Measuring Voter-controlled Privacy

Hugo Jonker*†, Sjouke Mauw† and Jun Pang†

*Eindhoven University of Technology, Department of Mathematics and Computer Science
†University of Luxembourg, Faculty of Sciences, Technology and Communication
Email: {hugo.jonker,sjouke.mauw,jun.pang}@uni.lu

***Abstract*—In voting, the notion of receipt-freeness has been proposed to express that a voter cannot gain any information to prove that she has voted in a certain way. It aims to prevent vote buying, even when a voter chooses to renounce her privacy.**

In this paper, we distinguish various ways that a voter can communicate with the intruder to reduce her privacy and classify these according to their ability to reduce the privacy of a voter. We develop a framework combining knowledge reasoning and trace equivalences to formally model voting protocols and define vote privacy for the voters. Our framework is quantitative, in the sense that it defines a measure for the privacy of a voter. Therefore, the framework can precisely measure the level of privacy for a voter for any of the identified privacy classes. The quantification enables the framework to capture receipts that reduce, but not nullify, the privacy of the voter. This has not been identified and dealt with by other formal approaches.

I. Introduction

With the growth and commercialisation of the Internet, people become more and more concerned about their privacy in the digital world. Privacy has been a fundamental property for systems which provide users e-services, ranging from electronic voting to on-line auctions to health-care.

In voting, vote privacy is the property that an outside observer cannot determine how a voter voted. Although this seems sufficient to ensure privacy, Benaloh and Tuinstra [1] introduce receipt-freeness, which expresses that a voter cannot gain any information to prove to an intruder (someone trying to force the voter to cast a specific vote) that she voted in a certain way. Receipt-freeness aims to prevent vote buying, even when a voter chooses to renounce her privacy. Another stronger notion of privacy is coercion-resistance [2], stating that a voter cannot cooperate with the intruder to prove how she voted. Coercion-resistance also has received considerable attention. These strong notions of privacy actually capture the essential idea that privacy must be enforced by a voting system upon its users, instead offering it. We believe this should be a crucial principle when designing voting systems.

In the literature, many research efforts have been devoted to ensure privacy properties for (electronic) voting. Several schemes claiming to be receipt-free (e.g. [1], [3], [4]) have been proposed and were later shown to have receipts [5], [6], [7]. Resolving this kind of situation underlines the need for formal methods, which are mathematically based techniques to specify and verify systems. Several formalisations of privacy properties in voting have been proposed. Delaune, Kremer and Ryan [8], [9] develop their formalisation based on observational equivalences of processes, whereas we believe privacy requirements should explicitly model the intruder's knowledge. Based on his knowledge, the intruder can distinguish different traces. Moreover, the privacy of a voter in a voting system also depends on how much knowledge (and when) the voter shares with the intruder. In [10], the tool ProVerif is extended to check some (*not all*) equivalences of [8], [9]. Recently, automatic verification techniques within the applied pi calculus framework have been proposed by Backes, Hriţcu and Maffei [11]. But their approach has its focus on remote electronic voting protocols, and mainly deals with coercion-resistance. The constructive approach of Jonker and de Vink [12] is a logical characterization of receipt-freeness, but this work is in a less mature state and thus currently only leads itself to high-level analysis. Baskar, Ramanujam and Suresh [13] define a language to specify voting protocols and use an epistemic logic to reason about receipt-freeness. Although it is relatively easy to express privacy properties based on logic of knowledge, it is rather difficult to develop verification techniques within such a logical framework. Moreover, the aforementioned approaches only offer qualitative methods to analyse whether a voting protocol satisfies some privacy property, instead of offering methods to quantify privacy. However, a qualitative approach leaves several possibilities for an intruder to reduce voter privacy. Examples include forcing a voter *not* to vote for a certain candidate, or determining at which end of the political spectrum was voted. We believe that computing the exact possible choices of a voter is the only way to detect such attacks. In our view, any privacy reductor is a receipt, not just those that nullify privacy.

In this paper, we focus on vote buying, as it is more scalable than coercion in elections [14], and we consider the possibilities for a voter to reduce her privacy, which we call *voter-controlled privacy*. The voter can *subjectively* provide the intruder information, to prove that she has voted in a certain way in order to sell her vote. The intruder tries to find out whether the voter did vote as she said, based on his knowledge gained by observing the election and communicating with the voter. The privacy of any voting system largely depends on the possibilities afforded to the intruder to communicate with a voter. Voting literature [15], [5], [4], uses the following types of information-hiding techniques to ensure privacy: mixnets [16], blind signatures [17], homomorphic encryption [5]. Furthermore, private untappable one-way channels (voter to authority or authority to voter) or two-way channels are often assumed [15], [5], [4]. The ability

to break privacy relies on the information acquired by the intruder. Such channels provide a means to keep information from the intruder and thus help privacy. However, information can be forwarded to the intruder by the voter. And vice versa, a voter could use information supplied by the intruder (which then serves as a witness). These observations lead us to distinguish various independent ways that a voter can communicate with the intruder to rescind her privacy and classify them according to their ability to reduce the privacy of a voter. These classes are ordered according to privacy-reducing ability, with the classic definition of receipt-freeness of Benaloh and Tuinstra [1] being a weakest element in this ordering. The goal of this paper is to provide a method for quantifying privacy of voters within this ordering.

As made clear by Benaloh and Tuinstra [1], voter privacy depends on which knowledge the voter shares with the intruder, and at what time during the protocol this knowledge is shared. (This is captured by the ordering of the conspiracy classes.) Therefore, we develop a framework combining knowledge reasoning along the lines of [12], [13] and trace equivalences of [8], [9] to formally model voting protocols and define vote privacy for the voters. This enables us to describe the knowledge of the intruder, as well as of other entities in a voting protocol, at any point during execution of the system. To distinguish execution traces with respect to the current knowledge of the intruder, we adapt the reinterpretation function proposed by Garcia et al. [18]. The framework is quantitative, in the sense that it can precisely measure privacy loss for any voter conspiring like one of the conspiracy classes, which is achieved by establishing choice groups for voters, taking different communication mechanism between voters and the intruder into account, along the lines of anonymity groups as introduced by Mauw, Verschuren and de Vink [19] and Chothia et al. [20]. Thus, we can capture privacy reducing (but not nullifying) attacks in vote buying, which have not been identified and dealt with in other formal approaches.

Our contribution in this paper is twofold. First, we introduce a new formal framework combining knowledge reasoning and trace equivalences to model and reason about voting protocols. Second, we provide a quantitative definition of vote privacy, which can be used to quantify privacy and enables the detection of some subtle attacks related to vote buying. The different cooperation modes of a voter and the intruder give rise to a conspiracy ordering.

Outline. In Sect. II, we define the setting of our framework. In Sect. III a formal framework to model voting systems and its operational semantics is presented, which is followed by a quantitative definition of privacy in voting in Sect. IV, based on knowledge reasoning and trace equivalences. In Sect. V, the ordering describing how the intruder and a voter can communicate is introduced, and how to measure the loss of privacy within this ordering is formally defined. We perform two case studies in Sect. VI and discuss the power of our framework by identifying some new attacks which can only be detected by a quantitative approach in Sect. VII. Finally, Sect. VIII concludes the paper and lists some research works for the future.

II. Setting

This paper limits itself to studying privacy of a voting system at the protocol level. In the following we define the setting of our approach.

A. Election type

We call the type of elections studied in this paper 1v1v (short for "one voter, one vote"). 1v1v elections have the following properties:

- every voter may cast one vote, and no more;
- each vote has equal weight;
- the election takes place in phases;
- the cast votes (ballots) are made public after the elections.

Phases are used to synchronise all parties, e.g. for ending the voting period. All voting systems have a well-defined endpoint (publication of the result), and hence there are at least two phases (prior to results and results). We assume the ballots are made public to enable public verification of the announced results.

We assume that the way voters vote is independent from the voting process, which implies that the voters' choices can be given a priori. We make this explicit by introducing a function γ that specifies for each voter how she votes.

B. Privacy-enhancing techniques

To enable application to existing voting systems, the formal framework in Sect. III must encompass the privacy-enhancing techniques mentioned in the introduction. Below, these techniques are briefly explained.

- *Homomorphic encryption* is an encryption technique where an operation on two ciphertexts equals another operation on the plaintexts, i.e. for some operations $\otimes$ and $\oplus$ we have $\{\!|\varphi_1|\!\}_k \otimes \{\!|\varphi_2|\!\}_k = \{\!|\varphi_1 \oplus \varphi_2|\!\}_k$.
- *Blind signing* is a signing technique where the signing agent does not see what he is signing – it is blinded. An often-used analogy is that of a sealed envelope with a letter and carbon paper in it. Anyone can sign the letter by signing the envelope, without opening the envelope. The recipient can open the envelope (deblind the message) to acquire the signed letter (message).
- *Mixnets* are a way to establish an anonymous channel between two parties. Each relaying node in the channel permutates the sequence of input messages in output. To prevent directly matching input to output, several cryptographic solutions may be used. For the purpose of this paper, a mixnet is assumed to be a correctly functioning sender-anonymous channel (that is, the channel hides the identity of the sender of a message, but not its contents or its recipient).
- *Untappable channels* are often used assumptions in voting literature. They strengthen the notion of a private channel. Using an untappable channel prevents the intruder from learning the contents of any message communicated, it also makes him unaware of any use of

the channel. Untappable channels may be either one-way or two-way. Sect. V discusses the consequences of untappable channels in depth.

C. The intruder model

We consider a standard Dolev-Yao intruder [21] throughout the paper. This intruder model can be characterised as follows:

- assumption of perfect cryptography - no cryptographic message can be opened without the correct key;
- full network control - all messages sent on the public network are read by the intruder; all received messages on the public network are created or forwarded by the intruder; the intruder can also remove messages from the network.

III. Modelling voting systems

In this section, we develop a formal syntax and provide semantics for expressing the communication behaviour of voting systems. The syntax is rudimentary in that it expresses behaviour of agents as a sequence of events. This restricted format serves the purpose of this paper (enabling reasoning on quantification of vote-privacy). Should more expressivity be desired, the syntax can easily be extended (e.g. with control flow operators).

A. Syntax

A voting system consists of a set of voters $\mathcal{V}$, who make a choice of their preferred candidate from a set of candidates C, and a set of authorities Aut who, e.g., verify eligibility of a voter, and announce the final results. In a voting system, these parties communicate terms. For ease of reference, $Agents = \mathcal{V} \cup Aut$.

As mentioned before, we assume that the way voters vote is independent of the voting process, and hence can be given a priori. This is captured by the relation $\gamma\colon \mathcal{V} \to \mathcal{C}$, which specifies for each voter $v \in \mathcal{V}$ which candidate $c \in \mathcal{C}$ she votes for ($\gamma(v)$). Furthermore, we use projection functions $\pi_i, i > 0$ which return the i^{th} component of a tuple.

The terms communicated in a voting protocol are built up from candidates in $\mathcal{C}$, random numbers in $Nonces$, and cryptographic keys in $Keys$. The set of keys of a particular agent a is given by $Keys_a$. These basic terms can be composed through pairing and encryption. Of the privacy primitives mentioned in Sect. II, both homomorphic encryption and blind signatures operate on terms. Hence, the term modelling needs to accommodate this.

Definition 1 (terms)**.** *Given a set $Vars$ of variables, a set $\mathcal{C}$ of candidates, a set $Nonces$ of nonces, and a set $Keys$ of keys, ranged over by var, c, n and k, respectively. The class $Terms$ of terms, ranged over by φ, is given by the BNF*

$$\varphi ::= var \mid c \mid n \mid k \mid (\varphi_1, \varphi_2) \mid \{\varphi\}_k \mid \{\!|\varphi|\!\}_k \mid [\![\varphi]\!]_k.$$

Terms may be paired ((φ_1, φ_2)), encrypted with a key ($\{\varphi\}_k$), homomorphically encrypted ($\{\!|\varphi|\!\}_k$) and blinded ($[\![\varphi]\!]_k$). A term is called open *if it contains variables and* closed *if it contains no variables. The set of variables of an open term φ are given by* $\mathrm{fv}(\varphi)$.

Terms encrypted with k can be decrypted using the inverse key k^{-1}. For symmetric encryption, $k^{-1} = k$, whereas in asymmetric encryption, $pk(a)$ denotes the public key k and $sk(a)$ the secret key k^{-1} of agent a. Signing is denoted as encryption with the secret key.

A closed term φ *matches* an open term φ' if φ and φ' have the same structure. This is captured by the predicate $\mathrm{match}\colon Terms \times Terms$ as follows: $\mathrm{match}(\varphi_a, \varphi_b) \equiv$

$$\begin{aligned}
&\varphi_a = \varphi_b \ \vee \ \varphi_b \in Vars \ \vee \\
&\langle \exists \varphi'_a, \varphi'_b, k\colon \mathrm{match}(\varphi'_a, \varphi'_b) \ \wedge ((\varphi_a = \{\varphi'_a\}_k \wedge \varphi_b = \{\varphi'_b\}_k) \ \vee \\
&\quad (\varphi_a = \{\!|\varphi'_a|\!\}_k \wedge \varphi_b = \{\!|\varphi'_b|\!\}_k) \vee (\varphi_a = [\![\varphi'_a]\!]_k \wedge \varphi_b = [\![\varphi'_b]\!]_k))\rangle \\
&\vee \langle \exists \varphi'_a, \varphi''_a, \varphi'_b, \varphi''_b\colon \varphi_a = (\varphi'_a, \varphi''_a) \wedge \varphi_b = (\varphi'_b, \varphi''_b) \wedge \\
&\quad \mathrm{match}(\varphi'_a, \varphi'_b) \wedge \mathrm{match}(\varphi''_a, \varphi''_b)\rangle
\end{aligned}$$

Variables represent unspecified terms. An example is the voter's choice: it is represented by variable vc until instantiated. Each agent a captures assigned variables in a partial mapping $\mu_a\colon Vars \to Terms$. The mapping of variable var onto term φ is denoted as $var \mapsto \varphi$. The composition of μ' after μ is denoted by $\mu' \circ \mu$. Variable mappings are extended to open terms. For closed term φ_a and matching open term φ_b (i.e. $\mathrm{match}(\varphi_a, \varphi_b)$), there exists a mapping μ such that $\mu(\varphi_b) = \varphi_a$. This variable mapping is captured by the function vm, where $\mathrm{vm}(\varphi_a, \varphi_b) =$

$$\begin{cases}
\mathit{empty} & \text{if } \varphi_b \in \mathcal{C} \cup Nonces \cup Keys \\
\varphi_b \mapsto \varphi_a & \text{if } \varphi_b \in Vars \\
\mathrm{vm}(\varphi'_a, \varphi'_b) & \text{if } (\varphi_a = \{\varphi'_a\}_k \wedge \varphi_b = \{\varphi'_b\}_k) \vee \\
 & (\varphi_a = \{\!|\varphi'_a|\!\}_k \wedge \varphi_b = \{\!|\varphi'_b|\!\}_k) \vee \\
 & (\varphi_a = [\![\varphi'_a]\!]_k \wedge \varphi_b = [\![\varphi'_b]\!]_k), \\
 & \text{for some } k \in Keys \\
\mathrm{vm}(\varphi'_a, \varphi'_b) \circ \mathrm{vm}(\varphi''_a, \varphi''_b) & \text{if } \varphi_i = (\varphi'_i, \varphi''_i), \text{for } i \in \{a, b\}
\end{cases}$$

A term φ may be derived from a set of terms T (notation $T \vdash \varphi$) if it is an element of T or if it can be derived by repeatedly applying the following rules:

$$\begin{array}{ll}
T \vdash \varphi_1,\ T \vdash \varphi_2 & \Longrightarrow\ T \vdash (\varphi_1, \varphi_2) \\
T \vdash (\varphi_1, \varphi_2) & \Longrightarrow\ T \vdash \varphi_1 \\
T \vdash (\varphi_1, \varphi_2) & \Longrightarrow\ T \vdash \varphi_2 \\
T \vdash \varphi_1,\ T \vdash k & \Longrightarrow\ T \vdash \{\varphi_1\}_k \\
T \vdash \{\varphi_1\}_k,\ T \vdash k^{-1} & \Longrightarrow\ T \vdash \varphi_1 \\
T \vdash \varphi_1, T \vdash k & \Longrightarrow\ T \vdash \{\!|\varphi_1|\!\}_k \\
T \vdash \{\!|\varphi_1|\!\}_k, T \vdash k^{-1} & \Longrightarrow\ T \vdash \varphi_1 \\
T \vdash \varphi_1, T \vdash k & \Longrightarrow\ T \vdash [\![\varphi_1]\!]_k \\
T \vdash \{[\![\varphi_1]\!]_k\}_{sk(a)}, T \vdash k & \Longrightarrow\ T \vdash \{\varphi_1\}_{sk(a)} \\
T \vdash \{\!|\varphi_1|\!\}_k, T \vdash \{\!|\varphi_2|\!\}_k & \Longrightarrow\ T \vdash \{\!|\varphi_1 \oplus \varphi_2|\!\}_k
\end{array}$$

An agent's knowledge is a set of terms closed under derivability. Closure of a set K under derivability is defined as $\overline{K} = \{\varphi \mid K \vdash \varphi\}$. As proven in [13], $\overline{T} \vdash t$ is decidable.

Terms are communicated between the agents. These communications may occur over public, anonymous, or untappable channels. The distinction between these channels will become clear in the semantics for these channels.

Definition 2 (events)**.** *The class of communication events*

(public send)
$$\frac{k_a \vdash \varphi' \wedge sp = a\colon (k_a, \mu, s(a, y, \varphi) \cdot \sigma) \in S \wedge \mu(\varphi) = \varphi'}{(K_I, S) \xrightarrow{s(a,y,\varphi')} (K_I \cup \{\varphi'\}, S \cup \{a\colon (k_a, \mu, \sigma)\} \setminus \{sp\}\ )}$$

(public receive)
$$\frac{K_I \vdash \varphi \wedge sp = a\colon (k_a, \mu, r(x, a, \varphi') \cdot \sigma) \in S \wedge \mu' = \mathrm{vm}(\varphi, \mu(\varphi')) \circ \mu \wedge \mathrm{match}(\varphi, \mu(\varphi'))}{(K_I, S) \xrightarrow{r(x,a,\varphi)} (K_I, S \cup \{a\colon (k_a \cup \{\varphi\}, \mu', \sigma)\} \setminus \{sp\}\ )}$$

(anonymous send)
$$\frac{k_a \vdash \varphi' \wedge sp = a\colon (k_a, \mu, as(a, y, \varphi) \cdot \sigma) \in S \wedge \mu(\varphi) = \varphi'}{(K_I, S) \xrightarrow{as(y,\varphi')} (K_I \cup \{\varphi'\}, S \cup \{a\colon (k_a, \mu, \sigma)\} \setminus \{sp\}\ )}$$

(anonymous receive)
$$\frac{K_I \vdash \varphi \wedge sp = a\colon (k_a, \mu, ar(a, \varphi') \cdot \sigma) \in S \wedge \mu' = \mathrm{vm}(\varphi, \mu(\varphi')) \circ \mu \wedge \mathrm{match}(\varphi, \mu(\varphi'))}{(K_I, S) \xrightarrow{ar(a,\varphi)} (K_I, S \cup \{a\colon (k_a \cup \{\varphi\}, \mu', \sigma)\} \setminus \{sp\}\ )}$$

(untappable communication)
$$\frac{\begin{array}{l} sp_a = a\colon (k_a, \mu_a, us(a, b, \varphi_a) \cdot \sigma_a) \in S\ \wedge k_a \vdash \varphi' \wedge \mu_a(\varphi_a) = \varphi'\ \wedge \\ sp_b = b\colon (k_b, \mu_b, ur(a, b, \varphi_b) \cdot \sigma_b) \in S \wedge \mu'_b = \mathrm{vm}(\varphi_a, \mu(\varphi_b)) \circ \mu_b \wedge \mathrm{match}(\varphi_a, \mu_b(\varphi_b)) \end{array}}{(K_I, S) \xrightarrow{\tau} (K_I, S \cup \{a\colon (k_a, \mu_a, \sigma), b\colon (k_b \cup \{\varphi'\}, \mu'_b, \sigma_b)\} \setminus \{sp_a, sp_b\})}$$

(phase synchronisation)
$$\frac{Phase = \{a\colon (k_a, \mu_a, phase(i) \cdot \sigma_a) \in S\} \wedge \forall a \in Aut\colon a\colon (k_a, \mu_a, phase(i) \cdot \sigma_a) \in S}{(K_I, S) \xrightarrow{phase(i)} (K_I, S \cup \{a\colon (k_a, \mu_a, \sigma_a) \mid a\colon (k_a, \mu_a, phase(i) \cdot \sigma_a) \in S\} \setminus Phase)}$$

Fig. 1. Operational semantics

$Events$ is given by:

$$\begin{aligned} Events = \ & \{s(a, a', \varphi), r(a, a', \varphi), as(a, a', \varphi), ar(a', \varphi), \\ & us(a, a', \varphi), ur(a, a', \varphi), phase(i) \\ & \mid a, a' \in Agents, \varphi \in Terms, i \in \mathbb{N}\}, \end{aligned}$$

where s, r, as, ar, us, ur denote sending and receiving over public, anonymous and untappable channels, respectively. Finally, $phase(i)$ is the event denoting that an agent is ready to start phase i.

An agent's behaviour is given by its specification. A specification $sp \in Spec$ is a tuple of knowledge, variable mapping, and events: $Spec = \mathcal{P}(Terms) \times (Vars \to Terms) \times Events^\star$. Not all expressible specifications make sense (e.g. two consecutive receive events using the same variables). There are some static requirements on well-formed behaviours which we omit here for space considerations. Variable mappings are extended to events (replacing all mapped variables in the term of the event) and to specifications (applying to each event in the list of events). This is denoted as $\mu(ev)$ for an event ev and $\mu(\sigma)$ for an event list σ.

A voting system specifies, for each agent, its behaviour. Hence, a voting system is a mapping from agent identities to specifications, as follows.

Definition 3 (voting system)**.** *The class of voting systems, $Prot$, is defined as $Prot = Agents \to Spec$. Instantiation of a voting system $\mathcal{VS} \in Prot$ with choice function γ is denoted as $\mathcal{VS}(\gamma)$. $\mathcal{VS}(\gamma)(a) =$*

$$\begin{cases} \mathcal{VS}(a) & \text{if } a \notin \mathcal{V} \\ (\pi_1(\mathcal{VS}(a)), \mu_a(\pi_2(\mathcal{VS}(a))), \pi_3(\mathcal{VS}(a))) & \text{if } a \in \mathcal{V} \end{cases}$$

where $\mu_a = vc \mapsto \gamma(a)$.

B. Semantics

State of a voting system is a tuple of intruder knowledge and agent specifications: $State = \mathcal{P}(Terms) \times (Agents \to Spec)$. Each agent's specification (knowledge, variable mapping and list of events) is given by $Agents \to Spec$. To denote the specification of agent a in a state (K_I, S), we write $a\colon (k_a, \mu_a, \sigma_a) \in S$. When there is no risk of confusion, we omit the agent's identity from the specification, as in $a\colon (k, \mu, \sigma) \in S$. The initial state of voting system $\mathcal{VS}$ with respect to choice function γ is $(K_I^0, \mathcal{VS}(\gamma))$, for the intruder with initial knowledge K_I^0.

The operational semantics of a voting system, given in Fig. 1 and detailed below, models a Dolev-Yao intruder. Events may involve unspecified agents $x, y \in Agents$, which we omit from the premise of the rules. Events involving parties $\notin Agents$ (e.g., the intruder) explicitly name these parties. The semantics of a voting system results in a labelled transition system where the labels are elements of $Labels$:

$$\begin{aligned} Labels = \{ & s(a, a', \varphi), r(a, a', \varphi), as(a', \varphi), ar(a', \varphi), \tau, \\ & phase(i) \mid a, a' \in Agents, \varphi \in Terms, i \in \mathbb{N}\}. \end{aligned}$$

Note that the sender of a message over the anonymous channel is hidden. Furthermore, phases denote synchronisation points. A $phase(i)$ event may only occur if all authorities have agreed that the election will evolve into a new phase. As a consequence, those voters who are ready to do so will move the new phase as well.[1] We use the label τ to denote the occurrence of a communication over untappable channels.

Send events work as follows. The sending agent can send term φ if he can derive a matching instantiated φ' from his

[1]We conjecture that our semantics of phase synchronisation is similar to the strong phase semantics as proposed in [10].

current knowledge and variable mapping. The result of both a public send event and an anonymous send event is that the intruder's knowledge is extended (with φ'), and that the send event is removed from the agent's list of events. The distinction between public and anonymous communication is that the latter omits the sender in the label of the transition (*public* / *anonymous send* in Fig. 1).

In receive events, an agent can receive term φ' if the intruder can derive φ, the agent's next event is a receive event of term φ' and φ matches φ'. As a result of receiving φ, the agent's knowledge is extended with φ and the free variables of φ' are assigned values in his variable mapping, since an agent may (partly) specify the structure of a term it receives using the match predicate. An anonymous receive event is similar to a public receive event, except that it omits the sender in the label of the transition (*public* / *anonymous receive* in Fig. 1).

We model untappable communications to be synchronised, as the intruder has no influence or power whatsoever over this communication channel. The intruder does not learn anything about the communication, hence his knowledge is not updated. Other than that, the rule combines the ideas of the public send and public receive events (*untappable communication* in Fig. 1).

Phase synchronisation only occurs if all authorities from Aut agree to progress to phase i. In this case, all other agents ready to execute the phase event will move to the new phase as well (*phase synchronisation* in Fig. 1).

The set of traces of a voting system is defined as follows using these semantic rules:

Definition 4 (traces)**.** *The class of traces $Traces$ consists of lists of labels. The traces of a voting system $\mathcal{VS}$ and a choice function γ are given by*

$$\begin{aligned} Tr(\mathcal{VS}(\gamma)) = \{\alpha \in \mathit{Labels}^\star \mid \alpha = \alpha_0, \ldots, \alpha_{n-1} \wedge \\ \exists s_0, \ldots, s_n \in \mathit{State}\colon s_0 = (K_I^0, \mathcal{VS}(\gamma)) \wedge \\ \forall 0 \leq i < n\colon s_i \xrightarrow{\alpha_i} s_{i+1}\} \end{aligned}$$

The set of traces of a voting system $\mathcal{VS}$ is now given by

$$Tr(\mathcal{VS}) = \bigcup_{\gamma \in \mathcal{V} \to \mathcal{C}} Tr(\mathcal{VS}(\gamma))$$

We denote the intruder knowledge in the last state of a trace t as K_I^t. The empty trace is denoted by ϵ.

IV. Privacy in voting systems

The model developed in the previous section enables us to express if an intruder can distinguish two executions of the system, as previously expressed by Mauw, Verschuren and de Vink [19] and later by Garcia et al. [18] for passive intruders. Traces t, t' are to be considered equivalent if the intruder cannot distinguish them. To formalise this equivalence, the distinguishing ability of the intruder is formalised as the intruder's ability to distinguish two messages, which is then lifted to traces. The intruder can uniquely identify any plaintext message he sees. Furthermore, the intruder can distinguish any encrypted message for which he possesses the decryption key, or which he can construct himself.

Definition 5 (reinterpretation [18])**.** *Let ρ be a permutation on the set of terms $Terms$ and let K_I be a knowledge set. The map ρ is a* semi-reinterpretation under K_I *if it satisfies the following.*

$$\begin{aligned} \rho(p) &= p, \text{ for } p \in \mathcal{C} \cup Nonces \cup Keys \\ \rho((\varphi_1, \varphi_2)) &= (\rho(\varphi_1), \rho(\varphi_2)) \\ \rho(\{\varphi\}_k) &= \{\rho(\varphi)\}_k, \text{ if } K_I \vdash \varphi, k \vee K_I \vdash \{\varphi\}_k, k^{-1} \\ \rho(\{\!|\varphi|\!\}_k) &= \{\!|\rho(\varphi)|\!\}_k, \text{ if } K_I \vdash \varphi, k \vee K_I \vdash \{\varphi\}_k, k^{-1} \\ \rho([\![\varphi]\!]_n) &= [\![\rho(\varphi)]\!]_n, \text{ if } K_I \vdash n \end{aligned}$$

Map ρ is a reinterpretation under K_I *iff it is a semi-reinterpretation and its inverse ρ^{-1} is a semi-reinterpretation under $\rho(K_I)$.*

Reinterpretation is extended straightforwardly to events and to traces by applying ρ to the message fields of events (in traces).

Some events in a trace are hidden from the intruder, hence the intruder has a restricted view of a trace. In particular, the intruder cannot see any τ transitions (communications over untappable channels). The visible part of a trace is captured by the function $obstr\colon Traces \to Traces$ as follows:
$obstr(\epsilon) = \epsilon$, and $obstr(\ell \cdot t) = \begin{cases} \ell \cdot obstr(t) & \text{if } \ell \neq \tau \\ obstr(t) & \text{if } \ell = \tau \end{cases}$

Definition 6 (trace indistinguishability)**.** *Traces t, t' are indistinguishable for the intruder, notation $t \sim t'$ iff there exists a reinterpretation ρ such that*

$$obstr(t') = \rho(obstr(t)) \ \wedge \ \overline{K_I^t} = \rho(\overline{K_I^{t'}}).$$

The above definition of the intruder's ability to distinguish traces extends to his ability to distinguish sets of traces as follows.

Definition 7 (choice indistinguishability)**.** *Given voting system $\mathcal{VS}$, choice functions γ, γ' are indistinguishable to the intruder, notation $\gamma \simeq_{\mathcal{VS}} \gamma'$ iff*

$$\begin{aligned} &\forall t \in Tr(\mathcal{VS}(\gamma))\colon \exists t' \in Tr(\mathcal{VS}(\gamma'))\colon t \sim t' \quad \wedge \\ &\forall t \in Tr(\mathcal{VS}(\gamma'))\colon \exists t' \in Tr(\mathcal{VS}(\gamma))\colon t \sim t' \end{aligned}$$

Note that, as the ballots are made public, the intruder knows the result. As such, he can discard choice functions not matching the result.

The set of choice functions indistinguishable for the intruder in a given system is now succinctly defined as follows.

Definition 8 (choice group)**.** *The choice group for a voting system $\mathcal{VS}$ and a choice function γ is given by*

$$cg(\mathcal{VS}, \gamma) = \{\gamma' \mid \gamma \simeq_{\mathcal{VS}} \gamma'\}.$$

The choice group for a particular voter v, i.e. the set of candidates indistinguishable from v's chosen candidate, is given by

$$cg_v(\mathcal{VS}, \gamma) = \{\gamma'(v) \mid \gamma' \in cg(\mathcal{VS}, \gamma)\ \}.$$

V. Conspiring voters

A. The conspiracy classes

The above framework captures the behaviour of a passive voter, who does not actively cooperate with the intruder to prove how she has voted. However, as remarked in the introduction, we focus on voters trying to renounce their vote-privacy. A conspiring voter can try to share her knowledge with the intruder. The classic receipt-freeness case assumes the voter shares her final knowledge. However, the timing of knowledge sharing is important (as explained in [12]) – the voter needs to share her personal knowledge before it becomes public, in order to prove that she really has the receipt. Additionally, the use of untappable channels means that during the course of an election, a voter may learn or commit to knowledge that the intruder is unaware of. Recall that untappable channels hide communications between a voter and the authorities completely from the intruder. By sharing knowledge received over such a channel with the intruder, and by using intruder-supplied information to send over such a channel (and thus commit to), a voter may seek to circumvent the privacy provisions of untappable channels. The timing of sharing information between the conspiring voter and the intruder hence is important.

Cases where the voter shares her full knowledge (post-election or pre-election) are independent from cases where the voter conspires to circumvent untappable channels (sharing information, using intruder-supplied information, or both). In absence of untappable channels, all communications are visible to the intruder. In this case, the sooner a voter shares her knowledge with the intruder, the more traces the intruder can distinguish. Classical receipt-freeness, *classic-rf*, tries to break vote-privacy by sharing knowledge after elections. However, sharing knowledge beforehand, *start-rf*, gives the intruder more knowledge during the elections. This situation is depicted below in Fig. 2(i).

In presence of untappable channels, the intruder is not aware of every addition to the voter's knowledge. The voter can mitigate this by conspiring mid-election. Her willingness to do so is captured in Fig. 2(ii). The conspiring voter may choose to share information the intruder cannot learn otherwise (*rf-share*) or use intruder-supplied terms in communications hidden from the intruder (*rf-witness*) to later prove how she voted. The combination of these two notions is at the top of the ordering (*rf-full*).

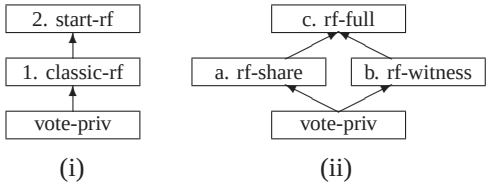

Fig. 2. (i) pre-, post- and (ii) mid-election knowledge sharing

A voter may use privacy-reducing techniques from both hierarchies to reduce her privacy. We denote this, e.g., as a *type 1a* voter, or a *type 2c* voter.

B. Modelling conspiratory behaviour

A conspiring voter behaves differently from a regular voter, as she will communicate with the intruder in certain circumstances. To incorporate the different conspiracy classes into the framework of Sect. III and Sect. IV, we extend the set of events $Events$ with events: $\{is(\varphi), ir(\varphi)\}$, where $is(\varphi)$ denotes the agent sending term φ to the intruder, and $ir(\varphi)$ denotes the converse receive event. These events have similar semantics to the public sending/receiving events.

Modelling conspiracy changes the voter specification as follows:

- **1. classic-rf:** at the end of the protocol, the voter sends her knowledge set to the intruder (is is extended to send knowledge set).
- **2. start-rf:** at the beginning of the protocol, the voter sends her knowledge set to the intruder.
- **a. rf-share:** each $ur(a, v, \varphi)$ is followed by an $is(\varphi)$.
- **b. rf-witness:** The intruder supplies the voter-controllable parts of the term used in each $us(v, a, \varphi)$. To do so, the intruder must know what terms are voter-controllable. To this end, we introduce a function $\mathrm{vars}(v, \varphi)$ that returns the variables of φ that agent v can control. The voter sends this information to the intruder, who replies with a similar term, changing the values to his liking. The voter then uses the newly supplied values to communicate.
- **c. rf-full:** this combines rf-share and rf-witness.

The variables controllable by voter v in term φ are given by the function vars as follows: $\mathrm{vars}(v, \varphi) =$

$$\begin{cases} \{\varphi\} & \text{if } \varphi \in Vars \\ \mathrm{vars}(v, \varphi_a) \cup vars(v, \varphi_b) & \text{if } \varphi = (\varphi_a, \varphi_b) \\ \mathrm{vars}(v, \varphi') & \\ \qquad \text{if } (\varphi = \{\varphi'\}_k \vee \varphi = \{\!|\varphi'|\!\}_k \vee \varphi = [\![\varphi']\!]_k), & \\ \qquad\qquad \text{for } k \in Keys_v & \\ \emptyset & \text{otherwise} \end{cases}$$

Changes to the specification are modelled by introducing a protocol transformation function that transforms the specification of one agent as outlined above.

Definition 9 (protocol transformation)**.** *Recall the type of the class of voting protocols, $Prot\colon Agents \to (\mathcal{P}(Terms) \times (Vars \to Terms) \times Events^\star)$. We introduce $\Delta_i\colon Agents \times Prot \to Prot$, a protocol transformation function for $i \in \{1, 2, a, b, c\}$ a conspiracy class, where $\Delta_i(v, \mathcal{VS})(a) =$*

$$\begin{cases} \mathcal{VS}(a) & \text{if } a \neq v \\ (\pi_1(\mathcal{VS}(v)), \pi_2(\mathcal{VS}(v)), \delta_i(v, \pi_3(\mathcal{VS}(v)))) & \text{if } a = v \end{cases}$$

The transformation function $\delta_i\colon \mathcal{V} \times Events^\star \to Events^\star$ depends on the specific conspiring behaviour as follows.

- $\delta_1(v, sp) = sp \cdot is(k_v)$.
- $\delta_2(v, sp) = is(k_v) \cdot sp$.
- $\delta_a(v, ev \cdot sp) =$
  $\begin{cases} ur(ag, v, \varphi) \cdot is(\varphi) \cdot \delta_a(v, sp) & \text{if } ev = ur(ag, v, \varphi) \\ ev \cdot \delta_a(v, sp) & \text{otherwise} \end{cases}$

- $\delta_b(v, ev \cdot sp) =$
$$\begin{cases} is(\mathrm{vars}(v,\varphi)) \cdot ir(\varphi') \cdot us(v, ag, \varphi'') \cdot \delta_b(v, sp) \\ \qquad\qquad\qquad\qquad \textit{if } ev = us(v, ag, \varphi) \\ ev \cdot \delta_b(v, sp) \qquad\qquad \textit{otherwise} \end{cases}$$
- $\delta_c(v, sp) = \delta_b(v, \delta_a(v, sp))$.

The transformation δ_b works as follows: the voter controlled information in the term φ is sent as a pairing of all elements in the set $\mathrm{vars}(v, \varphi)$ to the intruder. The intruder's reply is received in term φ', a similar pairing, but with newly introduced variables. The term φ'' that the voter sends on the untappable channel has the same structure as the original term ($\mathrm{match}(\varphi'', \varphi)$), but uses the corresponding intruder-supplied fresh variables ($\mathrm{fv}(\varphi'') = \mathrm{fv}(\varphi')$).

The above transformations are extended to combinations of conspiring behaviour ($i \in \{1a, 2a, 1b, 2b, 1c, 2c\}$), e.g. $\Delta_{1a}(v, \mathcal{VS}) = \Delta_1(v, \Delta_a(v, \mathcal{VS}))$. Using the above protocol transformations, we can define choice group for conspiring voters in a voting system within our framework (see Sect. IV).

Definition 10 (vote privacy)**.** *In voting system $\mathcal{VS}$, with a given choice function γ, the choice group of voter v with respect to different conspiracy classes $i \in \{1, 2, a, b, c, 1a, 2a, 1b, 2b, 1c, 2c\}$, is given by*

$$cg_v^i(\mathcal{VS}, \gamma) = \{\gamma'(v) \mid \gamma' \in cg(\Delta_i(v, \mathcal{VS}), \gamma)\}.$$

Given this definition of privacy, receipt-freeness is a measure of the voter's privacy as follows.

Definition 11 (receipt-freeness)**.** *Voting system $\mathcal{VS}$ is receipt-free for conspiring behaviour i iff*

$$\forall v \in \mathcal{V}, \gamma \in \mathcal{V} \to \mathcal{C}\colon cg_v^i(\mathcal{VS}, \gamma) = cg_v(\mathcal{VS}, \gamma).$$

Note that the classic notion of receipt-freeness as introduced by Benaloh and Tuinstra coincides with $\forall v, \gamma\colon \left|cg_v^1(\mathcal{VS}, \gamma)\right| > 1$. The methods of Delaune, Kremer and Ryan can be expressed as the size of the choice group being larger than one. We consider a receipt a modifier of privacy instead of a privacy-nullifier. The amount of receipt-freeness is thus given by the difference in privacy between a regular voter and a conspiring voter. The above privacy definitions capture this by determining the exact choice group and thus the exact voter privacy for any level of conspiracy.

VI. Applications

The above presented framework is designed to provide a very precise and formal analysis. However, a full analysis of an existing voting system goes beyond the scope of this paper. Therefore, this section only illustrates application of the framework at a high level. For this purpose, models of the FOO [17] system and the ThreeBallot system (3BS, [22]) are examined.

FOO. Our specification of the FOO protocol is similar to that of [13], and thus omitted here. The FOO protocol has been studied in literature and is known to have receipts [12], [13]. FOO does not use untappable channels, thus there are no conspiring voters of type a, b, c. The only possibility for a voter to conspire is by sharing knowledge as in Fig. 2(i). A conspiring voter of type 1 already nullifies her privacy – $\forall v, \gamma\colon \left|cg_v^1(FOO, \gamma)\right| = 1$. This is because the keys used to encrypt the vote are shared with the intruder. This makes any other reinterpretation of the voter's message carrying her vote impossible. In FOO, a table with all cast ballots is published. Later, the counting authority updates the table to include all voter keys. One possible amelioration of FOO (call this FOO') is to publish the updated table. This prevents conspirators of type 1, which implies that the choice groups with and without type 1 conspiracy are equal: $\forall \gamma, v\colon cg_v(FOO, \gamma) = cg_v^1(FOO', \gamma)$. However, type 2 conspirators remain unaffected: $\forall \gamma, v\colon \left|cg_v^2(FOO, \gamma)\right| = \left|cg_v^2(FOO', \gamma)\right| = 1$.

ballot 1a	ballot 1b	ballot 1c
can 1 ☐	can 1 ☐	can 1 ☐
⋮ ⋮	⋮ ⋮	⋮ ⋮
can N ☐	can N ☐	can N ☐
identifier 1a	identifier 1b	identifier 1c

Fig. 3. A Threeballot in 3BS.

ThreeBallot. In 3BS, a vote is split over three single ballots (see Fig. 3), which together form one Threeballot. Each ballot carries a unique identifier. Upon voting, the three ballots are cast and the voter takes home a receipt (certified copy) of one ballot. The copy allows the voter to verify that her Threeballot is actually cast. To vote for a candidate, a voter ticks two boxes in the row of that candidate; every other candidate-row only receives one tickmark. The voter is free to place the ticks in any column, as long as there is one row with two ticked boxes (her choice) and all other rows have one ticked box. Given the specific way of voting in 3BS, only a limited subset of the cast ballots can form a valid Threeballot with a given receipt (to be more precise, only those ballots combined with the receipt such that there is only one row with two tickmarks). For example, consider a receipt with a tickmark for every candidate. This can only be matched with one entirely blank ballot, and one ballot containing precisely one tickmark.

An obvious attack (already pointed out by the designer in [23]) is to agree a priori with the intruder on how to fill in the ballots (captured by class b conspiracy). The intruder can then easily verify if all three ballots are cast. This reduces privacy more strongly than a voter merely showing her receipt after the elections (class 1 conspiracy). This, in turn, gives less privacy than not showing the receipt: $cg_v^b(3BS, \gamma) \subseteq cg_v^1(3BS, \gamma) \subseteq cg_v(3BS, \gamma)$.

As pointed out in [22], 3BS can also be used in elections where voters are allowed to vote for multiple candidates. In this case, a Threeballot may contain multiple rows with two tickmarks. This means that the number of ballots forming a valid Threeballot with a given receipt is increased. As the number of valid combinations directly affects privacy, voter privacy is improved. In the framework, this improvement is precisely captured by the size of choice groups.

The high-level analysis of 3BS illustrates that the framework can quantify partial loss of privacy. Furthermore, the framework can be tailored to the class of conspiracy possible.

VII. Discussion

In the previous section, we have shown that our framework is able to quantify privacy loss in 3BS, where the receipt of a voter reduces the size of her choice group, but not necessarily to one. This kind of attack cannot be analysed by other formal approaches as discussed in Sect. I. Moreover, our framework can capture previously established privacy nullification attacks as well as a new class of privacy reduction attacks. An example of such new attacks is the *not-voted-for* attack, where the intruder learns that the voter did not vote for a party. This may be sufficient for reward (or retaliation, when not complying with the intruder's wishes). A similar problem manifests itself in Dutch national elections. In these elections, voters vote for persons, but every candidate is affiliated with one party. Effectively, voters end up voting for parties. Thus, voter privacy should extend to parties, and not just candidates. Another example is related to *coalition* between the intruder and several successfully conspiring voters. Based on the final result of the election and the known votes of these voters, the intruder can further reduce the privacy of honest voters. A simple example of such a scenario is when a conspiring voter cast the only vote for a particular candidate – then the intruder additionally learns that no other voter voted for that candidate. Finally, the result of an election provides a bias for a voter's choice – the probability of a voter having voted for the winner is higher than the probability of a voter having voted for a losing candidate. Our framework makes it possible to account for such information bias.

VIII. Conclusion

We have developed a framework to precisely measure voter-controlled privacy, with respect to different ways how one voter can share her knowledge with the intruder. This quantitative framework is based on knowledge reasoning and trace equivalences. It allows us to capture the exact meaning of receipt-freeness in the context of vote buying, and to detect attacks that has escaped the focus of published methods in the literature as well. There are several directions we would like to continue in the future.

1) The modelling language in this paper is quite limited. Behaviour of agents is described only by a sequence of events. We plan to extend the language to encompass control flow constructors (e.g. the conditional choice).
2) The conspiracy classes only investigate conspiring voters. We intend to extend the framework's transformation function to model conspiring authorities.
3) In the current paper, we have focused on receipt-freeness. We intend to extend our framework to formally define coercion-resistance.
4) We will apply our work in full detail to more voting systems. We believe that this quantitative approach to privacy in voting will identify more new attacks.
5) We plan to investigate the potential effects of various counting methods on privacy loss.

References

[1] J. Benaloh and D. Tuinstra, "Receipt-free secret ballot elections (extended abstract)," in *Proc. 26th ACM Symposium on the Theory of Computing*. ACM, 1994, pp. 544–553.

[2] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proc. 4th ACM Workshop on Privacy in the Electronic Society*. ACM, 2005, pp. 61–70.

[3] T. Okamoto, "An electronic voting scheme," in *Proc. 14th IFIP World Computer Congress, Conference on IT Tools*, 1996, pp. 21–30.

[4] B. Lee and K. Kim, "Receipt-free electronic voting through collaboration of voter and honest verifier," in *Proc. Japan-Korea Joint Workshop on Information Security and Cryptology*, 2000, pp. 101–108.

[5] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *Proc. 19th Conference on the Theory and Application of Cryptographic Techniques*, ser. LNCS, vol. 1807. Springer, 2000, pp. 539–556.

[6] M. Hirt, "Multi-party computation: Efficient protocols, general adversaries, and voting," Ph.D. dissertation, ETH Zurich, 2001.

[7] B. Lee and K. Kim, "Receipt-free electronic voting with a tamper-resistant randomizer," in *Proc. 4th Conf. on Information and Communications Security*, ser. LNCS, vol. 2513. Springer, 2002, pp. 389–406.

[8] S. Delaune, S. Kremer, and M. Ryan, "Coercion-resistance and receipt-freeness in electronic voting," in *Proc. 19th IEEE Computer Security Foundations Workshop*. IEEE Computer Society, 2006, pp. 28–42.

[9] ——, "Verifying privacy-type properties of electronic voting protocols," *Journal of Computer Security*, 2008, to appear. [Online]. Available: http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/DKR-jcs08.pdf

[10] S. Delaune, M. Ryan, and B. Smyth, "Automatic verification of privacy properties in the applied pi-calculus," in *Proc. 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, ser. IFIP Conference Proceedings, vol. 263. Springer, 2008, pp. 263–278.

[11] M. Backes, C. Hriţcu, and M. Maffei, "Automated verification of remote electronic voting protocols in the applied pi-calculus," in *Proc. 21st IEEE Computer Security Foundations Symposium*. IEEE Computer Society, 2008, pp. 195–209.

[12] H. Jonker and E. de Vink, "Formalising receipt-freeness," in *Proc. 9th Conference on Information Security*, ser. LNCS, vol. 3444. Springer, 2006, pp. 476–488.

[13] A. Baskar, R. Ramanujam, and S. Suresh, "Knowledge-based modelling of voting protocols," in *Proc. 11th Conference on Theoretical Aspects of Rationality and Knowledge*. ACM, 2007, pp. 62–71.

[14] J. Skripsky, "Minimal models for receipt-free voting," 2002, ubpublished manuscript.

[15] T. Okamoto, "Receipt-free electronic voting schemes for large scale elections," in *Proc. 5th Workshop on Security Protocols*, ser. LNCS, vol. 1361. Springer, 1997, pp. 25–35.

[16] R. Aditya, B. Lee, C. Boyd, and E. Dawson, "An efficient mixnet-based voting scheme providing receipt-freeness," in *Proc. 1st Conference on Trust and Privacy in Digital Business*, ser. LNCS, vol. 3184. Springer, 2004, pp. 152–161.

[17] A. Fujioka, T. Okamoto, and K. Ohta, "A practical secret voting scheme for large scale elections," in *Proc. 3rd Workshop on the Theory and Application of Cryptographic Techniques*, ser. LNCS, vol. 718. Springer, 1992, pp. 244–251.

[18] F. Garcia, I. Hasuo, W. Pieters, and P. van Rossum, "Provable anonymity," in *Proc. 3rd ACM Workshop on Formal Methods in Security Engineering*. ACM, 2005, pp. 63–72.

[19] S. Mauw, J. Verschuren, and E. de Vink, "A formalization of anonymity and onion routing," in *Proc. 9th European Symposium on Research Computer Security*, ser. LNCS, vol. 3193. Springer, 2004, pp. 109–124.

[20] T. Chothia, S. Orzan, J. Pang, and M. Torabi Dashti, "A framework for automatically checking anonymity with μCRL," in *Proc. 2nd Symposium on Trustworthy Global Computing*, ser. LNCS, vol. 4661. Springer, 2007, pp. 301–318.

[21] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 12, pp. 198–208, 1983.

[22] R. Rivest and W. Smith., "Three voting protocols: Threeballot, VAV, and Twin," in *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop*. USENIX, 2007.

[23] R. Rivest, "The threeballot voting system," MIT, Tech. Rep., 2006.