Bulletin Boards in Voting Systems: Modelling and Measuring Privacy

Hugo Jonker
University of Luxembourg
Computer Science and Communications

Jun Pang
University of Luxembourg
Computer Science and Communications

***Abstract*—Transparency is crucial to ensuring fair, honest elections. Transparency is achieved by making information (e.g. election result) public. In e-voting literature, this publication is often described in terms of a *bulletin board*. While privacy of voting systems has been actively studied in recent years, resulting in various analysis frameworks, to date there has not been an explicit modelling of bulletin board in any such framework. Privacy implications of bulletin boards are thus understudied.**

In this paper, we extend the semantics of the framework of Jonker, Mauw and Pang to model a bulletin board and capture coercion-resistance. The usage of the extended framework is illustrated by an application to the Prêt à Voter voting system. Moreover, we present an information-theoretical measure of privacy loss in elections.

I. Introduction

A prime requirement for election systems is to ensure voters have reason to trust the result of an election. Correctness alone cannot achieve this – a correctly functioning process need not inspire trust. Transparency of the voting process is the key to foster such trust. In evoting literature (e.g. [1]–[5]), such transparency is often achieved by making election information (such as the final result) public via a so-called bulletin board.

Similarly, privacy is of paramount importance for fair and honest elections. Without voter-privacy, a malicious agent can threaten or bribe voters, and find out whether or not they complied with his bidding. Such threats undermine the very purpose of voting. There exist several notions of privacy-type properties 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 [6] introduce receipt-freeness, which expresses that a voter cannot gain any information to prove to an intruder 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 [7], stating that a voter cannot cooperate with the intruder to prove how she voted.

Many voting systems have been proposed claiming to satisfy certain privacy properties [1]–[4], [7], [8]. A uniform approach is necessary to consistently evaluate such claims. Various frameworks have been proposed to this end (e.g. [9]–[12]). For instance, Delaune, Kremer and Ryan [11] formalise vote-privacy, receipt-freeness and coercion-resistance using observational equivalences of processes. Backes, Hriţcu and Maffei [10] use the same framework to have a more precise modelling of coercion-resistance and focus on remote electronic voting protocols. The quantitative framework by Jonker, Mauw and Pang [12] combine knowledge reasoning along the lines of [9] and trace equivalences of [10], [11] to formally model voting protocols and define vote-privacy and receipt-freeness for the voters. It can precisely measure privacy for voters by establishing choice groups. However, to this date, an explicit modelling of a bulletin board and its effects on voter's privacy has not been studied.

Contribution and paper organisation. In this paper, we first shortly discuss the usage of bulletin boards in voting and the setting of our work (Sect. II). Our main contribution is to extend the work by Jonker, Mauw and Pang [12] to account for bulletin boards, focusing on the syntax and semantics extensions (Sect. III and Sect. IV). We formalise privacy notions, in particular, we show how to formalise coercion-resistance in the framework in a natural way. In addition, we introduce and formalise a new privacy notion: forced vote spoiling (Sect. V). These extensions make the framework more complete and expressive for modelling and analysing existing voting systems. To illustrate the usage of our definitions, we apply the enhanced framework to the *Prêt à Voter* voting system by Ryan [4] (Sect. VI). Furthermore, we discuss an information-theoretical measure of privacy loss in elections by a few examples (Sect. VII) and conclude the paper with some future research directions (Sect. VIII).

II. Bulletin Boards and Our Setting

In e-voting literature, bulletin boards are used in any instance where public access to information is desired. Examples include publishing information from the election authorities, e.g. non-interactive zero-knowledge proofs [13], and announcing the final result. Furthermore, in some systems, the voter is expected to publish information. For example, in [3], the voter's vote is publicly announced. Thus, both voters and election authorities should be able to add information to the bulletin board, as well as obtain information from the bulletin board.

In line with the privacy framework of Jonker, Mauw and Pang [12], we aim to capture the effects of a bulletin board at the level of interactions between agents, not its internal workings. The original framework considers three different types of communication channels: public, (sender) anonymous,

and untappable. Thus, to add information to the bulletin board, we assume that any of the same types of channels may be used. This means that any publication sent over a public or anonymous channel can be blocked by the intruder. In addition to these three channels, sometimes the assumption is used that election officials can post directly to the bulletin board. For such direct connection, we introduce an *unblockable send*. Finally, we assume that the bulletin board is public, that is, information on the bulletin board may be read by any party as soon as the bulletin board has received the information.

We model the contents of a bulletin board as a list of terms, using function $list2set$ to convert lists to sets. A list is denoted as $L = (a_1, a_2, \ldots)$. Appending a_n to the end of list L is denoted as $L' = (L, a_n)$. Interaction with the bulletin board is modelled as regular interaction with an agent, though we purposfully omit an agent specification of the bulletin board. Note that this modelling of a bulletin board also implies no security guarantees on the part of the bulletin board (e.g., the board does not sign its contents), and its communications are subject to the intruder's whim (as per the standard Dolev-Yao intruder).Any desired security properties will have to arise from how the bulletin board is used, thus ensuring that this model does not introduce security where there is none.

The intruder is modelled as a standard Dolev-Yao intruder,who has full control of the public network (note that untappable channels are *not* under his control).

III. Syntax of the Framework

In this section, we recall the framework [12] and extend it to account for interaction with bulletin boards. Changes wrt. the original framework are marked in **boldface**.

We use $\mathcal{V}$ to denote the set of voters, who make a choice of their preferred candidate $\gamma(v) \in \mathcal{C}$. Similarly, Aut denotes the set of authorities. The terms communicated in a voting protocol are built up from variables in $Vars$, candidates in $\mathcal{C}$, random numbers in $Nonces$, and cryptographic keys in $Keys$, ranged over by var, c, n, k, respectively. The class $Terms$ of terms, ranged over by φ, is given by the BNF

$$\varphi ::= \mathrm{var} \mid c \mid n \mid k \mid (\varphi_1, \varphi_2) \mid \{\varphi\}_k.$$

Terms may be paired ((φ_1, φ_2)) or encrypted with a key ($\{\varphi\}_k$). The set of variables of a term φ is 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, $pk(a), sk(a)$ denote the public and secret key of agent a, respectively. Signing is denoted as encryption with the secret key.

Variables represent unspecified terms. An example is the voter's choice: it is represented by variable vc until instantiated. Variables are instantiated by substitution. The substitution of var by φ is denoted as $\mathrm{var} \mapsto \varphi$, the application of a substitution σ to a term φ as $\sigma(\varphi)$.

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

$$\begin{array}{ll}
\varphi \in T & \Longrightarrow T \vdash \varphi \\
T \vdash \varphi_1,\ T \vdash \varphi_2 & \Longrightarrow T \vdash (\varphi_1, \varphi_2) \\
T \vdash (\varphi_1, \varphi_2) & \Longrightarrow T \vdash \varphi_1 \\
T \vdash (\varphi_1, \varphi_2) & \Longrightarrow T \vdash \varphi_2 \\
T \vdash \varphi_1,\ T \vdash k & \Longrightarrow T \vdash \{\varphi_1\}_k \\
T \vdash \{\varphi_1\}_k,\ T \vdash k^{-1} & \Longrightarrow T \vdash \varphi_1
\end{array}$$

An agent's knowledge K is a set of terms closed under derivability. This is defined as $\overline{K} = \{\varphi \mid K \vdash \varphi\}$.

Terms are communicated between agents or send to/read from the bulletin board. These communications may occur over public, anonymous, untappable, or **unblockable** channels.[1] The class of communication events Ev is given by:

$$\begin{array}{l}
Ev = \{\ s(a,a',\varphi), r(a,a',\varphi), as(a,a',\varphi), ar(a',\varphi), ph(i), \\
\quad us(a,a',\varphi), ur(a,a',\varphi), \boldsymbol{ubs(a,a',\varphi), ipub(\varphi)} \\
\quad \mid a, a' \in Ag \cup \boldsymbol{\{BB\}}, \varphi \in Terms, i \in \mathbb{N}\ \},
\end{array}$$

where s, r, as, ar, us, ur denote sending and receiving over public, anonymous and untappable channels, respectively, ubs denotes sending over an unblockable channel, and $ph(i)$ denotes an agent ready to start phase i. Finally, $ipub$ denotes the intruder adding a term to the bulletin board.

The behaviour of an agent is determined by the order in which events occur. This order is defined by the agent's process. The class $Procs$ of processes, ranged over by P, is given by the BNF

$$\begin{array}{rcl}
P & ::= & \delta \mid ev.P \mid P_1 + P_2 \mid \\
& & P_1 \lhd \varphi_a = \varphi_b \rhd P_2 \mid ev.X(\varphi_a, \ldots, \varphi_n).
\end{array}$$

Here, δ denotes a deadlock, $ev.P$ denotes action prefix, $P_1 + P_2$ non-deterministic choice, and $P_1 \lhd \varphi_a = \varphi_b \rhd P_2$ conditional choice (behaving as P_1 if φ_a is syntactically equal to φ_b, otherwise as P_2). Finally, we have guarded recursion. We assume a class of process variables, which is ranged over by X. For every process variable X with arity n, there is a defining equation of the form $X(\mathrm{var}_1, \ldots, \mathrm{var}_n) = P$, with the syntactic requirement that the free variables of P are precisely $\mathrm{var}_1, \ldots, \mathrm{var}_n$. Without loss of generality, we assume a naming convention such that all free variables in the defining equation of a process variable are globally unique, thus limiting their scope to that defining equation. Substitution is extended to events and processes in the obvious manner.

A voting system specifies for each agent the agent's state. The state of an agent a is a tuple of its knowledge knw_a (a set of terms) and its behaviour (i.e. the order in which events are executed) as determined by its process P_a.

Definition 1 (voting system [12]): The class of voting systems $VotProt$ is of type $Ag \to (\mathcal{P}(Terms) \times Procs)$. A voting system $\mathcal{VS}$ may be instantiated with voter choices, as given by choice function $\gamma\colon \mathcal{V} \to \mathcal{C}$. This instantiation is denoted as $\mathcal{VS}^\gamma$, which, for each voter, substitutes the voter choice variable vc by the choice specified by γ in her process.

$$\mathcal{VS}^\gamma(a) = \begin{cases} \mathcal{VS}(a) & \text{if } a \notin \mathcal{V}, \\ (\ \pi_1(\mathcal{VS}(a)), \pi_2(\sigma(\mathcal{VS}(a)))\ ) & \text{else.} \end{cases}$$

Here, π_i denotes an extraction function that extracts the i^{th} component from a tuple, and $\mu = vc \mapsto \gamma(a))$.

[1] The distinction between these channels will become clear in Sect. IV.

IV. Formal Semantics

In this section, we highlight the extensions to the formal semantics of the framework. The operational semantics is defined in two layers. First, the semantics of an individual agent is defined in a context with an intruder with knowledge K_I and a bulletin board BB. Based on these semantics, we define the semantics of a voting system. The operational semantics of a voting system can be seen as the parallel composition of all agents. For the agent semantics, we present only a subset of the semantic rules. Given these, the remaining semantic rules can be easily adapted from the original framework.

A. Agent semantics

The semantics of agents describes the effect of the events on the agent state. Recall that agent state is defined as a tuple containing a knowledge set and a process. Furthermore, we consider the agent in context with the intruder and the bulletin board. Hence, for agent semantics we consider transitions of states of the form: $(K_I, \mathsf{BB}, knw_a, P_a)$, representing intruder knowledge, the bulletin board's contents, and the agent's state, respectively. Intruder knowledge is represented by a set of terms, the bulletin board by a list of terms, and the agent state by a tuple of agent knowledge knw_a and agent process P_a.

In the assumed intruder model, each tappable communication by an agent is a communication with the intruder. Hence, the semantic rules below take the intruder's knowledge into account. The states considered below thus consist of a tuple of intruder knowledge K_I and agent state.

There are some restrictions on the terms that may occur in an event. A term φ occurring in a send event must be closed ($\mathrm{fv}(\varphi) = \emptyset$) at the moment of sending.

1) publish: An agent may try to publish a closed term φ if and only if the agent can derive φ from his own knowledge. This is denoted as a send to the bulletin board BB. For a public publication $s(a, \mathsf{BB}, \varphi)$, the semantics rule is as follows.

$$\frac{knw_a \vdash \varphi \qquad \mathrm{fv}(\varphi) = \emptyset}{\begin{array}{c}(K_I, \mathsf{BB}, knw_a, s(a, \mathsf{BB}, \varphi).P_a) \\ (K_I \cup \{\varphi\}, \mathsf{BB}, knw_a, P_a)\end{array} \xrightarrow{s(a,\mathsf{BB},\varphi)}}$$

The rule for anonymous publication $as(a, \mathsf{BB}, \varphi)$ is similar: these terms also end up in the hands of the intruder (albeit without revealing the sender). Terms are added to the bulletin board by the intruder using $ipub(\varphi)$ (see system semantics), as public and anonymous channels are under the intruder's control. In case of an untappable publication, the intruder has less control: he cannot stop the publication, but he will (immediately) become aware of it (as the bulletin board is public). The difference with an unblockable send is that the intruder cannot identify the sender of an untappable message only from the communication event.[2] Untappable publications are modelled as follows:

$$\frac{knw_a \vdash \varphi \qquad \mathrm{fv}(\varphi) = \emptyset}{\begin{array}{c}(K_I, \mathsf{BB}, knw_a, us(a, \mathsf{BB}, \varphi).P_a) \\ (K_I \cup \{\varphi\}, (\mathsf{BB}, \varphi), knw_a, P_a)\end{array} \xrightarrow{us(a,\mathsf{BB},\varphi)}}$$

[2]Intruder observations are captured at the system semantics level.

Unblockable publications are modelled analoguous, using the $ubs(a, \mathsf{BB}, \varphi)$ action.

2) read: An agent can try to read the bulletin board. If successful, this extends his knowledge with the full contents of the bulletin board. Reading may be done anonymously or untappably, if the bulletin board supports such channels.

$$\frac{}{\begin{array}{c}(K_I, \mathsf{BB}, knw_a, r(a, \mathsf{BB}, \mathrm{var}).P_a) \\ (K_I, \mathsf{BB}, knw_a \cup list2set(\mathsf{BB}), P_a)\end{array} \xrightarrow{r(a,\mathsf{BB},list2set(\mathsf{BB}))}}$$

To capture the flavour of how process flow rules are updated with respect to the original framework, below we provide the semantic rule for guarded recursion.

3) guarded recursion: An invocation of process variable X with argument list $\varphi_1, \ldots, \varphi_n$ can be executed by agent a if the corresponding process P in the defining equation of X can execute, under the specified arguments.

$$\frac{\begin{array}{c}\sigma = \mathrm{var}_1 \mapsto \varphi_1 \circ \cdots \circ \mathrm{var}_n \mapsto \varphi_n \\ X(\mathrm{var}_1, \ldots, \mathrm{var}_n) = P \quad \mathrm{fv}(\varphi_1) = \ldots = \mathrm{fv}(\varphi_n) = \emptyset \\ (K_I, \mathsf{BB}, knw_a, \sigma(P)) \xrightarrow{ev} (K_I', \mathsf{BB}', knw_a', P')\end{array}}{(K_I, \mathsf{BB}, knw_a, X(\varphi_1, \ldots, \varphi_n)) \xrightarrow{ev} (K_I', \mathsf{BB}', knw_a', P')}$$

B. System semantics

The formalisation of the semantics at the system level remains largely unchanged with respect to the original framework. Nevertheless, as the core modelling of the bulletin board is at the level of the system semantics, we present the full system semantics. The main difference at this level is the addition of the intruder publishing event. To express this, we introduce the state of a voting system.

Definition 2 (voting system state): The state of a voting system is a tuple of intruder knowledge, a bulletin board (which is a list of terms) and a mapping of agents to agent states (a voting system), as follows.

$$State = \mathcal{P}(Terms) \times Terms^\star \times VotProt.$$

The knowledge and current process for each agent are given by $VotProt$. We denote the attribution of state (knw_a, P_a) to agent a as $a : (knw_a, P_a)$. The current state of agent a in system state (K_I, BB, S) is denoted as $a : (knw_a, P_a) \in S$. The initial state of voting system $\mathcal{VS}$ with respect to choice function γ is $(K_I^0, \varepsilon, \mathcal{VS}^\gamma)$, for initial intruder knowledge K_I^0 and the empty list ε.

The operational semantics of voting systems describe how the state of a voting system changes due to the interactions of its agents. The state of a voting system is given by the intruder knowledge and the state of each agent. Typically, the initial intruder knowledge contains public keys of all agents, compromised keys etc.

The operational semantics of voting systems in the context of a Dolev-Yao intruder (limited to public and anonymous channels) is given below. The semantic rules give rise to a labelled transition system (LTS), with labels denoting the events. Untappable communication is modelled as synchronous communication, hence the us and ur events are

replaced by uc events (denoting untappable communication) in the set of labels $Labels$ of the transition system: $Labels = \{uc(a,a',\varphi) \mid a,a' \in Ag \cup \mathbf{\{BB\}} \ \wedge \ \varphi \in Terms\} \cup Ev \setminus \{us(a,a',\varphi), ur(a,a',\varphi) \mid a,a' \in Ag \cup \mathbf{\{BB\}} \wedge \varphi \in Terms\}$. Both untappable communications and phase synchronisation are synchronous events by more than one agent. The other events are executed without synchronising with other agents. We distinguish the non-synchronous events as Ev_{nsync}, which is defined as follows.

$$Ev_{nsync} = \{\ s(a,a',\varphi), r(a,a',\varphi), as(,a',\varphi), ar(a',\varphi) \mid a,a' \in Ag \cup \mathbf{\{BB\}}, \varphi \in Terms\}.$$

The below system semantics uses the agent semantics to define the dynamic behaviour of the system. The rules may involve agents $a,b \in Ag$, which we omit from the premises of the rules. Note that the premise of the rules involves agent state transitions (a three-tuple of intruder knowledge, agent knowledge and agent process), and may specify restrictions on the system state (a mapping of agents to agent states).

1) non-synchronous events: The operational semantics for any $ev \in Ev_{nsync}$, including reading the bulletin board, is analoguous to the original framework.

$$\frac{(K_I, \mathsf{BB}, knw_a, P) \xrightarrow{ev} (K_I', \mathsf{BB}', knw_a', P') \quad ev \in Ev_{nsync} \quad a\colon (knw_a, P) \in S}{(K_I, \mathsf{BB}, S) \xrightarrow{ev} (K_I', \mathsf{BB}', \{a\colon (knw_a', P')\} \cup S \setminus \{a\colon (knw_a, P)\})}$$

2) untappable communications: Untappable communications between agents a,b where $b \neq \mathsf{BB}$ are modelled as synchronous communications between a and b.

$$\frac{(K_I, \mathsf{BB}, knw_a, P_a) \xrightarrow{us(a,b,\varphi)} (K_I, \mathsf{BB}, knw_a, P_a') \quad (K_I, \mathsf{BB}, knw_b, P_b) \xrightarrow{ur(a,b,\varphi)} (K_I, \mathsf{BB}, knw_b', P_b') \quad s_0 = \{a\colon (knw_a, P_a), b\colon (knw_b, P_b)\} \quad s_0 \subseteq Sb \neq \mathsf{BB}}{(K_I, \mathsf{BB}, S) \xrightarrow{uc(a,b,\varphi)} (K_I, \mathsf{BB}, \{a\colon (knw_a, P_a'), b\colon (knw_b', P_b')\} \cup S \setminus s_0)}$$

3) publishing: In the case of public and anonymous channels, a term is added to the bulletin board directly by the intruder (instead of agents), as follows.

$$\frac{K_I \vdash \varphi \qquad \mathrm{fv}(\varphi) = \emptyset}{(K_I, \mathsf{BB}, S) \xrightarrow{ipub(\varphi)} (K_I, (\mathsf{BB}, \varphi), S)}$$

In the case of an untappable send to the bulletin board, the semantics rule is as follows.

$$\frac{a\colon (knw_a, P_a) \in S \quad (K_I, \mathsf{BB}, knw_a, P_a) \xrightarrow{us(a,\mathsf{BB},\varphi)} (K_I', \mathsf{BB}', knw_a, P_a')}{(K_I, \mathsf{BB}, S) \xrightarrow{uc(a,\mathsf{BB},\varphi)} (K_I', \mathsf{BB}', \{a\colon (knw_a, P_a')\} \cup S \setminus \{a\colon (knw_a, P_a)\})}$$

The case for an unblockable send is analoguous.

4) phase synchronisation: As in the original framework, a phase transition can only be executed if all authorities are ready for it. The phase transition is then executed by all authorities and by all agents ready *and* willing to do so. Its semantics rule is referred to the original framework [12].

The system semantics gives rise to an LTS for each voting system. An execution of the system is a path in this LTS. A path is represented by a sequence of labels and is called a trace. The class of traces Tr consists of sequences of labels. The traces of voting system $\mathcal{VS}$ instantiated with choice function γ (denoted $\mathcal{VS}^\gamma$) are given by

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

We denote the intruder knowledge and the bulletin board in the last state of a trace t as K_I^t and BB^t, respectively.

Traces model the dynamic behaviour of the system. The next section determines the privacy of a given voter in a given trace. This is then extended to establish the privacy of a voter in a voting system.

C. Quantifying Voter-Privacy

The extended formal model developed above enables us to express if an intruder can distinguish two executions of the system. In comparison with the original framework, the only difference is that some of the newly introduced labels are not (completely) visible to the intruder. As before, these changes have been marked in **boldface**.

The distinguishing ability of the intruder is formalised as the intruder's ability to reinterpret terms: two terms φ_a, φ_b are indistinguishable to the intruder if the intruder may, given his knowledge, reinterpret the one as the other and vice versa (for precise definitions, see [12], [14]). This is captured by a reinterpretation function ρ, used e.g. as $\varphi_a = \rho(\varphi_b)$.

Some events in a trace are hidden from the intruder. In particular, the intruder cannot see any communications over untappable channels, nor the sender of an anonymous communication. The observable part of a trace t is captured by the function $obs(t)$ (see [12]).

Definition 3 (trace indistinguishability): Traces t, t' are indistinguishable for the intruder, notation $t \sim t'$ iff there exists a reinterpretation ρ such that the traces are equal under ρ, and the final intruder knowledge as well as the final contents of the bulletin board in both traces (denoted K_I^t, BB^t for trace t, respectively) are equal under ρ as well. Thus, $t \sim t'$ iff

$$obs(t) = \rho(obs(t')) \wedge \overline{K_I^t} = \rho(\overline{K_I^{t'}}) \wedge \mathbf{BB^t = \rho(BB^{t'})}.$$

This is extended to distinguishing sets of traces as follows.

Definition 4 (choice group [12]): Given voting system $\mathcal{VS}$, choice functions γ, γ' are indistinguishable to the intruder, notation $\gamma \simeq_{\mathcal{VS}} \gamma'$ iff

$$\forall t \in Tr(\mathcal{VS}^\gamma)\colon \exists t' \in Tr(\mathcal{VS}^{\gamma'})\colon t \sim t' \quad \wedge \quad \forall t \in Tr(\mathcal{VS}^{\gamma'})\colon \exists t' \in Tr(\mathcal{VS}^\gamma)\colon t \sim t'$$

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. Formalising Privacy

A conspiring voter behaves differently from a regular voter, as she will communicate with the intruder in certain circumstances. The original framework explored the ways in which voters can conspire in depth. Here we only recall two options that have a bearing on the case study. In each of these two cases, the conspiring voter v shares all her knowledge with the intruder (which can be expressed as $s(v, I, knw_v)$). The two cases are as follows:

1) v shares her knowledge at the end of the protocol,
2) v shares her knowledge at the beginning of the protocol.

In the first case, the last action of v is to share her knowledge, while in the second case, it is her first action. In line with the original framework, we write $\Theta_i(v,\mathcal{VS}), i \in \{1,2\}$ to denote a voting system $\mathcal{VS}$ where voter v is a conspirator of type 1 or type 2. For voting system $\mathcal{VS}$, and choice function γ, the choice group of conspiring voter v with respect to different conspiracy types $i \in \{1,2\}$, is given by

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

Juels, Catalano and Jakobsson [7] introduce the privacy-related concept *coercion-resistance*. This extends beyond reducing privacy of the vote. It includes receipt-freeness (which is captured by the framework) as well as prevention of:

- *forced abstention:* the intruder prevents the voter from voting.
- *simulation attack:* the voter gives her private keys to the intruder, who votes in her stead.
- *forced random voting:* the intruder forces the voter to vote for a random entry[3] in a list of encrypted candidates.

These attacks can be captured in the framework as follows.

a) Forced abstention: Forced abstention is trivial if the intruder has a full view of the network (the intruder knows who participated). Hence, forced abstention only makes sense if the intruder has incomplete knowledge of the communications. Thus, any non-trivial case of forced abstention needs to consider a set of voters whose interaction is not observed. Correspondingly, we write $OA \subseteq Ag$ for the set of observed agents. This limited view (captured by Π_{OA}) constrains the intruder's ability to distinguish two traces. Now, two traces t and t' are constrained-view indistinguishable with respect to OA (notation $t \sim_{OA} t'$), if there exists a reinterpretation ρ such that the observed traces are reinterpretable, and the final content of the bulletin board is reinterpretable as well. Thus, $t \sim_{OA} t'$ iff

$$\Pi_{OA}(t), \Pi_{OA}(t') \in Tr(\mathcal{VS}) \ \wedge\ \overline{K_I^{\Pi_{OA}(t)}} = \rho(\overline{K_I^{\Pi_{OA}(t')}}) \wedge \Pi_{OA}(obs(t)) = \rho(\Pi_{OA}(obs(t'))) \ \wedge\ \mathsf{BB}^t = \rho(\mathsf{BB}^{t'})$$

Here, $\Pi_{OA}(t)$ is the trace derived from t which consists only of actions by agents $\in OA$, i.e.:

$$\Pi_{OA}(\ell \cdot t) = \begin{cases} \ell \cdot \Pi_{OA}(t) & \text{if } \ell \in Ev_{OA} \\ \Pi_{OA}(t) & \text{otherwise} \end{cases}$$

where Ev_{OA} is the set of all events executed by agents $\in OA$ with any communication partner. We write $\gamma \simeq_{\mathcal{VS}}^{OA} \gamma'$ for choice function indistinguishability and $cg_v^{OA}(\mathcal{VS},\gamma)$ for the choice group of voter v in voting system $\mathcal{VS}$ based on the above definition of trace indistinguisability $t \sim^{OA} t'$. Note that $\gamma \simeq_{\mathcal{VS}}^{Ag} \gamma'$ is equivalent to $\gamma \simeq_{\mathcal{VS}} \gamma'$, and therefore cg_v^{Ag} is equivalent to cg_v.

We express absence of forced abstention for a given set of observed agents OA as follows: voter v cannot be forced to abstain if her choice group contains, but is not limited to, abstention. By denoting abstention as $\gamma(v) = \perp$, this is formalised as $\perp \in cg_v^{OA}(\mathcal{VS},\gamma) \ \wedge\ \left|cg_v^{OA}(\mathcal{VS},\gamma)\right| > 1$.

b) Simulation attacks: In a simulation attack, the intruder casts a vote himself. Simulation attacks are resisted if the intruder cannot tell whether the vote he cast affects the result or not. We extend the domain of γ to include the intruder I. The extended set is referred to as $\mathcal{V}_I$. The intruder's choice group cg_I then captures the intruder's uncertainty about his own vote. Using this, a system is simulation-attack-resistant if the intruder cannot tell whether his vote is counted or not, i.e.

$$\{\perp, \gamma(I)\} \subseteq cg_I(\mathcal{VS},\gamma).$$

c) Forced random voting: In forced random voting attacks, the intruder forces the voter to vote randomly. This means that whenever the voter process can make a choice, either conditionally or non-deterministically, the intruder instructs the voter how to proceed. This can be expressed in terms of the framework by rewriting every process P that denotes a choice for a specific agent. Let Δ_{rnd} be a process transformation function for forcing a specific agent's choices. Then we have, for any process representing a choice (i.e. any process P such that $P = P_1 + P_2$ or $P = P_1 \lhd \varphi_a = \varphi_b \rhd P_2$),

$$\Delta_{rnd}(v,P) = \begin{array}{l} r(I,v,\mathrm{var}).\Delta_{rnd}(v,P_1) \\ \quad \lhd \mathrm{var} = \mathit{true} \ \rhd\ \Delta_{rnd}(v,P_2), \end{array}$$

where v is the agent forced, var is a fresh variable and $true$ is a constant term $\in Terms$. Using this, the choice group of the voter can be determined as before.

A. Forced vote spoiling

In addition to coercion-resistance, we distinguish a new privacy attack "forced vote spoiling". Whereas in forced abstention the intruder aims to force a voter not to communicate, here the intruder aims to force the voter to produce an invalid ballot. If the intruder cannot observe a voter at all times during the elections, enforcing abstention may not be possible, but requiring proof of an invalid vote may very well be. If a

[3]The intruder does not need to know which candidate is chosen, as long as it is a random choice. This attack forces a more uniform distribution of votes, benefitting unpopular candidates at the expense of popular candidates.

system accepts invalid ballots and allows these to be made public, this can be a devastating attack on voter privacy. We express this by introducing $\top$ as an invalid ballot. If necessary, various different invalid ways of filling in the ballot may be expressed as $\top, \top', \top'', \ldots$. Using this, a system resists forced vote spoiling if the intruder cannot tell whether a voter voted invalid or not, i.e. $\{\top, \gamma(v)\} \subseteq cg_v(\mathcal{VS}, \gamma)$.

VI. Case Study: Voter Privacy in Prêt à Voter

The Prêt à Voter (PaV) voting system [4] aims to combine privacy and verifiability. A ballot in PaV has two columns: a per-ballot randomized listing of the candidates on the left, and space for a mark by the voter on the right. Below the right column is an *onion*, a ciphertext which decrypts to the left-hand side order of candidates.

After authenticating, the voter chooses a random envelope containing a ballot and enters a voting booth, where she marks her chosen candidate. Still inside the booth, she separates the two columns of the ballot and destroys the left-hand column. The right-hand column only contains the voter's mark and the onion, and is deposited in the ballot box. The voter receives an official copy of the ballot as a receipt. After the election is closed, the ballots are mixed using a mixnet, and then (threshold) decrypted, after which the result is announced.

Privacy of PaV: Voter-privacy in PaV requires that the order of candidates is only known to the voter, an assumption which the case study will point out. PaV can defeat forced random voting by allowing the voter to request new ballots until the desired order is obtained [15]. Forced abstention and simulation are handled by polling station procedures. Forced vote spoiling is achieved by requiring the voter to return with an official receipt of a spoilt ballot. This attack may be defeated by disallowing invalid votes. However, this would force all voters to cast valid votes.

A. Modelling Prêt à Voter

Fig. 1 offers a graphical representation of our PaV model, in which phase transitions are denoted by dotted lines. First, the voter authenticates herself and requests a random ballot i by sending i signed. The registrar sends ballot i, which contains a permutation π_i and the corresponding onion (π_i encrypted for the counter). The voter selects her preferred candidate $\gamma(v)$, and sends the right-hand column (modelled as a pair containing the index corresponding to her candidate, $\pi_i(\gamma(v))$, and the onion) to the registrar. The registrar returns an authenticated copy to the voter.

After the voting closes, the registrar publishes all the received votes. We assume the Registrar has an untappable connection with the bulletin board (meaning that the intruder cannot interfere with this). When this phase is finished, the counter reads all ballots from the bulletin board, mixes them, decrypts them and publishes the result. As this case study focuses on privacy, not verifiability, we view this latter process as merely sending the decrypted ballots to the bulletin board.

Note that the above model does not capture the provision that the voter destroys the left column of the ballot. This cannot be expressed in the framework and, in general, cannot be assured technically. Therefore it must be ensured via procedural means. Consequently, we expect to find that the cryptographic measures of PaV by themselves are insufficient to guarantee privacy in the analysis below, and thus we expect the analysis will find a privacy risk.

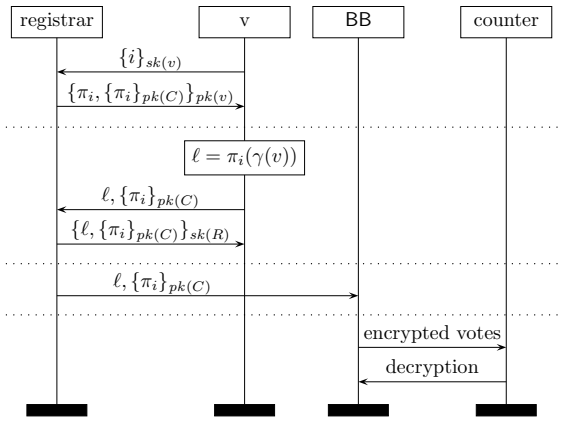

Fig. 1. Model of voter interaction in PaV

B. Measuring privacy of PaV

We show that in any non-trivial setting,[4] the choice group size of any non-unanimous choice function > 1. This implies that PaV does offer at least some privacy in non-trivial settings.

Lemma 1 (privacy of PaV): Suppose $|\mathcal{V}| > 1$ and $|\mathcal{C}| > 1$. Then for any choice function γ_1 such that there are voters va, vb such that $\gamma_1(va) \neq \gamma_1(vb)$, we have $|cg(PaV, \gamma_1)| > 1$.

Lemma 2 (voter-controlled privacy of PaV): For any non-trivial setting (i.e. $|\mathcal{V}| > 1, |\mathcal{C}| > 1$), the following holds:

1) PaV is *not* resistant against type 1 conspiring behaviour.
2) PaV is *not* resistant against type 2 conspiring behaviour.

To conserve space, we sketch a proof of the second lemma; detailed proofs are available in [16].

Proof: (type 2 conspiracy) A process modelling type 2 conspiring behaviour begins with sending her knowledge to the intruder ($s(v, I, knw_v)$). Note that $sk(v) \in knw_v$. Therefore, the intruder can open the ballot as sent by the Registrar to the voter ($\{\pi_i, \{\pi_i\}_{pk(C)}\}_{pk(v)}$). Therefore, the intruder knows which candidate order π_i the voter received. Consequently, the intruder is then able to determine precisely how the voter voted. The same reasoning holds for type 1 conspiracy (sharing final knowledge). ■

While Lemma 1 shows that PaV offers some privacy, the proof of Lemma 2 indicates that the permutation π_i puts privacy at risk. However, in PaV, the risk of exposing the link between voter and π_i is mitigated by procedural means (shredding π_i). This has an impact on adapting PaV for remote voting, where the lack of a controlled environment means that PaV cannot ensure that the candidate order cannot be linked to a voter. One method to alleviate this is to prove the order of the candidates to the voter using designated verifier

[4] A setting with more than one candidate and more than one voter.

proofs. To prevent any information leakage, such proofs should be communicated over untappable channels. A voting system along these lines, predating PaV, was described in [3].

VII. A Privacy Measure

The concept of choice group quantifies voter's privacy, but is too imprecise – it doesn't account for distribution of votes. If a voter's choice group only contains one candidate who received votes, that voter has no privacy, irrespective of the size of the choice group. To address this issue, we propose a new privacy measure using *relative entropy*. Our method quantifies the amount of probabilistic privacy information revealed to the intruder if a voter cooperates with him according to one of the conspiring behaviour classes [12].

Definition 5 (Relative entropy [17]): Let θ, θ' be two discrete probability distributions on a set S. The *relative entropy* of θ' w.r.t. θ is defined by

$$D(\theta', \theta) = \sum_{s \in S} \theta'(s) \cdot \log_2 \frac{\theta'(s)}{\theta(s)}.$$

Intuitively, the larger D is, the more information η' leaks compared to η. We assume a convention $0 \log_2 0 = 0$, and require that the domains of θ and θ' are the same, i.e., $dom(\theta) = dom(\theta')$. In general, $D(\theta', \theta) \neq D(\theta, \theta')$, so relative entropy is not a true metric. It does satisfy several important metric-properties, e.g., it is always non-negative, and equals zero only if $\theta = \theta'$.

Our main idea is to measure how much information the intruder can obtain after interacting with a compromised voter during the election, together with the information he observes from the election results published on the bulletin board. This boils down to calculating the relative entropy based on the published election results and the choice group computed for a particular voter.

Example 1 (Dutch elections): Consider the results of the 2010 parliamentary elections in the Netherlands[5]. Suppose the choice group of voter v only includes religious parties, in casu: CDA, CU, SGP and EPN. If the distribution of votes η was uniform over the whole elections, i.e. $\eta(\text{CDA}) = \ldots = \frac{1}{18}$, then knowing the vote is for a religious candidate induces a new distribution η' such that for all non-religious parties c, $\eta'(c) = 0$. The relative entropy is then $D(\eta', \eta) = 2.17$. However, votes were not uniformly distributed, but as follows. Total: $9,416,001$; of which CDA: $1,281,886$; CU: $305,094$; SGP: $163,581$ and EPN: 924. Using these numbers for distribution η (i.e. $\eta(\text{party}) = \frac{\#\text{votes for party}}{9,416,001}$), we obtain $D(\eta', \eta) = 4.75$. Thus, while the choice group did not change, the privacy loss more than doubled when the actual distribution of votes was taken into account.

VIII. Conclusion

We extended the framework of [12] to account for information made public via bulletin boards and to capture coercion-resistance (somewhat similar to the approaches by Delaune *et al.* [11] and Backes *et al.* [10]). We illustrated applicability of the extended framework by analysing Prêt à Voter. In addition, we proposed relative entropy as a privacy measurement in voting. We showed that combining choice group with election result can cause a significant reduction in privacy, even when some privacy remains. These ingredients together provide a powerful analysis tool to evaluate privacy of voting systems.

For future directions, we are interested in further applications of information-theoretic analysis to privacy of voting systems. This may, in particular, be used to investigate the effects of various counting methods and ballot forms on privacy loss, such as the *Italian attack* (see e.g. [6]). Furthermore, the framework can be extended to model conspiring authorities (an extension from conspiring voters). Finally, we are interested in modelling how a coalition of voters can execute a defensive strategy against a coercer requesting specific behaviour.

[5] Available from http://www.verkiezingsuitslagen.nl/

References

[1] A. Fujioka, T. Okamoto, and K. Ohta, "A practical secret voting scheme for large scale elections," in *Advances in Cryptology – AUSCRYPT '92*, ser. LNCS, vol. 718. Springer, 1992, pp. 244–251.

[2] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," in *Advances in Cryptology – EUROCRYPT'97*, ser. LNCS, vol. 1233. Springer, 1997, pp. 103–118.

[3] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *Advances in Cryptology – EUROCRYPT 2000*, ser. LNCS, vol. 1807. Springer, 2000, pp. 539–556.

[4] P. Y. A. Ryan, "A variant of the Chaum voter-verifiable scheme," in *Proc. Workshop on Issues in the Theory of Security*, 2005, pp. 81–88.

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

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

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

[8] D. Chaum, "Secret-ballot receipts: True voter-verifiable elections," *IEEE Security & Privacy*, vol. 2, no. 1, pp. 38–47, 2004.

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

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

[11] S. Delaune, S. Kremer, and M. D. Ryan, "Verifying privacy-type properties of electronic voting protocols," *Journal of Computer Security*, vol. 17, no. 4, pp. 435–487, 2009.

[12] H. L. Jonker, S. Mauw, and J. Pang, "A formal framework for quantifying voter-controlled privacy," *Journal of Algorithms in Cognition, Informatics and Logic*, vol. 64, no. 2-3, pp. 89–105, 2009.

[13] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications (extended abstract)," in *Proc. 20th Annual ACM Symposium on Theory of Computing*. ACM, 1988, pp. 103–112.

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

[15] D. Chaum, P. Y. A. Ryan, and S. A. Schneider, "A practical voter-verifiable election scheme," in *Proc. 10th ESORICS*, ser. LNCS, vol. 3679. Springer, 2005, pp. 118–139.

[16] H. Jonker, "Security matters: Privacy in voting and fairness in digital exchange," Ph.D. dissertation, Eindhoven University of Technology and University of Luxembourg, August 2009.

[17] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.