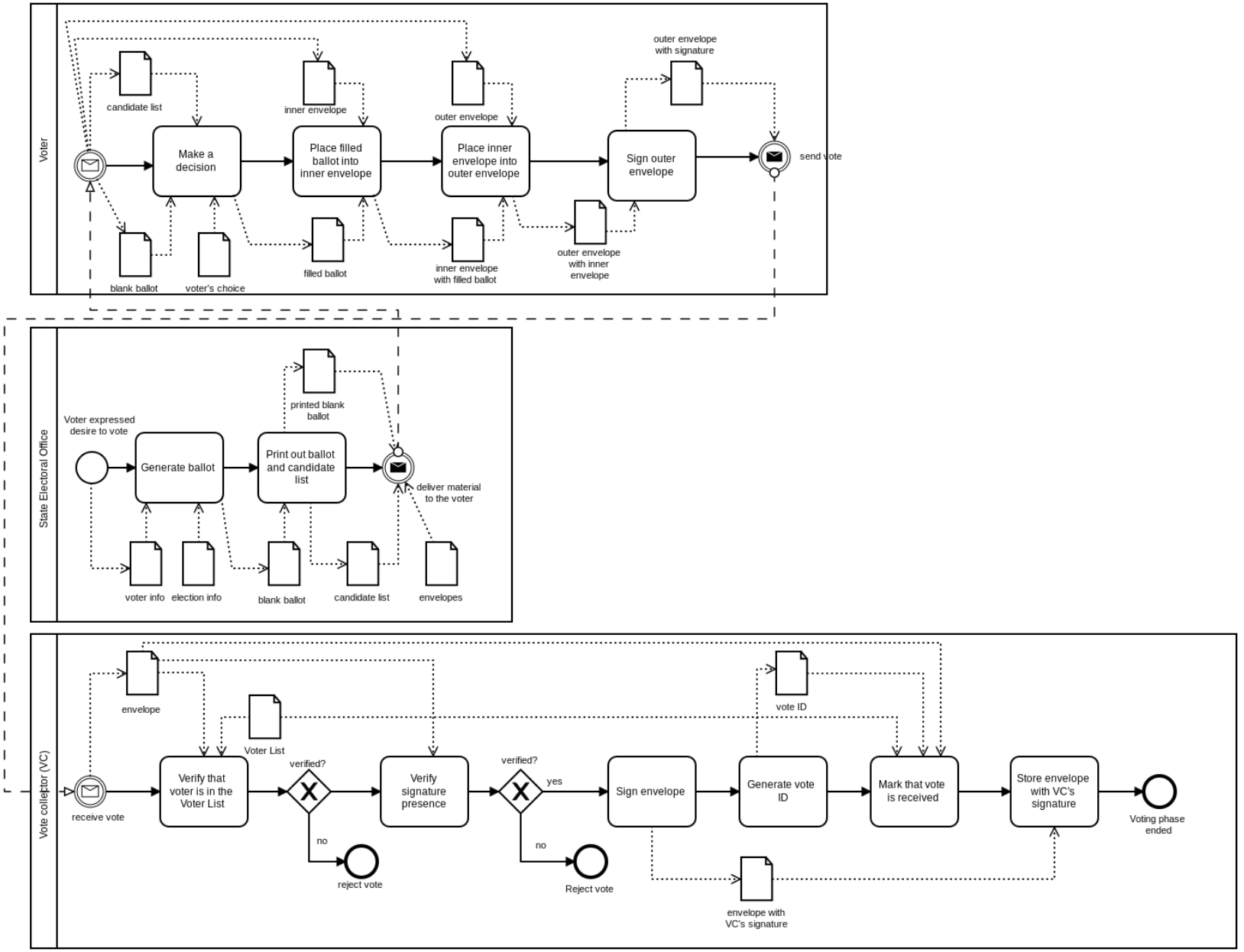

Figure A5. The third intermediate system S'''