

A formal taxonomy of privacy in voting protocols

Jannik Dreier, Pascal Lafourcade, Yassine Lakhnech

▶ **To cite this version:**

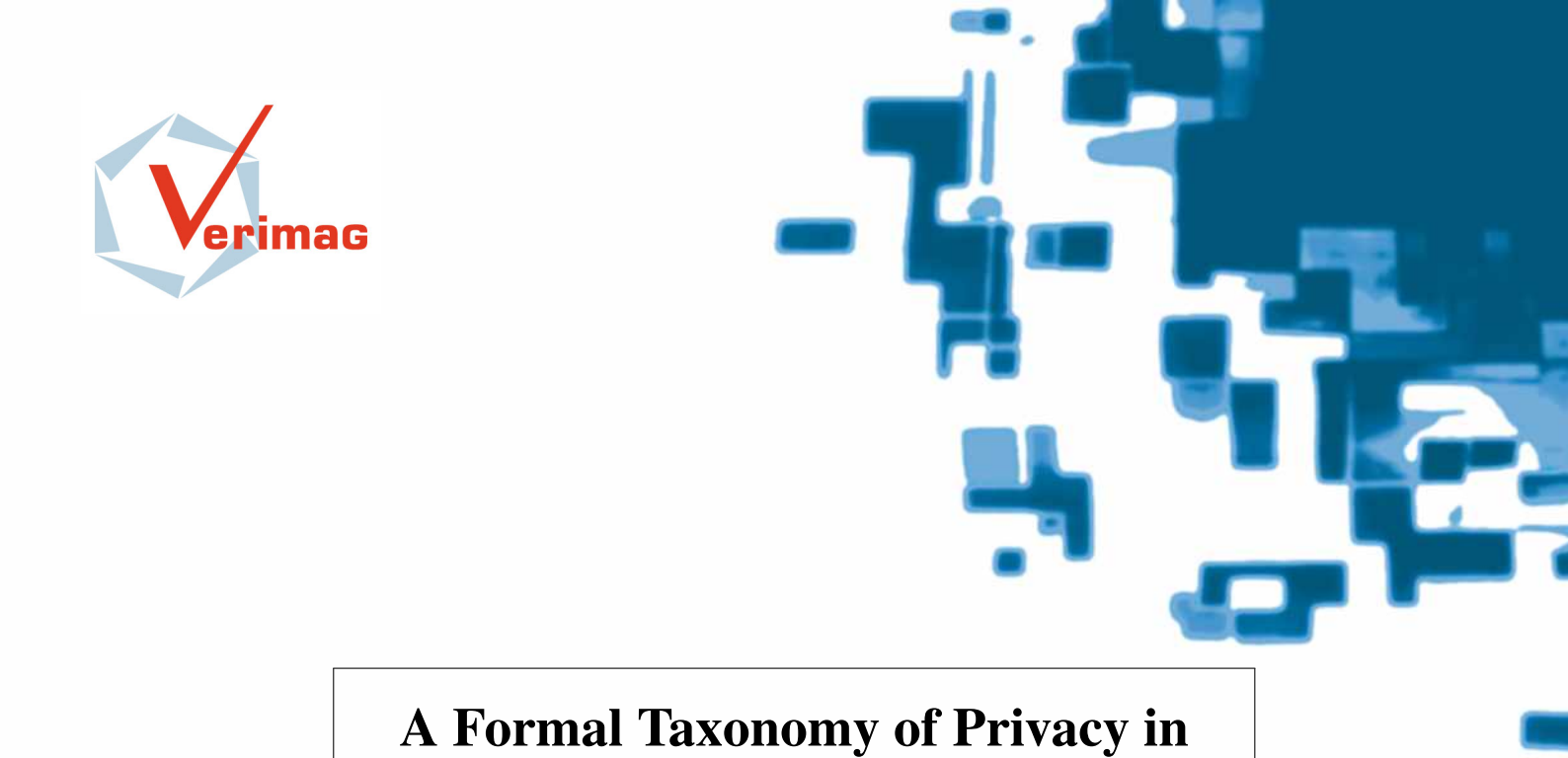
Jannik Dreier, Pascal Lafourcade, Yassine Lakhnech. A formal taxonomy of privacy in voting protocols. [Technical Report] VERIMAG UMR 5104, Université Grenoble Alpes, France. 2011. hal-01338067

HAL Id: hal-01338067
https://hal.science/hal-01338067

Submitted on 29 Jun 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



A Formal Taxonomy of Privacy in Voting Protocols

Jannik Dreier, Pascal Lafourcade, Yassine Lakhnech

Verimag Research Report n^o TR-2011-10

October 2011

Reports are downloadable at the following address
http://www-verimag.imag.fr



Unité Mixte de Recherche 5104 CNRS - INPG - UJF
Centre Equation
2, avenue de VIGNATE
F-38610 GIERES
tel : +33 456 52 03 40
fax : +33 456 52 03 50
http://www-verimag.imag.fr

A Formal Taxonomy of Privacy in Voting Protocols

Jannik Dreier, Pascal Lafourcade, Yassine Lakhnech

October 2011

Abstract

Privacy is one of the main issues in electronic voting. We propose a modular family of symbolic privacy notions that allows to assess the level of privacy ensured by a voting protocol. Our definitions are applicable to protocols featuring multiple votes per voter and special attack scenarios such as vote-copying or forced abstention. Finally we employ our definitions on several existing voting protocols to show that our model allows to compare different types of protocols based on different techniques, and is suitable for automated verification using existing tools.

Keywords: Security, Electronic Voting, Privacy, Anonymity, Formal Verification, Applied Pi Calculus.

How to cite this report:

```
@techreport {TR-2011-10,
  title = {A Formal Taxonomy of Privacy in Voting Protocols },
  author = {Jannik Dreier, Pascal Lafourcade, Yassine Lakhnech},
  institution = {{Verimag} Research Report},
  number = {TR-2011-10},
  year = {2011}
}
```

1 Introduction

Electronic voting systems have been designed and employed in practice for several years. However their use in general elections is controversial due to their security issues [10, 26, 9, 32]. Researchers have identified numerous security properties that are required for secure voting systems and protocols. These properties can be classified into three categories: i) Correctness and robustness properties, ii) verifiability and iii) privacy properties. Typical correctness and robustness properties include:

- *Eligibility*: Only the registered voters can vote, and nobody can submit more votes than allowed (typically only one).
- *Fairness*: No preliminary results that could influence other voters' decisions are made available.
- *Robustness*: The protocol can tolerate a certain number of misbehaving voters.

Verifiability is usually split into two properties:

- *Individual Verifiability*: Each voter can check whether his vote was counted correctly.
- *Universal Verifiability*: Anybody can verify that the announced result corresponds to the sum of all votes.

Last different levels of privacy can be achieved:

- *Vote-Privacy*: The votes are kept private. This can also be modeled as an unlinkability between the voter and his vote.
- *Receipt-Freeness*: A voter cannot construct a receipt which allows him to prove to a third party that he voted for a certain candidate. This is to prevent vote-buying.
- *Coercion-Resistance*: Even when a voter interacts with a coercer during the entire voting process, the coercer cannot be sure whether he followed his instructions or actually voted for another candidate.

However the design of complex protocols to fulfill all these partly antipodal requirements [7] is notoriously difficult and error-prone. To avoid bugs and analyze protocols, formal verification methods are an ideal tool and have been used in security and safety critical system for several years. In the area of voting protocols, many different formal models and definitions of the above mentioned properties have been proposed and used successfully to discover bugs (e.g. in Helios [29]). However, since the structure, setting and basic design of voting protocols can be quite different depending on the primitives used, many of these definitions are tailored to fit a specific (sub-)group of protocols. Sometimes a protocol can be proved secure in one model, but not in another. This hinders the objective comparisons of protocols.

In particular in the area of privacy properties - which we discuss in this paper - there is a great variety of models. Moreover, recent research shows that some existing definitions might be insufficient: Smyth and Cortier [29] pointed out that the ability to copy another voter's vote can enable attacks on privacy.

Our Contributions. We provide the three following contributions:

1. We propose a new family of privacy notions which allows to assess of the level of privacy provided by a voting protocol. These notions are based on formal definitions of the classical notions (Vote-Privacy, Receipt-Freeness and Coercion-Resistance) in the applied pi calculus [1], with a refined protocol model and including new attacks. The resulting family gives a deep understanding of the different levels and requirements for privacy. Additionally – by including a generalization of the notion of "Vote-Independence" [15] – our notions deal with attacks based on vote-copying that were not captured in the model by Delaune et al.
2. A deep understanding of privacy properties, and in particular the relationship between the different notions, is a prerequisite for the correct design of voting protocols. We provide a thorough comparison of existing and new notions.

3. Using several case studies [17, 27, 19, 4] we show that our model allows the analysis of different types of protocols. To automatically analyze protocols we use ProSwapper [20] and ProVerif [3]. Using these tools we can automate some proofs of Vote-Independence which were done by hand previously [16], and check a protocol supporting multiple votes (a variant of [27]).

Related Work. Previous research on formal verification of voting protocols concerned privacy properties (privacy, receipt-freeness and coercion-resistance) [11, 12, 25, 2, 15, 16], election verifiability [30, 21], or both [19, 18].

Although the informal specifications of the properties are very general, most of the formal models and definitions in the literature are tailored to a specific type of protocols. Many protocols where in fact developed together with their own definitions (e.g. [25, 19, 18]) and analyzed by hand in the original paper.

Juels et al. [19] (which became the bases for Civitas [8]) were the first to give a formal, but computational definition of coercion-resistance. It was later translated to the applied pi calculus and automated using ProVerif [2]. However – as their protocol is based on voting "credentials" – credentials also appear in the definition. Their model is thus unsuitable for protocols that do not use credentials (e.g. Bingo Voting [4] or the protocol by Lee et al. [24]).

More general definitions were developed by Delaune, Kremer and Ryan [11, 12]. They express different levels of privacy as observational equivalence in the applied pi calculus [1]. An attacker should not be able to distinguish one case in which the voter complies with the coercer's instructions and another in which he only pretends to do so and votes as he wishes. Unfortunately their definitions are too strong for the protocol proposed by Juels et al.: since in one case the targeted voter complies and posts only one correct ballot, and in the other he secretly posts his actual ballot and a fake one to cheat the coercer, both cases can be distinguished by counting the ballots.

Smyth and Cortier [28, 29] showed that being able to copy votes can compromise privacy if the number of participants is small or a noticeable fraction of voters can be corrupted. For example in the case of three voters, the third voter can try to copy the first voter's vote and submit it as his vote. This will result in (at least) two votes for the candidate chosen by the first voter and his choice can thus be inferred from the result. They also formally analyzed ballot secrecy in Helios using an adaption of the model by Delaune, Kremer and Ryan. However it was shown that, in general, the DKR model is not sufficient to capture vote-independence. For example the protocol by Lee et al. [24] was shown to be coercion-resistant in this model, despite its vulnerability to vote-copy attacks [15, 16].

Küsters and Truderung [22] proposed a first model independent definition of coercion-resistance for voting protocols. Their definition has to be instantiated using a concrete formal model. The exact security level can be defined with respect to certain chosen goals, and excluding explicit special cases. In contrast to our family of notions, their definition is based on traces and not bisimulations, and they only define coercion-resistance (in particular no simple vote-privacy).

Langer et al. [23] developed verifiability definitions and privacy notions based on (un-)linkability between a voter and his vote. Similarly to Küsters and Truderung, their definitions have to be instantiated with a concrete formal process and attacker model.

Computational definitions of receipt-freeness [6] and coercion-resistance [31] that can be applied to other applications than voting have also been proposed. Completely application-independent anonymity notions were proposed by Bohli and Pashalidis [5]. Although their definitions are very general, the application on voting protocols results in – for this context – rather unusual privacy notions (Pseudonymity etc.), compared to the classic properties such as receipt-freeness or coercion-resistance.

Outline of the Paper. In the next section, we give a brief introduction of the applied pi calculus and develop our model of a voting process. Section 3 starts by explaining informally our privacy notions and subsequently gives the formal definitions. In Section 4, we discuss the relationship between the different notions and explain the hierarchy implied by the definitions. We analyze several case studies to illustrate our definitions in Section 5. In the last section, we conclude and discuss future work.

2 Preliminaries

In the first part of this section, we introduce the applied pi calculus. We use it to model voting protocols and express privacy properties. In the second part, we define our model of a voting protocol in the applied pi calculus.

2.1 The Applied Pi Calculus

The applied pi calculus [1] is a formal language to describe concurrent processes. The calculus consists of *names* (which typically correspond to data or channels), *variables*, and a *signature* Σ of *function symbols* which can be used to build *terms*. Functions typically include encryption and decryption (for example $\mathtt{enc}(message, key)$, $\mathtt{dec}(message, key)$), hashing, signing etc. Terms are correct (i.e. respecting arity and sorts) combinations of names and functions. To model equalities we use an equational theory E which defines a relation $=_E$. A classical example which describes the correctness of symmetric encryption is $\mathtt{dec}(\mathtt{enc}(message, key), key) =_E message$.

There are two types of processes in the applied pi calculus: *plain processes* and *extended processes*. Plain processes are constructed using the following grammar:

$P, Q, R :=$	plain processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction ("new")
$\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$	conditional
$\mathtt{in}(u, x)$	message input
$\mathtt{out}(u, x)$	message output

Extended processes are plain processes or active substitutions:

$A, B, C :=$	active processes
P	plain process
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^M/_x\}$	active substitution

The substitution $\{^M/_x\}$ replaces the variable x with term M. We denote by $fv(A)$, $bv(A)$, $fn(A)$, $bn(A)$ the free variables, bound variables, free names or bound names respectively. A process is *closed* if all variables are bound or defined by an active substitution.

The *frame* $\Phi(A)$ of an extended process A is obtained when replacing all plain processes in A by 0. This frame can be seen as a representation of what is statically known to the exterior about a process. The domain $\mathtt{dom}(\Phi)$ of a frame Φ is the set of variables for which Φ defines a substitution. An evaluation context $C[_]$ denotes an extended process with a hole for an extended process.

The semantics of the calculus are given by *Structural equivalence* ($\equiv$), which is defined as the smallest equivalence relation on extended processes that is closed under application of evaluation contexts, α-conversion on names and variables such that:

PAR-0	$A\|0 \equiv A$
PAR-A	$A\|(B\|C) \equiv (A\|B)\|C$
PAR-C	$A\|B \equiv B\|A$
NEW-0	$\nu n.0 \equiv 0$
NEW-C	$\nu u.\nu v.A \equiv \nu v.\nu u.A$
NEW-PAR	$A\|\nu u.B \equiv \nu u.(A\|B)$ if $u \notin fn(A) \cup fn(b)$
REPL	$!P \equiv P\|!P$
REWRITE	$\{^M/_x\} \equiv \{^N/_x\}$ if $M =_E N$
ALIAS	$\nu x.\{^M/_x\} \equiv 0$
SUBST	$\{^M/_x\}\|A \equiv \{^M/_x\}\|A\{^M/_x\}$

and extended by *Internal reduction* ($\rightarrow$), the smallest relation on extended processes closed by structural equivalence and application of evaluation contexts such that:

COMM	$\mathtt{out}(a,x).P \mid \mathtt{in}(a,x).Q$	$\rightarrow$	$P \mid Q$
THEN	$\mathtt{if}\ M = M\ \mathtt{then}\ P\ \mathtt{else}\ Q$	$\rightarrow$	P
ELSE	$\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$	$\rightarrow$	Q
	for any ground terms such that $M \neq_E N$		

To describe the interaction of processes with the exterior, we use labeled operational semantics ($\xrightarrow{\alpha}$) where α can be an input or an output of a channel name or a variable of base type.

IN	$\mathtt{in}(a,x).P \xrightarrow{\mathtt{in}(a,M)} P\{^M/_x\}$
OUT-ATOM	$\mathtt{out}(a,u).P \xrightarrow{\mathtt{out}(a,u)} P$
OPEN-ATOM	$\dfrac{A \xrightarrow{\mathtt{out}(a,u)} A' \qquad u \neq a}{\nu u.A \xrightarrow{\nu u.\mathtt{out}(a,u)} A'}$
SCOPE	$\dfrac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$
PAR	$\dfrac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$
STRUCT	$\dfrac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'}$

Labeled transitions are not closed under the evaluation contexts. Note that a term M cannot be output directly. Instead, we have to assign M to a variable, which can then be output. This is to model that e.g. the output of $\mathtt{enc}(m,k)$ does not give the context access to m. In our definitions we will use the following equivalence and bisimilarity properties:

Definition 1 (Equivalence in a Frame)**.** *Two terms M and N are equal in the frame ϕ, written $(M = N)\phi$, if and only if $\phi \equiv \nu\tilde{n}.\sigma$, $M\sigma = N\sigma$, and $\{\tilde{n}\} \cap (fn(M) \cup fn(N)) = \emptyset$ for some names $\tilde{n}$ and some substitution σ.*

Definition 2 (Static Equivalence ($\approx_s$))**.** *Two closed frames ϕ and ψ are statically equivalent, written $\phi \approx_s \psi$, when dom(ϕ) = dom(ψ) and when for all terms M and N $(M = N)\phi$ if and only if $(M = N)\psi$. Two extended processes A and B are statically equivalent ($A \approx_s B$) if their frames are statically equivalent.*

The intuition behind this definition is that two processes are statically equivalent if the messages exchanged with the environment cannot be distinguished by an attacker (i.e. all operations on both sides give the same results). This idea can be extended to *labeled bisimilarity*.

Definition 3 (Labeled Bisimilarity ($\approx_l$))**.**
Labeled bisimilarity is the largest symmetric relation $\mathcal{R}$ on closed extended processes, such that $A\ \mathcal{R}\ B$ implies

1. $A \approx_s B$,
2. *if* $A \to A'$, *then* $B \to B'$ *and* $A' \mathcal{R} B'$ *for some* B',
3. *if* $A \xrightarrow{\alpha} A'$ *and* $fv(\alpha) \subseteq dom(A)$ *and* $bn(\alpha) \cap fn(B) = \emptyset$, *then* $B \to^* \xrightarrow{\alpha} \to^* B'$ *and* $A' \mathcal{R} B'$ *for some* B'.

In this case each interaction on one side can be simulated by the other side, and the processes are statically equivalent at each step during the execution, thus an attacker cannot distinguish both sides.

To formally describe the interaction between a voter and the attacker, we use the following two definitions. The first one turns a process P into another process P^{ch} that reveals all its inputs and secret data on the channel ch. This can be seen as trying to construct a receipt.

Definition 4 (Process P^{ch} [11])**.** *Let P be a plain process and ch be a channel name. We define P^{ch} as follows:*

- $0^{ch} \triangleq 0$,
- $(P|Q)^{ch} \triangleq P^{ch}|Q^{ch}$,
- $(\nu n.P)^{ch} \triangleq \nu n.\mathtt{out}(ch, n).P^{ch}$ *when n is a name of base type,*
- $(\nu n.P)^{ch} \triangleq \nu n.P^{ch}$ *otherwise,*
- $(\mathtt{in}(u, x).P)^{ch} \triangleq \mathtt{in}(u, x).\mathtt{out}(ch, x).P^{ch}$ *when x is a variable of base type,*
- $(\mathtt{in}(u, x).P)^{ch} \triangleq \mathtt{in}(u, x).P^{ch}$ *otherwise,*
- $(\mathtt{out}(u, M).P)^{ch} \triangleq \mathtt{out}(u, M).P^{ch}$,
- $(!P)^{ch} \triangleq !P^{ch}$,
- $(\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q)^{ch} \triangleq \mathtt{if}\ M = N\ \mathtt{then}\ P^{ch}\ \mathtt{else}\ Q^{ch}$.

In the remainder we assume that $ch \notin fn(P) \cup bn(P)$ before applying the transformation. The second definition does not only reveal the secret data, but also takes orders from an outsider before sending a message or branching, i.e. the process is under complete remote control.

Definition 5 (Process P^{c_1,c_2} [11])**.** *Let P be a plain process and c_1, c_2 be channel names. We define P^{c_1,c_2} as follows:*

- $0^{c_1,c_2} \triangleq 0$,
- $(P|Q)^{c_1,c_2} \triangleq P^{c_1,c_2}|Q^{c_1,c_2}$,
- $(\nu n.P)^{c_1,c_2} \triangleq \nu n.\mathtt{out}(c_1, n).P^{c_1,c_2}$ *when n is a name of base type,*
- $(\nu n.P)^{c_1,c_2} \triangleq \nu n.P^{c_1,c_2}$ *otherwise,*
- $(\mathtt{in}(u, x).P)^{c_1,c_2} \triangleq \mathtt{in}(u, x).\mathtt{out}(c_1, x).P^{c_1,c_2}$ *when x is a variable of base type and x is a fresh variable,*
- $(\mathtt{in}(u, x).P)^{c_1,c_2} \triangleq \mathtt{in}(u, x).P^{c_1,c_2}$ *otherwise,*
- $(\mathtt{out}(u, M).P)^{c_1,c_2} \triangleq \mathtt{in}(c_2, x).\mathtt{out}(u, x).P^{c_1,c_2}$,
- $(!P)^{c_1,c_2} \triangleq !P^{c_1,c_2}$,
- $(\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q)^{c_1,c_2} \triangleq \mathtt{in}(c_2, x).\mathtt{if}\ x = \mathit{true}\ \mathtt{then}\ P^{c_1,c_2}\ \mathtt{else}\ Q^{c_1,c_2}$ *where and x is a fresh variable and true is a constant.*

To hide the output of a process, we use the following definition.

Definition 6 (Process $A^{\setminus out(ch,\cdot)}$ [11])**.** *Let A be an extended process. We define the process $A^{\setminus out(ch,\cdot)}$ as* $\nu ch.(A|!in(ch, x))$.

2.2 Voting Protocol and Process

First of all, we define the notion of a *voting protocol*. Informally, a voting protocol specifies the processes executed by voters and authorities.

Definition 7 (Voting Protocol)**.** *A voting protocol is a tuple* $(V, A_1, \ldots, A_m, \tilde{n})$ *where* V *is the process that is executed by the voter, the* A_j*'s are the processes executed by the election authorities, and* $\tilde{n}$ *is a set of private channels.*

Note that the protocol only defines one process V which will be instantiated for each voter. Yet here may be several authorities, for example a registrar, a bulletin board, a mixer, a tallier, ...

In our definitions we reason about privacy using concrete instances of a voting protocol. An instance is called a *Voting Process*.

Definition 8 (Voting Process)**.** *A voting process of a voting protocol* $(V, A_1, \ldots, A_m, \tilde{n})$ *is a closed plain process*

$$\nu\tilde{m}.(V\sigma_{id_1}\sigma_{f_1}\sigma_{v_1}|\ldots|V\sigma_{id_n}\sigma_{f_n}\sigma_{v_n}|A_1|\ldots|A_l)$$

where $l \leq m$*,* $\tilde{m}$ *includes the secret channel names,* $V\sigma_{id_i}\sigma_{v_i}\sigma_{f_i}$ *are the processes executed by the voters where:*

- σ_{id_i} *is a substitution assigning the identity to a process (this determines for example the secret keys),*
- σ_{v_i} *specifies the vote(s) and if the voter abstains,*
- *and* σ_{f_i} *defines the other behavior, in particular if fake votes are issued,*

and A_j*s are the election authorities which are required to be honest.*

We notice that if an authority is not supposed to be honest, it is not modeled and left to the context, i.e. the attacker (thus $l \leq m$).

Note also that each voter runs the same process V, which is instantiated with a different σ_{id_i} (his identity), σ_{v_i} (his vote(s)) and σ_{f_i} (the fakes). If a protocol does not allow fakes, σ_{f_i} is empty.

This model allows us to reason about more than one correct behavior, which is necessary if for example a voter can decide to abstain from voting or if – in case of multiple votes[1] – he can vote between 0 and n times in the same election. In this case V defines all the possible executions, and σ_{v_i} and σ_{f_i} will determine which of them is executed. Another application are protocols where voters can submit fake ballots and/or several real ballots, even if only one of them is actually counted (like in the one by Juels et al. [19]). In that case σ_{v_i} determines those who are actually counted, and σ_{f_i} the others.

Example 1. *As a running example, we consider the following simple voting protocol.*

Informal description: *To construct his ballot, each voter encrypts his vote with the administrator's public key and signs it using his secret key. The resulting ballot is posted on the bulletin board. After the voting deadline is over, the administrator checks if each ballot is signed by an eligible voter. He then decrypts the correct ballots and publishes the result.*

Formal description in our model: *The protocol is a tuple* $(V, A, \emptyset)$ *where*

A	$=$	$\texttt{in}(ch, (sig, vote))$. $\texttt{if checksign}(sig, pkv) = vote$ $\texttt{then sync } 1$. $\texttt{out}(chR, \texttt{dec}(vote, ska))$	V	$=$	νr. $\texttt{let } evote = \texttt{enc}(v, pka, r) \texttt{ in}$ $\texttt{out}(ch, (\texttt{sign}(evote, skv), evote))$

where – by abuse of notation, this can rewritten in the "pure" calculus – $\texttt{sync } 1$ *is a synchronization point as defined by ProSwapper [20]. A substitution determining the identity of a voter would in this case assign the secret key, e.g.* $\sigma_{id_k} = \{^{sk_k}/_{skv}\}$*. The substitution specifying the vote as for example a vote for candidate* a *would be* $\sigma_{v_k} = \{^{a}/_{v}\}$*. As the protocol does not specify the possibility to create fakes,* σ_{v_k} *is the empty substitution.*

[1] By multiple votes we mean a protocol where each voter can vote several times in the same election, and each vote is transmitted in a separate ballot.

To facilitate notation we denote by S and S' two contexts which are like voting processes but with holes for two and three voters respectively.

Definition 9 (S and S')**.** *We define evaluation contexts S and S' such that*

- *S is like a voting process, but has a hole instead of three processes among $(V\sigma_{id_i}\sigma_{v_i}\sigma_{f_i})_{1\leq i\leq n}$*
- *S' which is like as voting process, but has a hole instead of two processes among $(V\sigma_{id_i}\sigma_{v_i}\sigma_{f_i})_{1\leq i\leq n}$.*

Finally, we formally define what it means for a voting process to abstain. An abstaining voter does not send any message on any channel, in particular no ballot. In the real world, this would correspond to a voter that does not even go to polling station. This is stronger than just voting for a particular "null" candidate $\perp$, which will still result in sending a ballot (a *blank* vote).

Definition 10 (Abstention)**.** *A substitution σ_{v_i} makes a voter abstain if $V\sigma_{id_i}\sigma_{v_i} \approx_l 0$.*

Note that abstention is determined by σ_{v_i} only, so the voter abstains for any σ_{f_i}.

3 Defining Privacy: A Modular Approach

In our setting, the attacker targets one voter (the *targeted voter*) and tries to extract information about the targeted voter's vote(s). If the attacker knows the votes of all other voters, he can infer the targeted voter's vote from the result. Thus we suppose that he is unsure about the vote(s) of one other voter (the *counterbalancing voter*).

We express privacy as an observational equivalence. Intuitively, an attacker should not be able to distinguish between an execution in which the targeted voter behaves and votes as the attacker wishes, and another execution where he only pretends to do so and votes differently. To ensure that the attacker cannot tell the difference by just comparing the result, the counterbalancing voter will compensate the different vote.

Starting from the definitions of Coercion-Resistance, Receipt-Freeness and Vote-Privacy in the literature [11, 2, 19] we propose extensions in the four following dimensions: Communication between the attacker and the targeted voter, Vote-Independence, security against forced-abstention-attacks and knowledge about the behavior of the counterbalancing voter.

1. *Communication between the attacker and the targeted voter:* We define three different levels:
   (a) In the simplest case, the attacker only observes publicly available data and communication. We call this case Vote-Privacy, denoted VP.
   (b) In the second case, the targeted voter tries to convince the attacker that he voted for a certain candidate by revealing his secret data. Yet the attacker should not be able to determine if he actually sent his real data, or a fake receipt. We call this case Receipt-Freeness, denoted RF.
   (c) In the strongest case, the voter pretends to be completely under the control of the attacker, i.e. he reveals his secret data and follows the intruder's instructions. Yet the attacker should be unable to determine if he complied with his instructions or if he only pretended to do so. We call this case Coercion-Resistance, denoted CR.

   It is easy to see that Coercion-Resistance is stronger than Receipt-Freeness, which is stronger than Vote-Privacy ($CR > RF > VP$).

2. *Vote-Independence/Corrupted Voter:* The attacker may control another legitimate voter (neither the targeted nor the counterbalancing voter). In that case he could be able to compromise privacy by trying to relate the corrupted voter's vote to the targeted voter's vote (e.g. by copying it) or using the corrupted voter's secret data, such as his credentials or keys. In our definitions, we distinguish two case for Eve (the attacker):
   (a) Eve is an Outsider (denoted O): The attacker is an external observer.

(b) Eve is an Insider (denoted I): The attacker has a legitimate voter under his control.

Intuitively, Insider is the stronger setting ($I > O$).

3. *Security against forced-abstention-attacks:* A protocol can ensure that a voter can still vote as intended, although a coercer wants him to abstain. Note that in contrast to the literature [19, 2], we define this property independently from of Coercion-Resistance, as we also want to apply it in the case of Vote-Privacy. Our model expresses this by requiring the observational equivalence to hold:

   (a) in any case, i.e. even if the voter is forced to abstain. We call this case security against Forced-Abstention-Attacks, denoted FA.

   (b) if the targeted voter does not abstain from voting (i.e. always participates). We call this case Participation Only, denoted PO.

   In this dimension security against Forced-Abstention-Attacks is a stronger property than Participation Only ($FA > PO$).

4. *Knowledge about the behavior of the counterbalancing voter*: To model different knowledge about the behavior of the counterbalancing voter, we consider two cases:

   (a) The observational equivalence holds for any behavior of this voter, i.e. any σ_{f_i}. This models an attacker that knows if the counterbalancing voter is going to post fake ballots, or a situation where there is no "noise" (=fake ballots) on the bulletin board. We call this case Any Behavior, denoted AB.

   (b) The observational equivalence holds for at least one behavior of this voter, which may additionally change, i.e. one σ_{f_i} and one $\sigma_{f'_i}$. In this case, the attacker is unsure about the number of fake ballots, i.e. there is enough noise. We call this case Exists Behavior, denoted EB.

   Any Behavior is stronger than EB ($AB > EB$).

The strongest possible property is thus $CR^{I,FA,AB}$, the weakest $VP^{O,PO,EB}$. If we leave out the parameter, we take the weakest setting as a default, i.e. VP denotes $VP^{O,PO,EB}$.

3.1 Definitions in the applied pi calculus

Our definition is parametrized using the following parameters (as explained above):

- $Privacy = \{CR, RF, VP\}$ ("Coercion-Resistance", "Receipt-Freeness" or "Vote-Privacy").
- $Eve = \{I, O\}$ ("Insider" or "Outsider").
- $Abs = \{FA, PO\}$ ("Security against Forced-Abstention-Attacks"or "Participation Only").
- $Behavior = \{AB, EB\}$ ("Any Behavior"or "Exists Behavior").

Definition 11 ($Privacy^{Eve,Abs,Behavior}$)**.** *A protocol fulfills $Privacy^{Eve,Abs,Behavior}$ if for any voting process $\mathcal{S}$ there exists a process V' and for any substitution σ_{f_A} and σ_{f_C}, and any context A such that $A = \nu\tilde{ch}.(_|A'^{chout})$) where $\tilde{ch}$ are all unbound channels and names in A' and in the "hole",*

- *if $Behavior$ is EB: there exist substitutions σ_{f_B}, $\sigma_{f'_B}$ and $\sigma_{f'_A}$,*
- *if $Behavior$ is AB: for any substitutions $\sigma_{f_B} = \sigma_{f'_B}$ there exists a substitution $\sigma_{f'_A}$,*

such that for all votes σ_{v_A} and σ_{v_B} where $V\sigma_{v_B}$ does not make a voter abstain[2], one of the following holds depending on the privacy setting:

[2]This condition is needed to ensure that in the case PO no voter can abstain.

- *if Privacy is Vote-Privacy (VP):*

$$A\left[\mathcal{S}\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right] \approx_l A\left[\mathcal{S}\left[V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v_A}|\mathcal{V}_C\right]\right]$$

- *if Privacy is Receipt-Freeness (RF):*
  - $V'^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}$
  - $A\left[\mathcal{S}\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right] \approx_l A\left[\mathcal{S}\left[V'|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v_A}|\mathcal{V}_C\right]\right]$
- *if Privacy is Coercion-Resistance (CR):*
  For any context $C = \nu c_1.\nu c_2.(_|P')$ with $\tilde{n} \cap fn(C) = \emptyset$ and
  $\mathcal{S}\left[C\left[V\sigma_{id_A}^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l \mathcal{S}\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]$
  we have
  - $C\left[V'\right]^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}$
  - $A\left[\mathcal{S}\left[C\left[V\sigma_{id_A}^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right] \approx_l A\left[\mathcal{S}\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v_A}|\mathcal{V}_C\right]\right]$

where

- *If Eve is:*
  - *Insider(I): $\mathcal{S} := S$ and $\mathcal{V}_C := V\sigma_{id_C}^{c_1,c_2}$*
  - *Outsider (O): $\mathcal{S} := S'$ and $\mathcal{V}_C := 0$*
- *If Abs is:*
  - *Participation Only (PO): $V\sigma_{id_A}$ does not abstain, i.e. $V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A} \not\approx_l 0$.*
  - *Security against Forced-Abstention-Attacks (FA), he may abstain.*

The context A represents the attacker. We chose to make the attacker's behavior explicit as some protocols (such as the one by Juels et al. [19]) require σ_{f_B} and $\sigma_{f'_B}$ to be chosen as a function of the attacker (see our technical report [14] for details, note however that V' is chosen only as a function of the protocol). To ensure that A does not only forward the channels (which would leave the attacker to the outside again and thus contradict our intention of choosing the processes as a function of A), we require A to bind all free names and channels inside. He will only forward the knowledge he is able to obtain during the execution of the protocol.

The following examples illustrate how the parameters change the definition and give intuitions.

Example 2 ($VP^{O,PO,AB}$)**.** *A protocol fulfills $VP^{O,PO,AB}$ if for any voting process S' and any substitutions σ_{f_A}, σ_{f_B} and σ_{f_C}, and for any context A such that $A = \nu\tilde{ch}.(_|A'^{chout})$ where $\tilde{ch}$ are all unbound channels and names in A' and in the "hole", there exist a substitution $\sigma_{f'_A}$ such that for all votes σ_{v_A} and σ_{v_B} where σ_{v_B} and σ_{v_A} does not make a voter abstain we have:*

$$A\left[S'\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|0\right]\right] \approx_l A\left[S'\left[V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_A}|0\right]\right]$$

Note that it is sufficient to prove

$$S'\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}\right] \approx_l S'\left[V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_A}\right]$$

as labeled bisimilarity is closed under the application of contexts. If there is only one correct behavior of V (i.e. no fakes), we can rewrite this as

$$S'\left[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}\right] \approx_l S'\left[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}\right] \quad (1)$$

This coincides with the definition of Vote-Privacy given by Delaune et al. [11]: Two situations where two voters swap votes are bisimilar.

We also note that Receipt-Freeness in the DKR-model corresponds to $RF^{O,PO,AB}$ in our model, and Coercion-Resistance in the DKR-model corresponds to $CR^{O,PO,AB}$ in our model.

Example 3 (Application)**.** *Consider our running example of a simple voting protocol. We show that it ensures $VP^{O,PO,AB}$ as defined above. We suppose the secret keys to be secret and the administrator to be honest. In that case, Proverif is able to prove (1), which gives Lemma 1.*

Lemma 1. *The simple voting protocol ensures $VP^{O,PO,AB}$.*

The code used is available on our website [13].

It is easy to see that this protocol does not guarantee Vote-Privacy for an inside attacker ($VP^{I,PO,AB}$), as he can simply access the votes on the bulletin board and copy them. He can identify which vote was posted by which voter using the signatures.

The protocol is not receipt-free ($RF^{Eve,Abs,Behavior}$) either as the randomness used for encrypting the vote can be used as a receipt. Since the bulletin board reveals which voters participated, it is not resistant against forced-abstention attacks.

Example 4 ($VP^{O,FA,AB}$)**.** *A protocol fulfills $VP^{O,FA,AB}$ if for any voting process S' and any substitutions σ_{f_A}, σ_{f_B} and σ_{f_C}, and for any context A such that $A = \nu\tilde{ch}.(_|A'^{chout}))$ where $\tilde{ch}$ are all unbound channels and names in A' and in the "hole", there exists a substitution $\sigma_{f'_A}$ such that for all votes σ_{v_A} and σ_{v_B} where σ_{v_B} does not make a voter abstain we have:*

$$A\left[S'\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|0\right]\right] \approx_l A\left[S'\left[V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_A}|0\right]\right]$$

In this case, σ_{v_A} can make a voter abstain. As σ_{v_B} may not specify abstention, we have an observational equivalence between a situation where the targeted voter abstains, and a situation where he votes and the counterbalancing voter abstains. This captures the security against forced-abstention-attacks.

Example 5 ($RF^{I,PO,AB}$)**.** *A protocol fulfills $RF^{I,PO,AB}$ if for any voting process S there exists a process V', and for any substitutions σ_{f_A}, σ_{f_B} and σ_{f_C}, and for any context A such that $A = \nu\tilde{ch}.(_|A'^{chout}))$ where $\tilde{ch}$ are all unbound channels and names in A' and in the "hole", there exists a substitution $\sigma_{f'_A}$ such that for all votes σ_{v_A} and σ_{v_B} where σ_{v_B} and σ_{v_A} do not make a voter abstain we have*

$$V'^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}$$

and

$$A\left[S\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|V\sigma_{id_C}^{c_1,c_2}\right]\right] \approx_l A\left[S\left[V'|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_A}|V\sigma_{id_C}^{c_1,c_2}\right]\right].$$

Note that again it is sufficient to prove

$$S\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|V\sigma_{id_C}^{c_1,c_2}\right] \approx_l S\left[V'|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_A}|V\sigma_{id_C}^{c_1,c_2}\right]$$

as labeled bisimilarity is closed under the application of contexts. If there is only one correct behavior of $V\sigma_A$, this coincides with the definition of Vote-Independence with Passive Collaboration in the DKR-model [15]: If a protocol is receipt-free, there exists a counter-strategy (V') that allows the targeted voter to fake the receipt and vote differently. Analogously, Vote-Independence in the DKR-model corresponds to $VP^{I,PO,AB}$, and Vote-Independence with passive Collaboration corresponds to $CR^{I,PO,AB}$ in our model.

Example 6 ($CR^{O,FA,AB}$)**.** *A protocol fulfills $CR^{O,FA,AB}$ if for any voting process $\mathcal{S}$ there exists a process V' and for any substitution σ_{f_A} and σ_{f_C}, and any context A such that $A = \nu\tilde{ch}.(_|A'^{chout}))$ where $\tilde{ch}$ are all unbound channels and names in A' and in the "hole", for any substitutions $\sigma_{f_B} = \sigma_{f'_B}$ there exists a substitution $\sigma_{f'_A}$, such that for all votes σ_{v_A} and σ_{v_B} where $V\sigma_{v_B}$ does not make a voter abstain, the following holds: For any context $C = \nu c_1.\nu c_2.(_|P')$ with $\tilde{n} \cap fn(C) = \emptyset$ and $S'\left[C\left[V\sigma_{id_A}^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|0\right] \approx_l S'\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|0\right]$ we have*

- $C\left[V'\right]^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}$
- $A\left[S'\left[C\left[V\sigma_{id_A}^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|0\right]\right] \approx_l A\left[S'\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v_A}|0\right]\right]$

The intuition behind this definition is the following: the context C also belongs to the attacker and tries to force the targeted voter to vote for a certain candidate or to make him abstain (depending on σ_{id_A}), whereas V' tries to vote differently and to escape coercion.

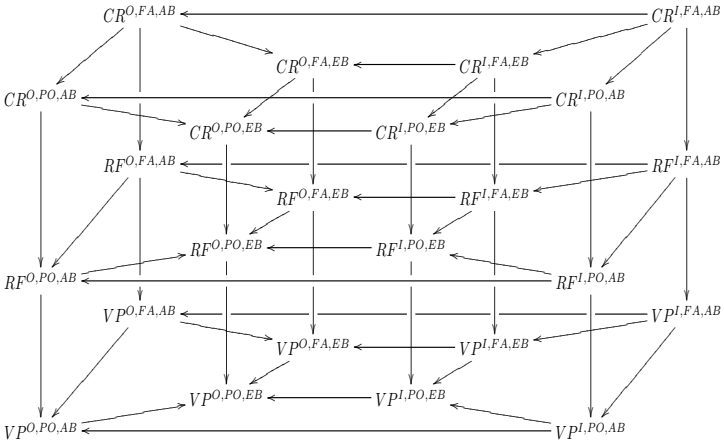


Figure 1: Hierarchy of privacy notions. $A \rightarrow B$ means that any protocol that respects property A also respects property B.

4 Hierarchy

As already described in Section 3, we have a hierarchy of notions in each dimension.

Proposition 1. *For* $Privacy \in \{VP, RF, CR\}$, $Abs \in \{FA, PO\}$ *and* $Behavior \in \{AB, EB\}$ *we have:*

1. *Any attack that works for an outsider can also be used for an insider: If a protocol respects* $Privacy^{I,Abs,Behavior}$*, then it also respects* $Privacy^{O,Abs,Behavior}$.
2. *If a protocol is secure against Forced-Abstention attacks, it is also secure in the "PO" case: If a protocol respects* $Privacy^{Eve,FA,Behavior}$*, it also respects* $Privacy^{Eve,PO,Behavior}$.
3. *If the property holds for any behavior, there exists a behavior for which it holds: If a protocol respects* $Privacy^{Eve,Abs,AB}$*, it also respects* $Privacy^{Eve,Abs,EB}$.
4. *Coercion-Resistance is stronger than Receipt-Freeness, which is stronger than Vote-Privacy:*
   - *If a protocol respects* $CR^{Eve,Abs,Behavior}$*, it also respects* $RF^{Eve,Abs,Behavior}$.
   - *If a protocol respects* $RF^{Eve,Abs,Behavior}$*, it also respects* $VP^{Eve,Abs,Behavior}$.

This was already shown before in the DKR-model [11], the extension to our model is straightforward. All the formal proofs are given in in Appendix A.

Taking these properties together, we arrive at the hierarchy shown in Figure 1. For protocols that do not use fakes, the forth dimensions "$Behavior$" collapses and we obtain a simple tower (see Figure 2).

5 Case Studies

We applied our family of notions on several case studies, chosen to show that each of our dimensions corresponds to a different property of real-world protocols. The results are summed up in and Table 1, the position of the case studies within our hierarchy is shown in Figure 2.

5.1 FOO

The protocol by Fujioka et al. [17] is based on blind signatures and commitments. It was shown to ensure Vote-Privacy [11] and Vote-Independence [15] in the DKR-model, but is not receipt free as the randomness of the commitment can be used as a receipt. We will show that it ensures $VP^{I,PO,AB}$ (i.e. Vote-Privacy against an insider for any behavior of the counterbalancing voter, but just in the Participation Only case) even if multiple votes are permitted.

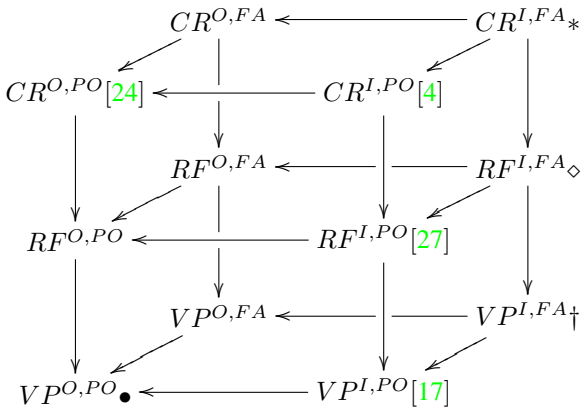


Figure 2: Collapsed hierarchy of privacy notions with examples: The simple voting protocol, our running example (•); Bingo Voting [4]; Bingo Voting with the assumption that the attacker does not know if a voter entered the voting booth (∗); Okamoto [27]; Okamoto with a private channel to the administrator (⋄); FOO [17]; FOO with a private channel to the administrator (†); and Lee et al. [24].

5.1.1 Protocol Description

The protocol is split into three phases. In the first phase the administrator signs the voter's commitment to his vote:

- Voter V_i chooses his vote v_i and computes a commitment $x_i = \xi(v_i, k_i)$ for a random key k_i.
- He blinds the commitment using a blinding function χ, a random value r_i and obtains $e_i = \chi(x_i, r_i)$.
- He signs e_i and sends the signature $s_i = \sigma_{V_i}(e_i)$ together with e_i and his identity to the administrator.
- The administrator checks if V_i has the right to vote and has not yet voted, and if the signature s_i is correct. If all tests succeed, he signs $d_i = \sigma_A(e_i)$ and sends it back to V_i.
- V_i unblinds the signature and obtains $y_i = \delta(d_i, r_i)$. He checks the signature.

In the second phase the voter submits his ballot:

- Voter V_i sends (x_i, y_i) to the collector C through an anonymous channel.
- The collector checks the administrators signature and enters (x_i, y_i) as the l-th entry into a list.

When all ballots are cast or when the deadline is over, the counting phase begins:

- The collector publishes the list of correct ballots.
- V_i verifies that his commitment appears on the list and sends (l, r_i) to C using an anonymous channel.
- The collector C opens the l-th ballot using r_i and publishes the vote.

5.1.2 Analysis

In the standard case of single votes, the protocol was shown to ensure Vote-Independence in the DKR-model [15]. As explained above, this suffices to prove $VP^{I,PO,AB}$ and we have the following result.

Lemma 2. *FOO with single votes respects* $VP^{I,PO,AB}$.

In addition to the original manual proof, we were able to verify this result automatically using ProSwapper and ProVerif. The code is available on our website [13].

We also considered a slightly modified version of the protocol which allows multiple votes. In this version each voter votes up to n times by repeating the the first and second phase for each vote. After the deadline has passed, he reveals all the random values at the same time. In this case, we still have the same result.

Lemma 3. *FOO with multiple votes respects* $VP^{I,PO,AB}$.

This can be proved using ProSwapper, ProVerif and a script that generates all possible behaviors [13].

However it is not secure against Forced-Abstention-Attacks, as the voters send their identity to the administrator over a public channel. This could be fixed by using a private channel instead.

5.2 Okamoto

The protocol by Okamoto [27] is similar to the protocol by Fujioka et al. discussed above, but it uses trap-door commitments to achieve receipt-freeness. It is however not Coercion-Resistant as the coercer can force the voter to use a specially prepared commitment [11].

5.2.1 Protocol Description

The main differences to FOO are the use of trap-door commitments and the existence of timeliness member to open the commitments. The first phase - during which the voter obtains a signature on his commitment - follows exactly the same protocol as FOO, except that this time ξ is a trapdoor-commitment. In the second phase the vote is submitted:

- Voter V_i sends the signed trap-door commitment to the collector C through an anonymous channel.
- The collector checks the administrators signature and enters (x_i, y_i) into a list.
- The voter sends (v_i, r_i, x_i) to the timeliness member through an untappable anonymous channel

When all ballots are cast and/or when the deadline is over, the counting phase begins:

- The collector publishes the list of correct ballots.
- V_i verifies that his commitment appears on the list.
- The timeliness member publishes a randomly shuffled list of votes v_i and a zero-knowledge proof that he knows a permutation π for which $x_{\pi(i)} = \xi(v_i, r_i)$.

5.2.2 Analysis

Similar to FOO, as a consequence of the analysis in [16], we have the following Lemma:

Lemma 4. *The protocol by Okamoto respects* $RF^{I,PO,AB}$.

Unfortunately ProVerif cannot prove this automatically as the equational theory is too complex. This is due to the trapdoor-commitment which yields a non-confluent equational theory.

5.3 Juels et al.

The protocol by Juels et al. [19] was proved to be Coercion-Resistant in a special model [2] that is not compatible with the DKR-model. We will show that it can be proved $CR^{I,FA,EB}$ in our model, i.e. Coercion-Resistant with security against forced-abstention attacks against an insider if the behavior of the counterbalancing voter is unknown.

5.3.1 Protocol Description

Each voter V_i registers with a registrar R and receives his voting credential $cred_i$ over an untappable channel. After all voters have been registered, the registrar publishes a list of chiphertexts – where each cyphertext corresponds to a valid credential encrypted with the authorities public key – on the bulletin board.

To vote, a voter encrypts his vote and his credential using the authorities' public key and adds a proof of correct construction. The resulting ballot is sent over an anonymous channel to the bulletin board.

After the voting is over, the authorities eliminate all ballots containing incorrect proofs and perform plaintext equivalence tests to remove ballots that use the same credential. If duplicates are found, they are eliminated using some pre-defined policy (e.g. only the latest ballot is kept). The remaining ballots are mixed using a mix-net. After the mixing has been done, the authorities check the ballots for valid credentials using plaintext equivalence tests. If a valid credential was used, the vote is decrypted and published, otherwise discarded.

5.3.2 Analysis

Proposition 2. *The protocol by Juels, Catalano and Jakobsson ensures* $CR^{I,FA,EB}$.

Proof. Sketch: The proof is based on the fact that a coerced voter can always cheat a coercer by providing a fake credential and vote as instructed using the fake credential. Later on, he can use the real credential to submit his real vote. This however means that we have one situation where the targeted voter posts two ballots, and another one where he posts only one. To account for this problem, $V\sigma_{id_B}\sigma_{v_B}\sigma_{f_B}$ has to submit a fake ballot, whereas $V\sigma_{id_B}\sigma_{v_A}\sigma_{f'_B}$ does not. This is why the protocol is not secure for any behavior of the counterbalancing voter. Intuitively, this can be seen as the need for noise on the bulletin board. If no honest voter posts fake ballots, coercion remains possible.

Another option for the attacker would be to try to use the credential he obtained from the voter to vote for a completely different candidate d. If the credential was correct, this vote might be counted (if it is not eliminated before the mixing). If it was a fake, it will definitely not be counted. Suppose that the credential was real, then we would have a vote for a and a vote for d with the same correct credential, and a vote for b and a fake by $V\sigma_{id_B}$. Such a situation however looks the same as a case where the $V\sigma_{id_B}\sigma_{v_B}\sigma_{f'_B}$ submitted two votes for c and d using the same credential[3], and V' voted for a and posted a fake to cheat the coercer.

Copying votes is not possible in this setting, as the attacker cannot know which ballots have been posted by which voter due to the anonymous channel.

Similarly, forced-abstention attacks are not possible as posting the ballots is anonymous, and the attacker cannot relate a ballot to a voter. □

The detailed model and proof can be found in Appendix B.

Interestingly we do not need noise on the bulletin board in the case of Vote-Privacy, i.e. we have Vote-Privacy for any behavior of the counterbalancing voter (AB).

Proposition 3. *The protocol by Juels et al. respects* $VP^{I,FA,AB}$.

In this case, no voter needs to fake a credential or a receipt, and the real credentials are sufficient.

5.4 Bingo Voting

Bingo Voting was developed by Bohli et al. [4] to achieve coercion-resistance as well as individual and universal verifiability by using a trusted random number generator (RNG). In our hierarchy Bingo Voting ensures $CR^{I,PO,AB}$ if the voting machine is to be trusted.

[3] Note that either the vote for a or for d will not be counted and is thus part of σ_{f_B}, not of σ_{v_B}.

5.4.1 Protocol Description

The protocol is split into three phases: The pre-voting phase, the voting phase and the post-voting phase. In the pre-voting phase the voting machine generates for every candidate p_j k (k is the number of voters) random values $n_{i,j}$. It commits to the $k \cdot l$ (l is the number of candidates) pairs $(n_{i,j}, p_j)$ and publishes the shuffled commitments.

In the voting phase, the voter enters the voting booth and selects the candidate he wants to vote for on the voting machine. The RNG generates a random number r which is transmitted to the voting machine and displayed to the voter. The voting machine chooses for each candidate, except for the voter's choice, a dummy vote. For the chosen candidate, the random value from the RNG is used and the receipt is created. Finally the voter checks that the number displayed on the RNG corresponds to the entry of his candidate on the receipt.

In the post-voting phase the voting machine announces the result, publishes all receipts and opens the commitments of all unused dummy votes. The machine also generates non-interactive zero-knowledge proofs that each unopened commitment was actually used as a dummy vote in one of the receipts.

We assume the voting machine to be honest, otherwise no privacy can be guaranteed as the vote is submitted in clear by the voter. The detailed model we used for the analysis can be found in [16].

5.4.2 Analysis

Lemma 5. *Bingo Voting respects $CR^{I,PO,AB}$, i.e. Coercion-Resistance against an inside attacker in the Participation Only case.*

This is - once again - a result of a proof in the DKR-model [16].

If we assume that the attacker cannot not know if a voter entered the voting both, we have would also have security against-forced abstention attacks, i.e. $CR^{I,FA,AB}$.

5.5 Lee et al.

The protocol by Lee et al. [24] was shown to be Coercion-Resistant in the DKR-model [11]. Yet the protocol is neither secure against an inside attacker nor against forced-abstention attacks, as we will show. It is based on trusted devices that re-encrypt ballots and prove their correct behavior to the voter using designated verifier proofs (DVPs).

5.5.1 Protocol Description

We simplified the protocol to focus on the important parts with respect to privacy and vote-independence. For example, we do not consider distributed authorities but model them as one honest authority.

- The administrator sets up the election, distributes keys and registers legitimate voters. Each voter is equipped with his personal trusted device. At the end, he publishes a list of legitimate voters and corresponding trusted devices.

- The voter encrypts his vote with the tallier's public key (using the El Gamal scheme), signs it and sends it to his trusted device over a private channel. The trusted device verifies the signature, re-encrypts and signs the vote, and returns it, together with a DVP that the re-encryption is correct, to the voter. The voter verifies the signature and the proof, double signs the ballot and publishes it on the bulletin board.

- The administrator verifies for all ballots if the voter has the right to vote and if the vote is correctly signed. He publishes the list of correct ballots, which is then shuffled by the mixer.

- The tallier decrypts the mixed votes and publishes the result.

Protocol	Privacy Notion	Comments
Juels et al. [19]	$CR^{I,FA,EB}$	Requires fakes to achieve coercion-resistance
Bingo Voting [4]	$CR^{I,PO,AB}$	Trusted voting machine, vulnerable to forced abstention
- variant	$CR^{I,FA,AB}$	Secure against forced abstention if the attacker is unaware of the voter entering the voting booth
Lee et al. [24]	$CR^{O,PO,AB}$	Trusted randomizer, vulnerable to vote-copying
Okamoto [27]	$RF^{I,PO,AB}$	Based on trap-door commitments
- variant	$RF^{I,FA,AB}$	Private channel to the administrator, secure against forced abstention attacks
Fujioka et al. [17]	$VP^{I,PO,AB}$	Based on blind signatures
- variant	$VP^{I,PO,AB}$	Permits multiple votes
Simple Voting Protocol	$VP^{O,PO,AB}$	The running example, vulnerable to vote-copying

Table 1: Results of the case studies

5.5.2 Analysis

Proposition 4. *The protocol by Lee et al. respects $CR^{O,PO,AB}$ (Coercion-Resistance against an outside attacker with Any Behavior of the counterbalancing voter).*

This is a result of the original proof in the DKR-model.

Yet the protocol is not secure against an inside attacker. As acknowledged by the authors in their original paper [24], it is possible to copy votes. More precisely, an attacker can access the ballots on the bulletin board before the mixing takes place. He can easily verify which ballot belongs to which voter as they are signed by the voters themselves. He can remove the signature and use the ciphertext as an input to his trusted device. The trusted device will re-encrypt and sign it. This allows the attacker to construct a correct ballot which contains the same vote as the targeted honest voter. By submitting this ballot he obtains a different election outcome in both cases of the observational equivalence. This attack was also found in the DKR-model [16].

Additionally, this protocol is not secure against forced-abstention attacks as the ballots on the bulletin board are signed by the voters. The attacker can thus easily verify if a voter voted, or not.

6 Conclusion

We proposed a modular family of formal privacy notions in the applied pi calculus which allows to assess the level of privacy provided by a voting protocol. We illustrated the family of notions using examples and applied it in several case studies, including a case with multiple votes per voter (a variant of FOO [17]) and forced abstention attacks (see Table 1). In particular we were able to show that the different dimensions of our definitions correspond to different properties of real-world protocols, and that in many cases the verification can be done automatically using existing tools.

Limitations and Future Work. In this paper we employ a possibilistic approach: We call a protocol secure if there is a way for the targeted voter to escape coercion. As we are in a symbolic model, we do not consider probabilities. Hence the adversary may in reality still have a certain probability of detecting that the coerced voter tried to resist coercion. This is beyond the scope of this paper, yet a computational translation of our definitions should be able take this into account.

Additionally, it would also be desirable to verify more properties automatically as manual proofs are often difficult and tend to be error-prone.

References

[1] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *POPL'01*, pages 104–115, 2001. 1, 1, 2.1

[2] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. *CSF*, 0:195–209, 2008. 1, 3, 3, 5.3

[3] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008. 3

[4] Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. Bingo voting: Secure and coercion-free voting using a trusted random number generator. In *E-Voting and Identity*, volume 4896 of *LNCS*, pages 111–124. Springer, 2007. 3, 1, 4, 2, 5.4, 5.5.2

[5] Jens-Matthias Bohli and Andreas Pashalidis. *Relations Among Privacy Notions*, pages 362–380. Springer, 2009. 1

[6] Ran Canetti and Rosario Gennaro. Incoercible multiparty computation (extended abstract). In *FOCS*, pages 504–513, 1996. 1

[7] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On some incompatible properties of voting schemes. In *Proceedings of the IAVoSS Workshop on Trustworthy Elections, 2006.* 1

[8] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. *IEEE Symposium on Security and Privacy*, 0:354–368, 2008. 1

[9] UK Electoral Commission. Key issues and conclusions: May 2007 electoral pilot schemes. 1

[10] Bundesverfassungsgericht (Germany's Federal Constitutional Court). Use of voting computers in 2005 bundestag election unconstitutional, March 2009. 1

[11] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17:435–487, 2009. 1, 4, 5, 6, 3, 3.1, 4, 5.1, 5.2, 5.5, A, 6, 7, 4

[12] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols: A taster. In *Towards Trustworthy Elections – New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 289–309. Springer, 2010. 1

[13] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Code and scripts used to automatically verify the examples. Available at `http://www-verimag.imag.fr/~dreier/papers/sfcs-code.zip`, 2011. 3, 5.1.2, 5.1.2

[14] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. A formal taxonomy of privacy in voting protocols. Technical Report TR-2011-10, Verimag Research Report, May 2011. Available at `http://www-verimag.imag.fr/TR/TR-2011-10.pdf`. 3.1

[15] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Vote-independence: A powerful privacy notion for voting protocols. In *Proceedings of the 4th Workshop on Foundations & Practice of Security (FPS)*, volume 6888 of *LNCS*, page 164ff. Springer, 2011. 1, 1, 3.1, 5.1, 5.1.2

[16] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. Vote-independence: A powerful privacy notion for voting protocols. Technical Report TR-2011-8, Verimag Research Report, April 2011. Available at `http://www-verimag.imag.fr/TR/TR-2011-8.pdf`. 3, 1, 5.2.2, 5.4.1, 5.4.2, 5.5.2, 4

[17] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology – AUSCRYPT '92*, volume 718 of *LNCS*, pages 244–251. Springer, 1992. 3, 5.1, 4, 2, 5.5.2, 6

[18] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections, 2002. 1

[19] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *WPES'05*, pages 61–70. ACM, 2005. 3, 1, 2.2, 3, 3, 3.1, 5.3, 5.5.2

[20] Petr Klus, Ben Smyth, and Mark D. Ryan. Proswapper: Improved equivalence verifier for proverif. http://www.bensmyth.com/proswapper.php, 2010. 3, 1

[21] Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *ESORICS 2010*, volume 6345 of *LNCS*. Springer, 2010. 1

[22] R. Küsters and T. Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *S&P*, pages 251–266. IEEE, 2009. 1

[23] Lucie Langer, Hugo Jonker, and Wolter Pieters. Anonymity and verifiability in voting: understanding (un)linkability. In *ICICS*. Springer, 2010. 1

[24] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *ICISC*, volume 2971 of *LNCS*. Springer Berlin / Heidelberg, 2004. 1, 4, 2, 5.5, 5.5.2

[25] Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *CRYPTO 2006*, volume 4117 of *LNCS*, pages 373–392. Springer, 2006. 1

[26] Participants of the Dagstuhl Conference on Frontiers of E-Voting. Dagstuhl accord, 2007. 1

[27] Tatsuaki Okamoto. An electronic voting scheme. In *Proceedings of the IFIP World Conference on IT Tools*, 1996. 3, 4, 2, 5.2, 5.5.2

[28] Ben Smyth and Veronique Cortier. Attacking and fixing helios: An analysis of ballot secrecy. Cryptology ePrint Archive, Report 2010/625, 2010. http://eprint.iacr.org/. 1

[29] Ben Smyth and Veronique Cortier. Attacking and fixing helios: An analysis of ballot secrecy. In *CSF'11*. IEEE, 2011. 1, 1

[30] Ben Smyth, Mark D. Ryan, Steve Kremer, and Mounira Kourjieh. Towards automatic analysis of election verifiability properties. In *ARSPA-WITS*, volume 6186 of *LNCS*, pages 146–163. Springer, 2010. 1

[31] Dominique Unruh and Jörn Müller-Quade. Universally composable incoercibility. In *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *LNCS*, pages 411–428. Springer, 2010. 1

[32] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties (Netherland's Ministry of the Interior and Kingdom Relations). Stemmen met potlood en papier (voting with pencil andpaper), May 2008. 1

A Proof of Proposition 1

For this proof, we need the following Lemmas by [11].

Lemma 6 ([11]). *Let P be a closed plain process and ch a channel name such that $ch \notin fn(P) \cup bn(P)$. We have $(P^{ch})^{\setminus out(ch,\cdot)} \approx_l P$.*

Lemma 7 ([11]). *Let $C_1 = \nu\tilde{u}_1.(_|B_1)$ and $C_2 = \nu\tilde{u}_2.(_|B_2)$ be two evaluation contexts such that $\tilde{u}_1 \cap (fv(B_2) \cup fn(B_2)) = \emptyset$ and $\tilde{u}_2 \cap (fv(B_1) \cup fn(B_1)) = \emptyset$. Then we have $C_1[C_2[A]] \equiv C_2[C_1[A]]$ for any extended process A.*

We will now prove the different propositions.

1. If a protocol respects $Privacy^{I,Abs,Behavior}$ then it also respects $Privacy^{O,Abs,Behavior}$.

   Proof. We detail the case $VP^{I,Abs,Behavior}$, the other cases are similar. We will use a proof by contradiction. Suppose that a protocol does not ensure $VP^{I,Abs,Behavior}$, i.e. we have

   $$A\left[S'\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}\right]\right] \not\approx_l A\left[S'\left[V'|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}\right]\right]$$

   which we can rewrite as

   $$A\left[S\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|V\sigma_{id_i}\sigma_{f_i}\sigma_{v_i}\right]\right] \not\approx_l A\left[S\left[V'|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}|V\sigma_{id_i}\sigma_{f_i}\sigma_{v_i}\right]\right]$$

   for some honest voter $V\sigma_{id_i}\sigma_{f_i}\sigma_{v_i}$. This yields the contradiction for a context (an attacker) enforcing $V\sigma^{c_1,c_2}_{id_C} \approx_l V\sigma_{id_i}\sigma_{f_i}\sigma_{v_i}$. $\lightning$ □

2. If a protocol respects $Privacy^{Eve,FA,Behavior}$, it also respects $Privacy^{Eve,PO,Behavior}$.

   Proof. This is trivial, in the case FA we consider all σ_{v_A}, whereas we exclude some in the case PO. Thus, if the bisimilarity holds for FA, it also holds for PO. □

3. If a protocol respects $Privacy^{Eve,Abs,AB}$, it also respects $Privacy^{Eve,Abs,EB}$

   Proof. In the case AB we consider all σ_{f_B} and have $\sigma_{f'_B} = \sigma_{f_B}$ and $\sigma_{v'_A} = \sigma_{v_A}$. This implies that there exist σ_{f_B}, $\sigma_{f'_B}$ and $\sigma_{v'_A}$ (the case EB) such that the bisimilarity holds. □

4. Coercion-Resistance is stronger than Receipt-Freeness, which is stronger than Vote-Privacy:

   - If a protocol respects $CR^{Eve,Abs,Behavior}$, it also respects $RF^{Eve,Abs,Behavior}$.

     Proof. The proof is similar to the proofs of Coercion-Resistance implies Receipt-Freeness and Vote-Independence with active collaboration implies Vote-Independence with passive Collaboration in the DKR model [11, 16].
     Let C be an evaluation context such that $C = \nu c_1.\nu c_2.(_|P)$ for some plain process P which fulfills

     $$\mathcal{S}\left[C\left[V\sigma^{c_1,c_2}_{id_A}\right]|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right] \approx_l \mathcal{S}\left[V\sigma_{id_A}\sigma_{f_A}\sigma^{chc}_{v_A}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]$$

     Note that such a C can be constructed directly from the vote process V. By hypothesis we know that there is a closed plain process V' such that

     $$C\left[V'\right]^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}$$

     and

     $$A\left[\mathcal{S}\left[C\left[V\sigma^{c_1,c_2}_{id_A}\right]|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right] \approx_l A\left[\mathcal{S}\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}|\mathcal{V}_C\right]\right]$$

     We have to find another process V'' such that

     $$V''^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}$$

     and

     $$A\left[\mathcal{S}\left[V\sigma_{id_A}\sigma_{f_A}\sigma^{chc}_{v_A}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right] \approx_l A\left[\mathcal{S}\left[V'|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}|\mathcal{V}_C\right]\right]$$

     Let $V'' = C[V']$. This directly fulfills the first requirement. We now use the condition on C and apply the context A

     $$A\left[\mathcal{S}\left[C\left[V\sigma^{c_1,c_2}_{id_A}\right]|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right] \approx_l A\left[\mathcal{S}\left[V\sigma_{id_A}\sigma_{f_A}\sigma^{chc}_{v_A}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right]$$

The second hypothesis gives

$$A\left[\mathcal{S}\left[C\left[V\sigma_{id_A}^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right] \approx_l A\left[\mathcal{S}\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}|\mathcal{V}_C\right]\right]$$

As labeled bisimilarity is transitive, we can conclude

$$A\left[\mathcal{S}\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right] \approx_l A\left[\mathcal{S}\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}|\mathcal{V}_C\right]\right]$$

which gives us the desired result for $V'' = C[V']$. □

- If a protocol respects $RF^{Eve,Abs,Behavior}$, it also respects $VP^{Eve,Abs,Behavior}$.

Proof. The proof is similar to the proof of Receipt-Freeness implies Vote-Privacy and Vote-Independence with passive Collaboration implies Vote-Independence in the DKR-model [11, 16]:
By hypothesis there is a closed plain process so that

$$V'^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}$$

and

$$A\left[\mathcal{S}\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right] \approx_l A\left[\mathcal{S}\left[V'|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}|\mathcal{V}_C\right]\right]$$

We apply the context $\nu chc(_|!\mathtt{in}(chc,x))$ on both sides, which gives

$$A\left[\mathcal{S}\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right]^{\backslash out(chc,\cdot)} \approx_l A\left[\mathcal{S}\left[V'|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}|\mathcal{V}_C\right]\right]^{\backslash out(chc,\cdot)}$$

By using Lemma 7 we obtain

$$A\left[\mathcal{S}\left[V'|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v_A}|\mathcal{V}_C\right]\right]^{\backslash out(chc,\cdot)} \equiv A\left[\mathcal{S}\left[V'^{\backslash out(chc,\cdot)}|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}|\mathcal{V}_C\right]\right]$$

and

$$A\left[\mathcal{S}\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right]^{\backslash out(chc,\cdot)} \equiv A\left[\mathcal{S}\left[\left(V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}^{chc}\right)^{\backslash out(chc,\cdot)}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right]$$

We can now apply Lemma 6 and use the fact that labeled bisimilarity is closed under structural equivalence and obtain

$$A\left[\mathcal{S}\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathcal{V}_C\right]\right] \approx_l A\left[\mathcal{S}\left[V'^{\backslash out(chc,\cdot)}|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}|\mathcal{V}_C\right]\right]$$

where we can chose $V'' = V'^{\backslash out(chc,\cdot)}$ to conclude.

□

B Proof of Proposition 2

We use the following equational theory:

$$\begin{aligned}
\mathtt{pdec}(\mathtt{penc}(m,\mathtt{pk}(sk),r),sk) &= m\\
\mathtt{pet}(\mathtt{penc}(m,\mathtt{pk}(sk),r),\mathtt{penc}(m,\mathtt{pk}(sk),r'),sk) &= \mathtt{true}\\
\mathtt{pet}(\mathtt{penc}(m,\mathtt{pk}(sk),r),\mathtt{penc}(m',\mathtt{pk}(sk),r'),sk) &= \mathtt{false}\\
\mathtt{first}(x,y) &= x\\
\mathtt{second}(x,y) &= y\\
\mathtt{checksign}(\mathtt{sign}(m,sk),pk(sk)) &= m
\end{aligned}$$

The equation models the public key encryption; the following two equations express plaintext equivalence tests. The `first` and `second` functions are used to build lists, and the last equation models signatures.

Listings 1-7 give the processes modeled in the applied pi calculus using the ProVerif syntax.

```
let pAdmin =
 in(skaCh, ska);
 in(pktCh, pkt);
 new cred1; new cred2; new cred3;
 out(credCh1, cred1);
 out(credCh2, cred2);
 out(credCh3, cred3);
 new r1; new r2; new r3;
 let cred1enc = penc(cred1, pkt, r1) in
 let cred2enc = penc(cred2, pkt, r2) in
 let cred3enc = penc(cred3, pkt, r3) in
 let voterlist = (cred1enc, (cred2enc, (cred3enc, nil))) in
 out(chLC, (sign(voterlist, ska), voterlist)).
```

Listing 1: The administrator process

The administrator The administrator (Process 1) receives his secret key and the tallier's public key. He generates a credential for each voter, sends them over private channels to the voters and publishes a signed and encrypted list of the credentials.

The voter The voter process (Process 2) receives his credential and the tallier's public key. Depending of the values of $vote$, v and $realvote$ (which can be determined using a substitution), he will vote or post fake ballots several times.

A voter under control of the attacker A voter controlled by the attacker is modeled by Process 3. This process can be obtained by calculating $\texttt{processV}^{c_1,c_2}$ as defined in Definition 5.

The resisting Voter The voter trying to resist coercion (Process 4) looks like the $\texttt{processV}^{c_1,c_2}$ (Process 3), but fakes the credential sent to the coercer and posts his real ballot using the real credential.

The mixer The mixer (Process 5) receives the ballots and stores them in a list. If he receives a ballot that reuses a credential, he will ignore it. After having received all m ballots, he mixes them (modeled as publishing all of them in parallel).

The tallier The tallier process (Process 6) receives the mixed ballots and the signed list of credentials. He checks the ballots for valid credentials using plaintext equivalence tests. If the credential is correct, the ballot is opened, otherwise discarded.

The main process The main process (Process 7) shows how the participation processes (three voters, the administrator, the mixer and the tallier) are combined in parallel.

The keying process The keying process (Process 7) generates the keys and distributes them.

Proof. To show that the protocol by Juels et al. fulfills $CR^{I,FA,EB}$, we have to show that for any voting process S and any substitution σ_{f_A} and σ_{f_C}, and for any context A such that $A = \nu\tilde{ch}.(_|A'^{chout})$ where $\tilde{ch}$ are all unbound channels and names in A' and in the "hole", there exist substitutions σ_{f_B}, $\sigma_{f'_B}$ and $\sigma_{f'_A}$, and a process V' such that for all votes σ_{v_A} and σ_{v_B} where σ_{v_B} does not make a voter abstain there exists a vote $\sigma_{v'_A}$ and the following holds: For any context C with $C = \nu c_1.\nu c_2.(_|P')$ and $\tilde{n} \cap fn(C) = \emptyset$,

$$S\left[C\left[V\sigma_{id_A}^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|V\sigma_{id_C}^{c_1,c_2}\right] \approx_l S\left[V\sigma_{id_A}\sigma_{f_A}\sigma_{v_A}^{chc}|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|V\sigma_{id_C}^{c_1,c_2}\right]$$

we have

```
let pVoter =
 in(credCh, cred);
 in(pktCh, pkT);
 if first(vote) = true then
 let vote = second(vote) in pVoter'.

let pVoter' =
 new fake; new r; new r';
 if first(realvote) = true then pVoter''
 else pVoter'''.

let pVoter' =
 out(bbCh, (penc(cred, pkT, r), penc(first(v), pkT, r')));
 if first(vote) = true then
 let vote = second(vote) in
 let realvote = second(realvote) in
 let v = second(v) in pVoter'.

let pVoter'' =
 out(bbCh, (penc(fake, pkT, r), penc(first(v), pkT, r')));
 if first(vote) = true then
 let vote = second(vote) in
 let realvote = second(realvote) in
 let v = second(v) in pVoter'.
```

Listing 2: The voter's process

- $C\left[V'\right]^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}$
- $A\left[S\left[C\left[V\sigma_{id_A}^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|V\sigma_{id_C}^{c_1,c_2}\right]\right] \approx_l A\left[S\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}|V\sigma_{id_C}^{c_1,c_2}\right]\right]$.

In our model, we will show that

$$C\left[\mathtt{pVoterR}\right]^{\setminus out(chc,\cdot)} \approx_l \mathtt{pVoter}\sigma_{id_A}\sigma_{f'_A}\sigma_{v_B}$$

and

$$A\left[\nu\tilde{ch}.(\mathtt{pKey}|\mathtt{pTally}|\mathtt{pMixer}|\mathtt{pAdmin}|C\left[\mathtt{pVoterC1C2}\sigma_{id_A}\right]|\mathtt{pVoter}\sigma_{id_B}\sigma_{f_B}\sigma_{v_B}|\mathtt{pVoterC1C2}\sigma_{id_C})\right]$$
$$\approx_l$$
$$A\left[\nu\tilde{ch}.(\mathtt{pKey}|\mathtt{pTally}|\mathtt{pMixer}|\mathtt{pAdmin}|C\left[\mathtt{pVoterR}\right]|\mathtt{pVoter}\sigma_{id_B}\sigma_{f'_B}\sigma_{v'_A}|\mathtt{pVoterC1C2}\sigma_{id_C})\right]$$

where $\nu\tilde{ch} = \nu pktCh.\nu sktCh.\nu pkaCh.\nu skaCh.\nu credCh1.\nu credCh2.\nu credCh3$, $\sigma_{id_A} = \{^{credCh1}/_{credCh}\}$, $\sigma_{id_B} = \{^{credCh2}/_{credCh}\}$ and $\sigma_{id_C} = \{^{credCh3}/_{credCh}\}$. Technically this proof is only correct for the case of three voters, a similar proof can however be made for any number of voters.

Note that, without loss of generality, we will suppose that σ_{v_A} contains only one vote or abstention, and that σ_{f_A} contains no fakes. If fakes are used, they can easily be counterbalanced in $\sigma_{f'_A}$. To show that the bisimilarity holds, we will discuss different cases.

1. *Abstention of* $V\sigma_{id_A}$. If $V\sigma_{id_A}$ abstains, he will receive his credential and the administrator's public key and forward them to the attacker on channel $c1$, and stop. The attacker has now two options:

   (a) He ignores the credential. Then $\sigma_{v'_A}$ has to be abstention as well. Thus, in the left hand case

```
let pVoterC1C2 =
 in(credCh, cred);
 out(c1, cred);
 in(pktCh, pkT);
 out(c1, pkT);
 in(c2, m);
 if m = true then pVoterC1C2'.

let pVoterC1C2' =
 new fake; out(c1, fake);
 new r; out(c1, r);
 new r'; out(c1, r');
 in(c2, m1);
 if m1 = true then pVoterC1C2''
 else pVoterC1C2'''.

let pVoterC1C2'' =
 in(c2, m2);
 out(bbCh, m2);
 in(c2, m3);
 if m3 = true then pVoterC1C2'.

let pVoterC1C2''' =
 in(c2, m4);
 out(bbCh, m4);
 in(c2, m5);
 if m5 = true then pVoterC1C2'.
```

Listing 3: The process of a voter under control of the attacker

```
let pVoterR =
 in(credCh, cred);
 new fake_cred;
 out(c1, fake_cred);
 in(pktCh, pkT);
 out(c1, pkT);
 out(bbCh, (penc(cred, pkT, r''), penc(v, pkT, r''')));
 in(c2, m);
 if m = true then pVoterR'.

let pVoterR' =
 new fake; out(c1, fake);
 new r; out(c1, r);
 new r'; out(c1, r');
 in(c2, m1);
 if m1 = true then pVoterR''
 else pVoterR'''.

let pVoterR'' =
 in(c2, m2);
 out(bbCh, m2);
 in(c2, m3);
 if m3 = true then pVoterR'.

let pVoterR''' =
 in(c2, m4);
 out(bbCh, m4);
 in(c2, m5);
 if m5 = true then pVoterR'.
```

Listing 4: The process V' resisting coercion

```
let pMixer =
 in(sktCh, skt);
 let count = zero in
 let clist = nil in pMixer'.

let pMixer' =
 if count = m then
  let list = clist in pMix
 else pMixer''.

let pMixer'' =
 in(bbCh, ballot);
 let (cred, vote) = ballot in
 let list = clist in pInList.

let pInList =
 if list <> nil then
  if pet(first(first(list)), cred, skt) = false then
   let list = second(list) in pInList
  else
   pMixer'
 else
  let clist = (ballot, clist) in
  let count = suc(count) in pMixer'.

let pMix =
 new r1; new r2;
 (out(bb2Ch, (reenc(first(first(list)), r1),
      reenc(second(first(list)), r2))) |
 (if second(list) <> nil then
  let list = second(list) in pMix)).
```

Listing 5: The mixer and his subprocesses

```
let pTally =
 in(pkaCh, pka);
 in(sktCh, skt);
 in(chLC, (signature, listcred));
 if checksign(signature, pka) = listcred then
 ! pOpen.

let pOpen =
 in(bb2Ch, (cred, vote));
 pOpen'.

let pOpen' =
 if listcred <> nil then
  if pet(first(first(listcred)), cred, skt) = false then
   let listcred = second(listcred) in pOpen'
  else
   out(bb3Ch, pdec(vote, skt)).
```

Listing 6: The tallier

```
process
new pktCh; new sktCh; new pkaCh; new skaCh;
new credCh1; new credCh2; new credCh3;
(pKey | pTally | pMixer | pAdmin |
 (let credCh = credCh1 in let v = a in pVoter) |
 (let credCh = credCh2 in let v = b in pVoter) |
 (let credCh = credCh3 in let v = c in pVoterC1C2))
```

Listing 7: The main process

```
let pKey =
 new skt; new ska;
 (out(sktCh, skt) | out(sktCh, skt) |
  out(skaCh, ska) | out(pkaCh, pka) |
  out(pktCh, pk(skt)) | out(pktCh, pk(skt)) |
  out(pktCh, pk(skt)) | out(pktCh, pk(skt)))
```

Listing 8: The keying process

we have the following frame[4] (modulo the behavior of V_{id_C}):

$$\phi_l^1 = \nu cred_1.\nu cred_2.\nu r.\nu r'.(\{^{cred_1}/_{x_1}\} \mid \{^{pkT}/_{x_2}\} \mid \{^{(penc(cred_2,pkT,r),penc(b,pkT,r'))}/_{x_3}\}) \tag{2}$$

and in the right hand case

$$\phi_r^1 = \nu cred_1.\nu fake_cred.\nu r.\nu r'.(\{^{fake_cred}/_{x_1}\} \mid \{^{pkT}/_{x_2}\} \mid \{^{(penc(cred_1,pkT,r),penc(b,pkT,r'))}/_{x_3}\}) \tag{3}$$

These frames are obviously statically equivalent, and remain statically equivalent for any behavior of V_{id_C}. The mixers will mix the votes, and after tallying on both sides one vote for b will be counted.

(b) He uses the credential to vote $c \neq b$. Then $\sigma_{v'_A}$ and $\sigma_{f'_A}$ have to counterbalance this, which is possible as we can chose it depending on the adversary's behavior. We obtain the following frame (modulo the behavior of V_{id_C}):

$$\phi_l^2 = \nu cred_1.\nu cred_2.\nu fake.\nu r_1.\nu r_2.\nu r_3.\nu r_4.(\{^{cred_1}/_{x_1}\} \mid \{^{pkT}/_{x_2}\} \mid \{^{(penc(cred_2,pkT,r_1),penc(b,pkT,r_2))}/_{x_3}\} \mid \{^{(penc(fake,pkT,r_3),penc(c,pkT,r_4))}/_{x_4}\}) \tag{4}$$

and a vote $(penc(x_1, pkT, r), penc(c, pkT, r'))$ by the attacker in the left hand case. In the right hand case we have

$$\phi_r^2 = \nu cred_1.\nu fake_cred.\nu cred_2.\nu r_1.\nu r_2.\nu r_3.\nu r_4.(\{^{fake_cred}/_{x_1}\} \mid \{^{pkT}/_{x_2}\} \mid \{^{(penc(cred_1,pkT,r_1),penc(b,pkT,r_2))}/_{x_3}\} \mid \{^{(penc(cred_2,pkT,r_3),penc(c,pkT,r_4))}/_{x_4}\}) \tag{5}$$

and a vote $(penc(x_1, pkT, r), penc(c, pkT, r'))$. The frames are statically equivalent and on both sides we have one fake, and a valid vote for b and c each. Thus the mixing and tallying will remain equivalent as well.

2. *Voting of* $V\sigma_{id_A}$. In that case, the condition on C ensures that he forces the targeted voter to vote σ_{v_a} using the correct credential. We will consider the same two subcases:

[4] x_i is the variable associated with the i-th output in an α-transition.

(a) The attacker ignores the revealed credential. Then $\sigma_{v'_A} = \sigma_{v_A}$. Thus, in the left hand case we have the following frame (modulo the behavior of V_{id_C}):

$$\begin{aligned}\phi_l^3 = \nu cred_1.\nu cred_2.\nu r_1.\nu r_2.\nu r_3.\nu r_4.\nu r_5.\nu r_6.(\{^{cred_1}/_{x_1}\} \mid \{^{pkT}/_{x_2}\} \mid \\ \{^{(penc(cred_1,pkT,r_1),penc(a,pkT,r_2))}/_{x_3}\}) \mid \{^{(penc(fake,pkT,r_3),penc(a,pkT,r_4))}/_{x_4}\}) \mid \\ \{^{(penc(cred_2,pkT,r_5),penc(b,pkT,r_6))}/_{x_5}\})\end{aligned} \tag{6}$$

and in the right hand case

$$\begin{aligned}\phi_r^3 = \nu cred_1.\nu cred_2.\nu fake_cred.\nu r_1.\nu r_2.\nu r_3.\nu r_4.\nu r_5.\nu r_6.(\{^{fake_cred}/_{x_1}\} \mid \{^{pkT}/_{x_2}\} \mid \\ \{^{(penc(cred_1,pkT,r_1),penc(b,pkT,r_2))}/_{x_3}\}) \mid \{^{(penc(fake_cred,pkT,r_3),penc(a,pkT,r_4))}/_{x_4}\}) \mid \\ \{^{(penc(cred_2,pkT,r_5),penc(a,pkT,r_6))}/_{x_5}\})\end{aligned} \tag{7}$$

In the left hand case, σ_{f_b} counterbalances the fact that on the right side, the targeted voter will post a fake. The resulting frames are obviously statically equivalent. The mixers will mix the votes, and after tallying on both sides one vote for a and one vote for b will be counted.

(b) He uses the credential to vote c. Then $\sigma_{v'_A}$ and $\sigma_{f'_A}$ have to counterbalance this, which is possible as we can chose it depending on the adversary's behavior. We obtain the following frame (modulo the behavior of V_{id_C}):

$$\begin{aligned}\phi_l^4 = \nu cred_1.\nu cred_2.\nu r_1.\nu r_2.\nu r_3.\nu r_4.\nu r_5.\nu r_6.(\{^{cred_1}/_{x_1}\} \mid \{^{pkT}/_{x_2}\} \mid \\ \{^{(penc(cred_1,pkT,r_1),penc(a,pkT,r_2))}/_{x_3}\}) \mid \{^{(penc(fake,pkT,r_3),penc(a,pkT,r_4))}/_{x_4}\}) \mid \\ \{^{(penc(cred_2,pkT,r_5),penc(b,pkT,r_6))}/_{x_5}\})\end{aligned} \tag{8}$$

and a vote $(penc(x_1, pkT, r), penc(c, pkT, r'))$ by the attacker in the left hand case. In the right hand case we have

$$\begin{aligned}\phi_r^4 = \nu cred_1.\nu cred_2.\nu fake_cred.\nu r_1.\nu r_2.\nu r_3.\nu r_4.\nu r_5.\nu r_6.(\{^{fake_cred}/_{x_1}\} \mid \{^{pkT}/_{x_2}\} \mid \\ \{^{(penc(cred_1,pkT,r_1),penc(b,pkT,r_2))}/_{x_3}\}) \mid \{^{(penc(cred_2,pkT,r_3),penc(c,pkT,r_4))}/_{x_4}\}) \mid \\ \{^{(penc(cred_2,pkT,r_5),penc(a,pkT,r_6))}/_{x_5}\})\end{aligned} \tag{9}$$

and a vote $(penc(x_1, pkT, r), penc(c, pkT, r'))$. The frames are statically equivalent and, on both sides, we have one fake, a valid vote for b, two concurring votes (i.e. using the same credentials) for a and c each. As the decision which concurring vote will be counted depends on the network, both sides are bisimilar as both cases are possible on each side.

Note that the "insider" $V_{id_C}^{c1,c2}$ does not help the attacker to distinguish both sides. He cannot copy votes as the votes are sent over an anonymous channel and cannot be linked to the voter who sent them (as there are no signatures etc.). Having a correct credential alone is therefore not helping the attacker. □