Vote-Independence: A Powerful Privacy Notion for Voting Protocols

Jannik Dreier, Pascal Lafourcade, Yassine Lakhnech

Vote-Independence: A Powerful Privacy Notion for Voting Protocols*

Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech

Université Grenoble 1, CNRS, Verimag, FRANCE
firstname.lastname@imag.fr

Abstract. Recently an attack on ballot privacy in Helios has been discovered [20], which is essentially based on copying other voter's votes. To capture this and similar attacks, we extend the classical threat model and introduce a new security notion for voting protocols: Vote-Independence. We give a formal definition and analyze its relationship to established privacy properties such as Vote-Privacy, Receipt-Freeness and Coercion-Resistance. In particular we show that even Coercion-Resistant protocols do not necessarily ensure Vote-Independence.

Keywords: Electronic Voting, Privacy, Anonymity, Security, Formal Verification, Coercion-Resistance, Receipt-Freeness

1 Introduction

Electronic voting schemes are systems that allow casting and tallying votes using machines. This promises to improve efficiency by providing results faster, using less personnel or adding comfort (e.g. the possibility to vote from home). However the recent use of commercial electronic voting systems for presidential or general elections in many countries has spread controversy on security issues [5, 6, 17, 22]. Primary concerns are verifiability (the possibility to verify the elections's outcome, i.e. to check if all votes have been counted correctly) and privacy (i.e. anonymity of the voter, secrecy of the vote). To address this issues, many different protocols have been developed to fulfill security requirements such as

- *Eligibility*: Only the registered voters can vote, and nobody can vote more than once.
- *Fairness*: No preliminary results are available which could influence other voters' decisions.
- *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.

* This work was partially supported by the ANR project AVOTE. An extended version containing the detailed proofs is available as a technical report [9]. The original publication is available on www.springerlink.com: http://www.springerlink.com/content/r8x1v14298647022/

- *Vote-Privacy*: The votes are kept private.
- *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 the voter followed his instructions or actually voted for another candidate.
- *Robustness*: The protocol should be able to tolerate a certain number of misbehaving voters.

A common aim is to verify these properties using formal models and definitions. This concerns privacy properties (privacy, receipt-freeness and coercion-resistance) [7, 8, 16], election verifiability [14, 21], or both [11–13]. We concentrate on privacy-type properties of voting protocols (i.e. Vote-Privacy, Receipt-Freeness and Coercion-Resistance).

While analyzing privacy in Helios [2], a web based voting system, B. Smyth and V. Cortier [19, 20] recently discovered an attack based on the possibility for an attacker to copy another voter's vote and to submit it as his own. If the number of participating voters is small or if a noticeable fraction of voters can be corrupted, this can break privacy as the contents of the vote can be inferred from the published election outcome. For example in the case of three voters (two honest ones and one under the control of the attacker), the attacker can try to copy the vote of the first honest voter. The candidate chosen by the targeted voter will then have at least two votes, and can thus be recognized in the official outcome. This reveals the content of the targeted vote.

Our Contributions. Based on this attack, we extend the established threat model and give a formal definition for the notion of "Vote-Independence (VI)" in the applied pi calculus. We show that our definition of "Vote-Independence" implies Vote-Privacy as defined in the literature [7] and that the concept can be generalized to improve Receipt-Freeness and Coercion Resistance as well. We define "Vote-Independence with passive Collaboration (VI-PC)" (which corresponds to Vote-Independence in same setting as Receipt-Freeness, i.e. with passive collaboration of the voter) and "Vote-Independence with active Collaboration (VI-AC)" (which corresponds to Coercion Resistance). We prove the hierarchy of the definitions and illustrate each level with a real world example: the protocol by Fujioka et al. [10] provides Vote-Independence (VI), the protocol due to Okamoto [18] ensures VI-PC, and the protocol by Bohli et al. [4] guarantees VI-AC. We also show that even Coercion-Resistant protocols may not ensure Vote-Independence (i.e. that our definitions are strictly stronger than the usual privacy notions) by analyzing the protocol by Lee at al. [15].

Outline of the Paper. The remainder of the paper is structured as follows. In Section 2, we recall the applied pi calculus and the standard privacy definitions. In Section 3 we elaborate our definitions of Vote-Independence. Then we analyze the hierarchy of our definitions, the relation to standard privacy properties and

discuss several examples in Section 4. Finally, we conclude and discuss future work.

2 Preliminaries

In this section we introduce the applied pi calculus, define our model of voting processes and recall existing privacy definitions.

2.1 The Applied Pi Calculus

We use the applied pi calculus [1] to model our security properties and the protocols to analyze. The calculus is an abstract language to describe concurrent processes and interactions, and is supported by the tool "ProVerif" [3].

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. We distinguish the type "channel" from other *base* types. To model equalities we use an equational theory E which defines a relation $=_E$. A classical example for symmetric encryption is $\mathtt{dec}(\mathtt{enc}(message, key), key) =_E message$.

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

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

and extended by *Internal reduction* ($\rightarrow$), the smallest relation on extended processes closed under 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 labelled operational semantics ($\xrightarrow{\alpha}$) where α can be an input or the 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	$\dfrac{A \xrightarrow{\alpha} A' \qquad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$
STRUCT	$\dfrac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'}$

Labelled transitions are not closed under the evaluation contexts. Note that a term M cannot be output directly, but only a variable as "reference" to it. 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 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 $dom(\phi) = dom(\psi)$ 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 simple: 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 *labelled bisimilarity*.

Definition 3 (Labelled Bisimilarity ($\approx_l$)). *Labelled 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 $A \rightarrow A'$, then $B \rightarrow B'$ 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 $B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'$ 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. Labelled bisimilarity implies "classic" bisimilarity [1], but is often easier to prove and can be used to express many classical security properties, in particular anonymity properties.

2.2 Voting Process

We use the definition by Delaune et al. [7] to model voting protocols in the applied pi calculus. The basic idea is simple: A voting process is the parallel composition of all voters and the trusted authorities, whereas untrusted authorities are left to the context (i.e. the attacker). Messages are exchanged over public or private channels. We limit ourselves to protocols where each voter votes only once.

Definition 4 (Voting Process [7]). *A voting process is a closed plain process*

$$VP \equiv \nu\tilde{n}.(V\sigma_1 | \ldots | V\sigma_n | A_1 | \ldots | A_m).$$

The $V\sigma_i$ are the voter processes, the A_js the honest election authorities, and the $\tilde{n}$ are channel names. We also suppose that $v \in dom(\sigma_i)$ is a variable which refers to the value of the vote. We define an evaluation context S which is like VP, but has a hole instead of three $V\sigma_i$, and an evaluation context S' which is like VP, but has a hole instead of two $V\sigma_i$.

Note that S and S' contain – by construction – only honest voters, i.e. voters that follow the protocol and do not collude with the attacker.

2.3 Privacy

Before discussing Vote-Independence, we recall the definition of the three basic privacy properties as given by Delaune et al. [7].

Vote-Privacy. The intuition for Vote-Privacy is the following: An attacker cannot distinguish two runs of the voting protocols where two voters swap their votes. This does not change the outcome and if the votes are private, the attacker should not know which vote belongs to which voter:

Definition 5 (Vote-Privacy [7]). *A voting process respects* Vote-Privacy (P) *if for all votes a and b*

$$S' \left[V_A \left\{{}^{a}/_{v}\right\} | V_B \left\{{}^{b}/_{v}\right\}\right] \approx_l S' \left[V_A \left\{{}^{b}/_{v}\right\} | V_B \left\{{}^{a}/_{v}\right\}\right]$$

Receipt-Freeness. To define Receipt-Freeness, we use the transformation P^{ch} which can be applied to a process P. The transformed process outputs all its inputs and its private data (in particular new names, for example random values) on a special channel ch to the attacker. In the case of a voting process V, this corresponds to trying to create a receipt of the vote. If a protocol is receipt-free, a voter should be able to fake all these outputs to the coercer, i.e. to output fake values without the attacker noticing. This means that there should exist some process V' so that the attacker is not able to distinguish between a successfully coerced voter V^{ch} that votes c and outputs the correct values, and a voter V' that fakes the values and votes a instead. To ensure that the coercer cannot tell both cases apart from the result, Delaune et al. introduce another voter that counterbalances the vote, and require that V' actually votes for a using Definition 7.

Definition 6 (Process P^{ch} [7]). *Let P be a plain process and ch be a channel name. We define P^{ch} as follows:*

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

In the remainder we assume $ch \notin fn(P) \cup bn(P)$ before applying the transformation.

Definition 7 (Process $A^{\setminus out(ch,\cdot)}$ [7]). *Let A be an extended process. We define the process $A^{\setminus out(ch,\cdot)}$ as $\nu ch.(A|!in(ch, x))$.*

Definition 8 (Receipt-Freeness [7]). *A voting process respects* Receipt-Freeness (RF) *if there exists a closed plain process V' such that for all votes a and c we have*

$$V'^{\backslash out(chc,\cdot)} \approx_l V_A\{^a/_v\}$$

and

$$S'\left[V_A\{^c/_v\}^{chc} | V_B\{^a/_v\}\right] \approx_l S'[V' | V_B\{^c/_v\}]$$

Coercion-Resistance. Similarly to Receipt-Freeness, Delaune et al. define a process that outputs all its inputs and secret values to the attacker. To express interactive coercion, it additionally waits for input from the context that tells it what to do before outputting values or branching (Definition 9).

Definition 9 (Process P^{c_1,c_2} [7]). *Let P be a plain process and c_1, c_2 be channel names. We define P^{c_1,c_2} as follows:*

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

The definition then follows the same basic idea as for Receipt-Freeness: there exists a process V' that can interact with the attacker and fake all necessary messages without the attacker noticing. Yet one has to add some condition to ensure that the attacker cannot distinguish both sides of the bisimilarity simply based on the result by forcing the coerced voter to vote d, which would change the outcome. Tho enforce this, Delaune et al. use a context C that models the part of the attacker which interacts with V_A. The conditions on C ensure that the attacker actually forces the voter to vote c, and not d and thus make sure the vote is counterbalanced by V_B.

Definition 10 (Coercion-Resistance [7]). *A voting process respects* Coercion-Resistance (CR) *if there exists a closed plain process V' such that for any $C = \nu c_1.\nu c_2.(\_|P)$ satisfying $\tilde{n} \cap fn(C) = \emptyset$ and $S[C[V_A\{^?/_v\}^{c_1,c_2}]|V_B\{^a/_v\}] \approx_l S\left[V_A\{^c/_v\}^{chc}|V_B\{^a/_v\}\right]$ we have for all votes a and c*

$$C[V']^{\backslash out(chc,\cdot)} \approx_l V_A\{^a/_v\}$$

and

$$S'[C[V_A\{^?/_v\}^{c_1,c_2}]|V_B\{^a/_v\}] \approx_l S'[C[V']|V_B\{^c/_v\}]$$

Note that we write $\{^?/_v\}$ to represent the fact that the coerced voters vote does not depend on the substitution, but on the interaction with the context C.

3 Vote-Independence

In the previous privacy definitions the attacker has the role of an outside observer that tries to infer something about someone's vote. In the case of Coercion-Resistance or Receipt-Freeness he might communicate with the targeted voter, but he cannot necessarily vote himself or collude with other voters - unlike what would generally happen in real-world elections.

To address this shortcoming and obtain a more realistic model of the attacker's abilities, we introduce the notion of Vote-Independence for different levels of collaboration. The idea is to extend the existing definitions to the case where the attacker can vote himself and might try to relate his vote to the vote of a targeted voter to compromise privacy (for example copy it as in the attack by B. Smyth and V. Cortier [20]).

3.1 Vote-Independence (without Collaboration)

Definition 11 (Vote-Independence). *A voting process respects* Vote - Independence (VI) *if for all votes a and b*

$$S\left[V_A\left\{{}^{a}/_{v}\right\}|V_B\left\{{}^{b}/_{v}\right\}|V_C^{c_1,c_2}\right] \approx_l S\left[V_A\left\{{}^{b}/_{v}\right\}|V_B\left\{{}^{a}/_{v}\right\}|V_C^{c_1,c_2}\right]$$

The intuition behind our definition is the following: We start from the definition of privacy, but add a voter under the control of the attacker in both cases. If an attacker can relate his vote to the vote of one of the voters (for example copy V_A's vote, i.e. vote for the same candidate), he will be able to distinguish both sides as the result of the vote will be different. This is the most basic definition, as the attacker has only access to publicly available data. Subsequently we add the possibility of collaborating voters.

Smyth and Cortier [20] used a similar idea in a recent extension to their original paper. Contrary to our definition, they implicitly include corrupted voters in the context S (or S' resp.). We chose to make the corrupted voter explicit to be able to easily compare both notions.

3.2 Vote-Independence with Passive Collaboration

Definition 12 (Vote-Independence with Passive Collaboration). *A voting process respects* Vote-Independence with Passive Collaboration (VI-PC) *if there exists a closed plain process V' such that for all votes a and c*

$$V'\backslash out(chc,\cdot) \approx_l V_A\left\{{}^{a}/_{v}\right\}$$

and

$$S\left[\left[V_A\left\{{}^{c}/_{v}\right\}^{chc}|V_B\left\{{}^{a}/_{v}\right\}|V_C^{c_1,c_2}\right] \approx_l S\left[V'|V_B\left\{{}^{c}/_{v}\right\}|V_C^{c_1,c_2}\right]\right.$$

Vote-Independence with Passive Collaboration can be seen analogously to Receipt-Freeness. The attacker should not be able to link his vote to another voter's vote,

even if this voter collaborates with him and gives him access to his secret values after voting (secret keys, random values, nonces, etc.). This is ensured in the definition as the attacker cannot decide if he is in a case where the attacked voter actually collaborates with him, or if the voter only pretends to collaborate and in reality votes differently. If he could use the information provided by the attacked voter to e.g. copy his vote, he would be able to distinguish these cases.

3.3 Vote-Independence with Active Collaboration

Definition 13 (Vote-Independence with Active Collaboration). *A voting process respects* Vote-Independence with Active Collaboration (VI-AC) *if there exists a closed plain process V' such that for all votes a and c and for any $C = \nu c_1.\nu c_2.(\_|P)$ satisfying $\tilde{n} \cap fn(C) = \emptyset$ and*

$$S\left[C\left[V_A\left\{{}^{?}/_{v}\right\}^{c_1,c_2}\right]|V_B\left\{{}^{a}/_{v}\right\}|V_C^{c_3,c_4}\right] \approx_l S\left[V_A\left\{{}^{c}/_{v}\right\}^{chc}|V_B\left\{{}^{a}/_{v}\right\}|V_C^{c_3,c_4}\right]$$

we have

$$C\left[V'\right]^{\backslash out(chc,\cdot)} \approx_l V_A\left\{{}^{a}/_{v}\right\}$$

and

$$S\left[C\left[V_A\left\{{}^{?}/_{v}\right\}^{c_1,c_2}\right]|V_B\left\{{}^{a}/_{v}\right\}|V_C^{c_3,c_4}\right] \approx_l S\left[C\left[V'\right]|V_B\left\{{}^{c}/_{v}\right\}|V_C^{c_3,c_4}\right]$$

In this definition, the attacker is even more powerful. Similarly to Coercion-Resistance, he can interact with the attacked voter during the entire voting process.

4 Hierarchy and Relation to Privacy

4.1 Hierarchy

Intuitively, $VI - AC$ is a stronger property than $VI - PC$, which is a stronger property than VI. The following proposition confirms this intuition:

Proposition 1. *We have:*

- *If a protocol respects Vote-Independence with Active Collaboration, it also respects Vote-Independence with Passive Collaboration.*
- *If a protocol respects Vote-Independence with Passive Collaboration, it also respects Vote-Independence (without collaboration).*

The detailed proofs can be found in our technical report [9].

4.2 Relation to Privacy

The only difference between P and VI (or $VI-PC$ and RF, or $VI-AC$ and CR) is the process $V_C^{c_1,c_2}$, i.e. the existence of a legitimate voter that is under control of the attacker. Intuitively this gives the attacker more power and thus VI (or $VI-PC$ or $VI-AC$) should be the stronger property. Indeed:

Proposition 2. *We have:*

- *If a protocol respects Vote-Independence, it also respects Vote-Privacy.*
- *If a protocol respects Vote-Independence with Passive Collaboration, it also respects Receipt-Freeness.*
- *If a protocol respects Vote-Independence with Active Collaboration, it also respects Coercion-Resistance.*

Informally we can argue that any attack on Vote-Privacy can be used to break Vote-Independence. In this case the voter under control of the attacker simply behaves as a normal voter and the intruder can employ the same attack. The formal proof is given in our technical report [9]. $CR \Rightarrow RF \Rightarrow P$ has been shown in the literature [7].

4.3 The Global Picture

Taking these properties together, we obtain the following hierarchy of notions. $A \Rightarrow B$ means that any protocol ensuring property A also ensures property B.

$$\begin{array}{ccc} \text{CR [15]} & \begin{array}{c}\Leftarrow\\ \not\Leftarrow\end{array} & \text{VI-AC [4]} \\ \Downarrow\not\Uparrow & & \Downarrow\not\Uparrow \\ \text{RF} & \begin{array}{c}\Leftarrow\\ \not\Leftarrow\end{array} & \text{VI-PC [18]} \\ \Downarrow\not\Uparrow & & \Downarrow\not\Uparrow \\ \text{P} & \begin{array}{c}\Leftarrow\\ \not\Leftarrow\end{array} & \text{VI [10]} \end{array}$$

The cited protocols [4, 10, 15, 18] illustrate the hierarchy and show that the inverse implications are not true, as discussed below.

4.4 Example: FOO

The protocol by Fujioka et al. [10] is based on commitments and blind signatures. It was proven to respect Vote-Privacy (P) [7], but is not Receipt-Free (RF) as the randomness of the commitment can be used as a receipt. We show that it ensures Vote-Independence (VI).

Informal Description. The protocol is split in three phases. In the first phase, the voter obtains the administrator's signature on a 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, 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 actual voting takes place:

- Voter V_i sends (x_i, y_i) to the collector C through an anonymous channel.
- The collector checks the administrator's signature and enters (x_i, y_i) 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 r_i together with the commitment's index l on the list to C using an anonymous channel.
- The collector C opens the l-th ballot using r_i and publishes the vote.

Model in Applied Pi Calculus. Our model is based on the one developed in [7], but we add a third voter. We use the following equational theory:

$$\begin{aligned} \mathtt{open}(\mathtt{commit}(m, r), r) &= m \\ \mathtt{checksign}(\mathtt{sign}(m, sk), \mathtt{pk}(sk)) &= m \\ \mathtt{unblind}(\mathtt{blind}(m, r), r) &= m \\ \mathtt{unblind}(\mathtt{sign}(\mathtt{blind}(m, r), sk), r) &= \mathtt{sign}(m, sk) \end{aligned}$$

The complete model can be found in our technical report [9].

Analysis.

Proposition 3. *FOO respects Vote-Independence (VI).*

Proof. Similarly to the proof of Vote-Privacy by [7], we do not need to trust any authority except for the key distribution process. Thus the voter $V_C^{c_1,c_2}$ under control of the attacker only interacts with the attacker (as untrusted authorities are left to the context, i.e. the attacker), except during the key distribution process at the beginning. In this process he obtains his key (which we do not require to be secret) and the administrator's public key, which is available to the attacker anyway. Thus the attacker is essentially in the same situation as in the proof of Vote-Privacy. The full proof can be found in our technical report [9]. □

Note that this protocol cannot respect Vote-Independence with Passive or Active Collaboration ($VI-PC$ or $VI-AC$), as this would imply Receipt-Freeness (see the hierarchy). This shows that $VI \not\Rightarrow VI-PC$.

4.5 Example: Okamoto

The protocol by Okamoto [18] uses trap-door commitments to achieve receipt-freeness [7]. However it is not Coercion-Resistant (CR) [7].

Informal Description. The protocol is very similar to the one by Fujioka et al. [10] discussed above. The only difference is the use of a trap-door commitment and a timeliness member to open the commitments. The first phase - during which the voter obtains a signature on his commitment - follows the same protocol, except for the fact that this time ξ is a trapdoor-commitment. In the second phase the actual voting takes place:

- Voter V_i sends the signed trap-door commitment to the collector C through an anonymous channel.
- The collector checks the administrator's 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 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 $x_{\pi(i)} = \xi(v_i, r_i)$.

Model in Applied Pi Calculus. Our model is based on the model used in [7], but we add a third voter. It is based on the following equational theory:

$$\begin{aligned}
\mathtt{open}(\mathtt{tdcommit}(m, r, td), r) &= m \\
\mathtt{tdcommit}(m_1, r, td) &= \mathtt{tdcommit}(m_2, f(m_1, r, td, m_2), td) \\
\mathtt{checksign}(\mathtt{sign}(m, sk), \mathtt{pk}(sk)) &= m \\
\mathtt{unblind}(\mathtt{blind}(m, r), r) &= m \\
\mathtt{unblind}(\mathtt{sign}(\mathtt{blind}(m, r), sk), r) &= \mathtt{sign}(m, sk)
\end{aligned}$$

The first equation models the creation of a trap-door commitment to m using a random value r and a trap-door td, whereas the second equation allows the construction of another random value to open a commitment differently. This requires knowledge of the trap-door td and the initial random value r.

Analysis.

Proposition 4. *The protocol by Okamoto respects* $VI - PC$.

Proof. To prove this, we need to find a process V' that successfully fakes all secrets to a coercer. In addition to normal receipt-freeness, we also have to ensure that the attacker cannot use the secrets to e.g. copy the vote of the targeted voter.

In this protocol the trap-door commitment allows the voter to return a faked random number to the attacker which opens the commitment to any value the voter wants. This means that - although the attacker has access to the commitment and the necessary values to open it - he will always open it in a way that yields a vote for c due to the fake randomness, even if the voter actually committed to a. The same reasoning applies for copying votes: Although it is technically possible to copy the vote of the targeted voter, the voter will provide a faked random value, which will make the timeliness member open the vote as a vote for c. This makes it impossible for the attacker to know if the voter complied with his instructions or only pretended to do so, even if he tries to relate his vote to the targeted voter's vote. The detailed model and complete proof can be found in our technical report [9]. □

Thus the protocol also respects simple Vote-Independence (VI). Note that this protocol cannot respect Vote-Independence with Active Collaboration ($VI - AC$), as this would imply Coercion-Resistance. This shows that $VI - PC \not\Rightarrow VI - AC$.

4.6 Example: 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). We use Bingo Voting to illustrate the existence of protocols that respect Vote-Independence with active Collaboration ($VI - AC$).

Informal Description. We consider an election with k voters and l candidates. 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 k random values $n_{i,j}$ for every candidate p_j (the dummy votes). It commits to the $k \cdot l$ 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 a dummy vote, except for the voter's choice. For this candidate, the random value from the RNG is used and the receipt (a list of all candidates and the corresponding random numbers) 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.

Model in Applied Pi Calculus. As we are only interested in privacy, we ignore the zero-knowledge proofs which are necessary to achieve verifiability. This yields a very simple equational theory:

$$\mathtt{open}(\mathtt{commit}(m, r), r) = m$$

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 can be found in the technical report [9].

Analysis.

Proposition 5. *Bingo Voting respects $VI - AC$.*

Proof. The receipts contain only random values which makes it impossible for the attacker to know if a certain number corresponds to the random value by the RNG or a dummy vote. Thus the voter V' does not even have to fake a receipt, he can simply forward his receipt and claim he voted for the coercer's choice. Constructing a related vote based on the receipt is not possible either since - while voting - the attacker has to transmit his choice in clear to the voting machine. Being able to e.g. copy V_A's vote would imply the break of simple privacy on the voter's vote using the receipt. The complete proof can be found in our technical report [9]. □

This implies that Bingo Voting is coercion resistant and provides Vote-Independence.

4.7 Example: Lee et al.

The protocol by Lee et al. [15] was proven to be Coercion-Resistant (CR) in [7], but does not respect Vote-Independence (VI) – and thus neither $VI - PC$ nor $VI - AC$ – as we show. It is based on trusted devices that re-encrypt ballots and use designated verifier proofs (DVPs) to prove their correct behavior to the voter.

Informal 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.

- 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.

Model in Applied Pi Calculus. Our model is based on the one developed in [7], but we add a third (corrupted) voter and an explicit mixing stage. This stage was left out in their model, but is essential to highlight the difference between Vote-Privacy and Vote-Independence. We use the following equational theory:

$$\begin{gathered}
\mathtt{decrypt}(\mathtt{penc}(m, \mathtt{pk}(sk), r), sk) = m \\
\mathtt{checksign}(\mathtt{sign}(m, sk), \mathtt{pk}(sk)) = m \\
\mathtt{rencrypt}(\mathtt{penc}(m, \mathtt{pk}(sk), r1), r2) = \mathtt{penc}(m, \mathtt{pk}(sk), f(r1, r2)) \\
\mathtt{checkdvp}(\mathtt{dvp}(x, \mathtt{rencrypt}(x, r), r, \mathtt{pk}(sk)), x, \mathtt{rencrypt}(x, r), \mathtt{pk}(sk)) = \mathtt{ok} \\
\mathtt{checkdvp}(\mathtt{dvp}(x, y, z, skv), x, y, \mathtt{pk}(skv)) = \mathtt{ok}
\end{gathered}$$

Analysis. In the extended model, the protocol by Lee et al. still ensures (CR), but it is not (VI).

Proposition 6. *The protocol by Lee et al. does not respect Vote-Independence (VI).*

Proof. As acknowledged by the authors in their original paper [15], 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. This obviously contradicts vote-independence. Our technical report [9] shows how this can be seen in the formal model. □

This example shows that vote-independence properties are strictly stronger than the corresponding privacy properties ($CR \not\Rightarrow VI - AC$, $RF \not\Rightarrow VI - PC$, $P \not\Rightarrow VI$), as even a coercion-resistant protocol fails to respect simple vote-independence.

5 Conclusion

Inspired by an attack based on copying votes, we extended the classical threat model and developed the notion of "Vote-Independence". We gave a formal definition and showed that it is stronger than standard vote-privacy. We generalized the definition to passive and active collaboration, and obtained refined properties on the same attack level as receipt-freeness and coercion-resistance.

Subsequently we analyzed practical examples which illustrate that our property is strictly stronger, i.e. that even coercion resistant protocols can fail with respect to Vote-Independence, and thus of practical interest.

Future Work. We plan to translate our symbolic definition to the computational model and extend our analysis e.g. to accommodate protocols permitting multiple votes. Additionally, it would be desirable to develop tools that at least partly automate and/or verify the necessary proofs.

References

1. Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '01, pages 104–115, New York, 2001. ACM.
2. Ben Adida, Olivier De Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a university president using open-audit voting: analysis of real-world use of helios. In *Proceedings of the 2009 conference on Electronic voting technology/workshop on trustworthy elections*, EVT/WOTE'09, pages 10–10, Berkeley, CA, USA, 2009. USENIX Association.
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, February–March 2008.
4. Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. Bingo voting: Secure and coercion-free voting using a trusted random number generator. In Ammar Alkassar and Melanie Volkamer, editors, *E-Voting and Identity*, volume 4896 of *Lecture Notes in Computer Science*, pages 111–124. Springer Berlin / Heidelberg, 2007.
5. UK Electoral Commission. Key issues and conclusions: May 2007 electoral pilot schemes. `http://www.electoralcommission.org.uk/elections/pilots/May2007`.
6. Bundesverfassungsgericht (Germanys Federal Constitutional Court). Use of voting computers in 2005 bundestag election unconstitutional, March 2009. Press release 19/2009 `http://www.bundesverfassungsgericht.de/en/press/bvg09-019en.html`.
7. Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17:435–487, December 2009.
8. Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols: A taster. In David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Mirosław Kutyłowski, and Ben

Adida, editors, *Towards Trustworthy Elections – New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*, pages 289–309. Springer, May 2010.

9. 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`.
10. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology – AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer Berlin / Heidelberg, 1992.
11. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections, 2002. Cryptology ePrint Archive, Report 2002/165, `http://eprint.iacr.org/`.
12. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, WPES '05, pages 61–70, New York, NY, USA, 2005. ACM.
13. Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *Proceedings of the 14th European Symposium On Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200. Springer, 2005.
14. Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *Proceedings of the 15th European Symposium on Research in Computer Security, ESORICS 2010*, volume 6345 of *Lecture Notes in Computer Science*, pages 389–404. Springer, 2010.
15. Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In Jong In Lim and Dong Hoon Lee, editors, *Information Security and Cryptology - ICISC 2003*, volume 2971 of *Lecture Notes in Computer Science*, pages 245–258. Springer Berlin / Heidelberg, 2004.
16. Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer, 2006.
17. Participants of the Dagstuhl Conference on Frontiers of E-Voting. Dagstuhl accord, 2007. `http://www.dagstuhlaccord.org/`.
18. Tatsuaki Okamoto. An electronic voting scheme. In *Proceedings of the IFIP World Conference on IT Tools*, pages 21–30, 1996.
19. Ben Smyth and Veronique Cortier. Attacking and fixing helios: An analysis of ballot secrecy. Accepted at CSF'11.
20. 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/`.
21. Ben Smyth, Mark D. Ryan, Steve Kremer, and Mounira Kourjieh. Towards automatic analysis of election verifiability properties. In Alessandro Armando and Gavin Lowe, editors, *Proceedings of the Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS'10)*, volume 6186 of *Lecture Notes in Computer Science*, pages 146–163, Paphos, Cyprus, October 2010. Springer.

22. 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. Press release http://www.minbzk.nl/onderwerpen/grondwet-en/verkiezingen/nieuws--en/112441/stemmen-met-potlood.