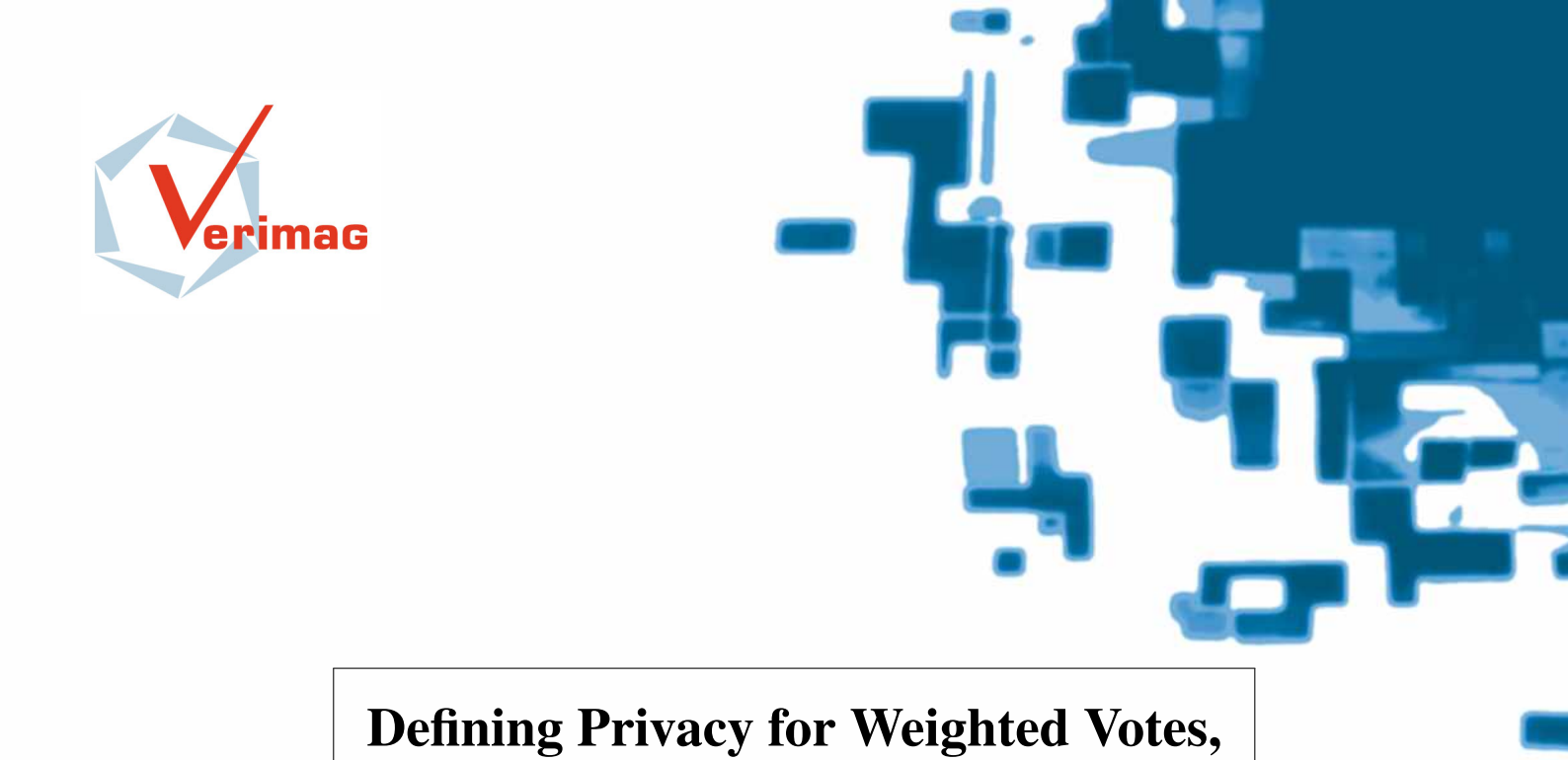
Defining Privacy for Weighted Votes, Single and Multi-voter Coercion

Jannik Dreier, Pascal Lafourcade, Yassine Lakhnech



Defining Privacy for Weighted Votes, Single and Multi-Voter Coercion

Jannik Dreier, Pascal Lafourcade, Yassine Lakhnech

Verimag Research Report n^o TR-2012-2

April 3, 2012

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

Defining Privacy for Weighted Votes, Single and Multi-Voter Coercion

Jannik Dreier, Pascal Lafourcade, Yassine Lakhnech

April 3, 2012

Abstract

Most existing formal privacy definitions for voting protocols are based on observational equivalence between two situations where two voters swap their votes. These definitions are unsuitable for cases where votes are weighted. In such a case swapping two votes can result in a different outcome and both situations become trivially distinguishable. We present a definition for privacy in voting protocols in the Applied π-Calculus that addresses this problem. Using our model, we are also able to define multi-voter coercion, i.e. situations where several voters are attacked at the same time. Then we prove that under certain realistic assumptions a protocol secure against coercion of a single voter is also secure against coercion of multiple voters. This applies for Receipt-Freeness as well as Coercion-Resistance.

Keywords: Electronic Voting, Privacy, Anonymity, Security, Formal Verification, Coercion-Resistance, Receipt-Freeness, Weighted Votes, Multi-Voter Coercion

How to cite this report:

```
@techreport {TR-2012-2,
  title = {Defining Privacy for Weighted Votes, Single and Multi-Voter Coercion},
  author = {Jannik Dreier, Pascal Lafourcade, Yassine Lakhnech},
  institution = {{Verimag} Research Report},
  number = {TR-2012-2},
  year = {2011}
}
```

1 Introduction

Privacy is a key requirement in elections as voters can otherwise be blackmailed, coerced or may be susceptible to vote-buying. Typically privacy is split into three different properties:

- *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.

The design of complex protocols such as voting protocols is known to be error-prone. This is why formal verification is an ideal tool to ensure the correctness and security of voting protocols. It has already been used to analyze properties such as Verifiability, Privacy, Receipt-Freeness and Coercion-Resistance [2, 5, 6, 8, 9, 19, 23, 27, 18, 28, 14, 15, 17].

However, most existing symbolic definitions of Privacy are based on the idea of swapping votes. If the votes are private, a case where Alice votes "yes" and Bob votes "no" should be indistinguishable from a case where Alice votes "no" and Bob votes "yes". Yet this definition is unsuitable for some situations, for example in companies where votes are weighted according to the proportion of shares held by each shareholder. Consider the following example: Alice owns 50% of the stocks, and Bob and Carol each hold 25%. The cases where Alice and Bob swap votes are now easily distinguishable if Carol votes "yes" all the time, as the result of the vote is different: 75% vs. 50% vote for "yes". Note that there are still situations where privacy is ensured in the sense that different situations give the same result. The last outcome (50% yes, 50% no) could - for example - also be announced if Alice votes "yes" and Bob and Carol vote "no". Protocols supporting vote weights have been proposed, for example Eliasson and Zúquete [11] developed a voting system supporting vote weights based on REVS [13], which itself is based on the protocol by Fujioka et al. [12].

Our Contributions: To address this issue, we define a symbolic privacy notion in the Applied π-Calculus [1] that takes weighted votes into account. Instead of requiring two executions where voters swap votes to be bisimilar, we require two executions to be bisimilar if they publish the same result, independent of the mapping between voters and votes. We analyze the relationship of our notion to the existing swap-based ones and give precise conditions for formally proving the equivalence between them. Then, we generalize our notion to Receipt-Freeness and Coercion-Resistance for weighted votes. We use a variant of the protocol by Eliasson and Zúquete [11] as a case study for our definition, and provide a partially automated proof using ProVerif [3].

In the cases of coercion most existing definitions only consider one attacked voter. Our model also allows to define a case with multiple coerced voters, and we analyze the relationship between this and the single-voter case. In particular, we give a formal proof that single- and multi-voter coercion are equivalent for a given protocol if it satisfies some modularity and de-composability properties. Using two existing protocols, we show that theses properties are realistic.

Related Work: Previous research on formal verification of voting protocols concerned privacy properties (privacy, receipt-freeness and coercion-resistance) [2, 5, 6, 8, 9, 19, 23, 27], election verifiability [18, 28], or both [14, 15, 17].

In the symbolic model, privacy is usually defined as observational equivalence of two cases where a pair of voters swap their votes [2, 5, 6, 8, 9, 27, 17]. The definitions mainly differ in the way they model voting processes and deal with specifics of protocols. Some of them can be verified automatically using standard tools (e.g. ProVerif [3] and ProSwapper [16]). This swap-based approach was not designed for weighted votes and, as explained above, may lead to unexpected results in this case.

The other main approach roots in the computational model. In this case the real-world protocol is compared to an ideal situation and the attacker's advantage is analyzed [23, 14, 15]. Our symbolic definition

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

(a) Plain process

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

(b) Extended process

Figure 1: Grammars for *plain* and *extended* or *active processes*

is somewhat related to this computational approach, as we also consider some information – the result – to be leaked even in an ideal situation, and only forbid further leakage. Another possibility is to consider the overall advantage of the attacker without comparing it to an ideal situation [20]. This advantage is always non-negligible as in certain situations the votes are always revealed, e.g. in the case of an unanimous vote.

A third approach was proposed by Langer et al. [21]. The authors developed verifiability and privacy notions based on (un-)linkability between a voter and his vote. Their definitions have to be instantiated with a concrete formal process and attacker model. To define unlinkability, they rely on indistinguishability of runs where votes are swaped, with the same issues as described above.

Küsters and Truderung [19] were the first to explicitly consider multi-voter coercion. In their abstract model, Single-Voter Coercion and Multi-Voter Coercion turned out to be different in general. Subsequently they proposed a modified definition of Coercion-Resistance that implies both Single- and Multi-Voter Coercion-Resistance. In our model Single- and Multi-Voter Coercion are equivalent under certain assumptions on the protocol, hence we do not need to change the initial definition. Additionally, the conditions allow us to precisely characterize the difference between both notions.

Outline: In the next section, we present the Applied π-Calculus and recall the privacy definitions given by Delaune et al. [5]. In Section 3, we introduce our privacy definition and show under which condition it is equivalent to the existing ones. Then, in Section 4, we define Single- and Multi-Voter Receipt-Freeness, analyze their relationship and prove their equivalence under certain assumptions. In Section 5 we define Single- and Multi-Voter Coercion-Resistance and again prove their equivalence under the same hypotheses, before concluding in Section 6.

2 Preliminaries

In this section we recall the Applied π-Calculus, introduce our model of voting protocols and present existing privacy definitions.

2.1 Applied π-Calculus

The Applied π-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 $\texttt{enc}(message, key)$, $\texttt{dec}(message, key)$ – 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 which describes the correctness of symmetric encryption is $\texttt{dec}(\texttt{enc}(message, key), key) =_E message$. Processes are constructed using the grammars detailed in Figure 1.

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 active 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 active process with a hole for an active process that is not under replication, a conditional, an input or an output. In the rest of the paper we use the following usual notions of equivalence and bisimilarity based on the semantics given in the original paper [1].

Definition 1 (Equivalence in a Frame [1])**.** *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 ($\approx_s$) [1])**.** *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 we have $(M = N)\phi$ 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 ($\approx_l$) [1])**.** *Labeled bisimilarity is the largest symmetric relation $\mathcal{R}$ on closed active 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.

2.2 Modeling Voting Protocols

We model voting protocols in the Applied π-Calculus as follows.

Definition 4 (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. We also assume the existence of a particular public channel res that is only used to publish the result of the election.*

Note that we have only one process for the voters. This means that different voters will execute the same process, but with different variable values (e.g. the keys, the vote etc.). To reason about privacy, we talk about instances of a voting protocol, which we call *voting processes*.

Definition 5 (Voting Process)**.** *A voting process of a voting protocol $(V, A_1, \ldots, A_m, \tilde{n})$ is a closed process*

$$\nu\tilde{n}'.(V\sigma_{id_1}\sigma_{v_1}| \ldots |V\sigma_{id_n}\sigma_{v_n}|A_1| \ldots |A_l),$$

where $l \leq m$, $\tilde{n}'$ includes the secret channel names $\tilde{n}$, $V\sigma_{id_i}\sigma_{v_i}$ are the processes executed by the voters, 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 and A_j's are the election authorities which are required to be honest.

The restricted channel names model private channels. Note that we only model the honest authorities as unspecified parties are subsumed by the attacker.

2.3 Existing Privacy Definitions

Before we can formally define privacy, we need the following two transformations. The first one turns a process P into another process P^{ch} that reveals all its inputs and secret data on the channel ch.

Definition 6 (Process P^{ch} [5])**.** *Let P be a plain process and ch be a channel name. P^{ch} is defined as follows:*

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

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

Definition 7 (Process P^{c_1,c_2} [5])**.** *Let P be a plain process and c_1, c_2 be channel names. P^{c_1,c_2} is defined as follows:*

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

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

Definition 8 (Process $A^{\setminus out(ch,\cdot)}$ [5])**.** *Let A be an extended process. We define the process $A^{\setminus out(ch,\cdot)}$ as* $\nu ch.(A|!\mathtt{in}(ch, x))$.

We now recall the privacy definitions given by Delaune et al. [5], which are the bases for many other definitions [2, 6, 8, 9, 27, 17, 26]. Their main idea for defining privacy is simple: A protocol respects privacy if any two instances where two voters swap votes are bisimilar.

Definition 9 (Swap-Privacy (SwP) [5])**.** *A protocol satisfies* Swap-Privacy (SwP) *if for any context S corresponding to a voting process with a hole for two voters and for all votes σ_{v_A} and σ_{v_B} we have*

$$S\left[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}\right] \approx_l S\left[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}\right].$$

In the literature S may sometimes contain corrupted or coerced voters (e.g. in [27]), here we will suppose that it contains only honest voters and authorities to be able to clearly distinguish single- and multi-voter coercion.

Defining Receipt-Freeness is a bit more complicated as the voter will execute some counter strategy, i.e. a different process, to fake the receipt, but it can still be expressed as a bisimilarity between two situations. In the first situation, the targeted voter votes a and reveals his secret data. In the second situation, he executes another process – the counter-strategy – which allows him to vote b and fake the secret data in a way that both instances are bisimilar. A protocol is receipt-free if such a process – a counter-strategy – exists.

Definition 10 (Swap-Receipt-Freeness (SwRF) [5])**.** *A protocol satisfies* Swap-Receipt-Freeness (SwRF) *if for any context S corresponding to a voting process with a hole for two voters and for all votes σ_{v_A} and σ_{v_B} there exists a process V' such that $V'^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B}$ and*

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

One could define Coercion-Resistance in the same way, but in that case the attacker could force the targeted voter to vote d and not a in the situation where he complies with the instructions. This would make both situations trivially distinguishable by just looking at the result. To prevent this, Delaune et al. [5] use a context C that is required to force the voter to vote a, but can otherwise interact in any way with the voter.

Definition 11 (Swap-Coercion-Resistance (SwCR) [5])**.** *A protocol satisfies* Swap-Coercion-Resistance (SwCR) *if for any context S corresponding to a voting process with a hole for two voters and for all votes σ_{v_A} and σ_{v_B} there exists a process V' such that for any context C with $C = \nu c_1.\nu c_2.(\_|P)$ and $\tilde{n} \cap fn(C) = \emptyset$, $S\left[C\left[(V\sigma_{id_A}\sigma_{v_A})^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{v_B}\right] \approx_l VP'_A\left[(V\sigma_{id_A}\sigma_{v_A})^{chc}|V\sigma_{id_B}\sigma_{v_B}\right]$ we have $C\left[V'\right]^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_A}\sigma_{v_B}$ and*

$$S\left[C\left[(V\sigma_{id_A}\sigma_{v_A})^{c_1,c_2}\right]|V\sigma_{id_B}\sigma_{v_B}\right] \approx_l S\left[C\left[V'\right]|V\sigma_{id_B}\sigma_{v_A}\right].$$

Delaune et al. [5] showed that any protocol ensuring (SwCR) ensures (SwRF), and any protocol ensuring (SwRF) ensures (SwP).

3 Defining Privacy

Our privacy definition is based on the observation that - as the result of the vote is always published - some knowledge about the voter's choices can always be inferred from the outcome. The classical example is the case of a unanimous vote where the contents of all votes are revealed just by the result. Yet - as already discussed in the introduction - there can also be other cases where some of the votes can be inferred from the result, in particular in the case of weighted votes. If for example Alice holds 66% of the shares and Bob 34%, both votes are always revealed when announcing the result: If one option gets 66% and the other 34%, it is clear which one was chosen by Alice or Bob. However, if we have a different distribution of the shares (e.g. 50%, 25% and 25%), some privacy is still possible as there several situations with the same result. Thus our main idea: If two instances of a protocol give the same result, an attacker should not be able to distinguish them. Note that this includes the classic definition where votes are swapped, if this give the same result.

3.1 Formal Definition

To express this formally, we need to define the result of an election. As defined above, we suppose that the result is always published on a special channel res. The following definition allows us to hide all channels except for a specified channel c, which we can use for example to reason about the result on channel res.

Definition 12 ($P|_c$)**.** *Let $P|_c = \nu\tilde{ch}.P$ where $\tilde{ch}$ are all channels except for c, i.e. we hide all channels except for c.*

Now we can formally define our privacy notion: If two instances of a protocol give the same result, they should be bisimilar.

Definition 13 (Vote-Privacy (VP))**.** *A voting protocol ensures* Vote-Privacy (VP) *if for any two instances* $VP_A = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ *and* $VP_B = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *we have*

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP_A \approx_l VP_B.$$

A simple interpretation of this definition is that everything apart from the result on channel res has to remain private. This obviously relies heavily on the notion of "result" and the modeling of the protocol. Typically the result will only contain only the sum of all votes, which corresponds to a simple and intuitive understanding of privacy.

Some protocols may leak some additional information, for example the number of ballots on the bulletin board. For instance in the protocol by Juels et al. [15] voters can post fake ballots. In this case, the above definition of the result may lead to a too restrictive privacy notion, since two situations with the same votes but a different number of fakes are required to be bisimilar. To address this issue, we can include the number of ballots in the result if we want to accept the additional leakage. This gives very fine-grained control about the level of privacy we want to model.

Note that if the link between a voter and his vote is also published as part of the result on channel res, our definition of privacy may be true although this probably does not correspond to the intuitive understanding of privacy. This is however coherent within the model since everything apart from the result is private; simply the result itself leaks too much information.

3.2 Link to Existing Definitions

To establish the relationship of our definition and the existing ones, we need to formally characterize their difference. Intuitively the swap-based definition assumes that swapping two votes will not change the result. This can be formalized as follows: If two instances of the protocol with the same voters give the same result, then the votes are a permutation of each other, and vice versa. This precludes weighted votes, thus the name "Equality of Votes".

Definition 14 (Equality of Votes (EQ))**.** *A voting protocol respects* Equality of Votes (EQ) *if for any* $VP_A = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A}|\ldots|V\sigma_{id_n}\sigma_{v_n^A}|A_1|\ldots|A_l)$ *and* $VP_B = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B}|\ldots|V\sigma_{id_n}\sigma_{v_n^B}|A_1|\ldots|A_l)$ *we have*

$$VP_A|_{res} \approx_l VP_B|_{res} \Leftrightarrow \exists\pi : \forall i : \sigma_{v_i^B} = \sigma_{v^A_{\pi(i)}},$$

where π *is a permutation.*

This allows us to formally prove that our definition is equivalent to the existing ones if (EQ) holds.

Theorem 1 (Equivalence of Privacy Definitions)**.** *If a protocol respects (EQ), then (VP) and (SwP) are equivalent.*

The full formal proof can be found in Appendix B. Intuitively, because of (EQ), two instances of a protocol can only have the same result if the votes are a permutation of each other. As any permutation can be written as a sequence of simple permutations (swaps), (SwP) is enough to generate any possible permutation, which gives (VP). Conversely, the definition of (SwP) becomes just a particular case of (VP).

It is easy to see that this condition (EQ) is necessary: If a protocol uses weighted votes (e.g. Alice 66%, Bob 34%), it may satisfy (VP), but not (SwP).

Similarly, consider the following example: In the official result announced on channel res, a pre-selected candidate always wins - this could be the case if the authorities are dishonest and want to manipulate the election outcome. If however at the same time the ballots on the bulletin board allow to calculate the result, such a protocol may ensure (SwP) – if the ballots cannot be linked to the voters –, but not (VP) because two instances with a different outcome based on the ballots will have the same "result" on res. Note that such a protocol would contradict (EQ) because we have instances where the votes are not a permutation of each other, but still give the same result.

3.3 Example: A Variant of FOO

Eliasson and Zúquete [11] propose an implementation of a voting system supporting vote weights based on REVS [13], which itself is based on the protocol by Fujioka et al. [12], often referred to as "FOO".

3.3.1 Informal description:

The protocol by Fujioka et al. [12]. is split into 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 A.
- 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.
- C 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.

3.3.2 Adding Vote Weights:

In [11] Eliasson and Zúquete discuss several possibilities on how to implement weights in this protocol:

- including the weight in the vote (which requires trusting the voter for correctness or zero-knowledge proofs to verify the weight)
- using different keys when the vote is signed by the administrator, where each key corresponds to a different weight
- using multiple ballots per voter, i.e. if for example voter A holds 70% and voter B 30% of the shares, voter A sends 7 and voter B 3 ballots.

We implemented the latter variant in the Applied π-Calculus. Using a manual proof (see Appendix H for details) we can show that

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow \sum_{i=1}^{n} v_i^A * w_i = \sum_{i=1}^{n} v_i^B * w_i. \quad (1)$$

Using a python script available on our website [7] that generates all cases to check based on the number of voters and the discrete weight distribution, we can use Proverif to then establish (2) which gives that this variant ensures (VP).

$$\sum_{i=1}^{n} v_i^A * w_i = \sum_{i=1}^{n} v_i^B * w_i \Rightarrow VP_A \approx_l VP_B \quad (2)$$

4 Receipt-Freeness

In this section we define receipt-freeness for weighted votes. We first consider the case where only one voter is attacked, then we define multi-voter attacks.

4.1 Single-Voter Receipt-Freeness (SRF)

We combine the idea by Delaune et al. (Def. 10) with our definition of Privacy: If two instances of a voting protocol give the same result, they should be bisimilar even if one voter reveals his secret data in one case or fakes it in the other.

Definition 15 (Single-Voter Receipt Freeness (SRF))**.** *A voting protocol ensures* Single-Voter Receipt Freeness (SRF) *if for any voting processes* $VP_A = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$, $VP_B = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *and any number* $i \in \{1,\ldots,n\}$ *there exists a process* V_i' *such that we have* $V_i'^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$ *and*

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP'_A\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right] \approx_l VP'_B\left[V_i'\right],$$

where VP'_A *and* VP'_B *are like* VP_A *and* VP_B*, but with holes for the voter* $V\sigma_{id_i}$*.*

As for (VP), our definition is equivalent to the existing one based on swapping if the protocol ensures (EQ), which is the case if it does not use weighted votes. Similarly to swap-based definitions, (SRF) is stronger than (VP). The proof is analogous to the proof in the swap-based model (see Appendices C and D for details).

4.2 Multi-Voter Receipt-Freeness (MRF)

We now generalize the idea of Receipt-Freeness to the case where multiple voters are attacked. Instead of only considering one attacked voter i, we consider a set I of attacked voters. To be receipt-free, it should be possible for all attacked voters to fake the receipt. Note that we assume that there is always at least one honest voter, except for the case with only one voter.

Definition 16 (Multi-Voter Receipt Freeness (MRF))**.** *A voting protocol ensures* Multi-Voter Receipt Freeness (MRF) *if for any voting processes* $VP_A = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$, $VP_B = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *and any subset* $I \subset \{1,\ldots,n\}$, $I \neq \{1,\ldots,n\}$ *if* $n > 1$*, then there exists processes* V_i' *such that we have* $\forall i \in I : V_i'^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$ *and*

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP'_A\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right] \approx_l VP'_B\left[\mid_{i\in I} V_i'\right],$$

where VP'_A *and* VP'_B *are like* VP_A *and* VP_B*, but with holes for all voters* $V\sigma_{id_i}, i \in I$*.*

By choosing $I = \{i\}$ we directly obtain that (MRF) implies (SRF). Under certain conditions the inverse is also true. To prove this, we define a "generalized voting process" which is like a voting process, but some of the voters might be under attack.

Definition 17 (Generalized Voting Process)**.** *A* Generalized Voting Process *is a voting process* VP *with variables for the voter's processes that can either be a "normal" voter or a voter communicating with the intruder, i.e.* $VP = \nu\tilde{n}.(V_1|\ldots|V_n|A_1|\ldots|A_l)$ *where* $V_i \approx_l V\sigma_{id_i}\sigma_{v_i}$ *or* $V_i^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i}$*.*

The next definition captures the key properties required for our proof. It expresses two modularity conditions of a voting protocol.

Definition 18 (Modularity (Mod))**.** *A voting protocol is* modular (Mod) *if it is* composable *and* decomposable*. A voting protocol is* composable *if for any generalized voting processes* VP_A *and* VP_B *there exists a generalized voting process* VP *such that* $VP \approx_l VP_A|VP_B$*. A voting protocol is* decomposable

if any generalized voting process $VP = \nu\tilde{n}.(V_1|\dots|V_n|A_1|\dots|A_l)$ *can be decomposed into processes* $VP_i = \nu\tilde{n}_i.(V_i|A_1^i|\dots|A_l^i)$ *where*

$$VP \approx_l VP_1|\dots|VP_n. \tag{3}$$

Imagine a protocol where in order to escape coercion the voters can claim that a certain ballot on the bulletin board is their ballot, but it was actually prepared by some honest authority to allow the voters to create a fake receipt. If we suppose that this ballot exists only once no matter how many voters are attacked, it would be enough for a single voter to fake his receipt. However we cannot compose two instances with one attacked voter each, as they would use the same fake ballot which would be noticeable for the attacker. Hence the above definition also captures the fact that faking the receipt to escape coercion can be done by each voter independently.

Another property we need for our proof is *Correctness*, i.e. the fact that if in two instances the voters' choices are the same, they give the same result[1].

Definition 19 (Correctness (Cor))**.** *A voting protocol is* correct *if for any generalized voting processes* $VP_A = \nu\tilde{n}_A.(V_{1,A}|\dots|V_{n,A}|A_1|\dots|A_l)$ *and* $VP_B = \nu\tilde{n}_B.(V_{1,B}|\dots|V_{n,B}|A_1|\dots|A_l)$ *with for any* i *and* $X \in \{A, B\}$*:* $V_{i,X}^{\setminus out(ch_{c_i},\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i}$*, we have*

$$VP_A|_{res} \approx_l VP_B|_{res} \tag{4}$$

It is easy to see that Correctness is implied by Equality of Votes as the identity is a permutation, hence any protocol ensuring (EQ) ensures (Cor) (see Appendix E for the proof). Putting everything together, we are able to prove the equivalence of (SRF) and (MRF).

Theorem 2. *If a protocol is modular, correct and ensures Single-Voter Receipt Freeness, it also ensures Multi-Voter Receipt Freeness.*

The full proof is given in Appendix F. The main idea is that we can decompose an instance with multiple attacked voters into instances with at most one attacked voter, where we can apply the single-voter assumption, and recompose the result. Note that the assumptions (Mod), (EQ) are satisfied by many well-known protocols (e.g. [4, 12, 25]), we illustrate this on an example.

Remark. We have to be careful when modeling protocols using a full PKI. If we model the PKI inside the voting process, decomposing a protocol would result in two instances using different keys, which will most probably be visible to an attacker and the bisimilarity (3) will not hold. A possible solution could be to externalize the PKI into a context K such that $K[VP] \approx_l K[VP_1|VP_2]$, which ensures that VP_1 and VP_2 use the same keys. This would allow us to obtain the same result for protocols such as [22, 29].

4.3 Example: Protocol by Okamoto

The protocol by Okamoto [25] uses trapdoor commitments to achieve (SwRF), but it is not (SwCR) [5].

4.3.1 Informal Description:

The protocol is split in 3 phases. In the first phase the voter obtains a signature on a commitment to his vote from the administrator:

- Each voter V_i chooses his vote v_i and computes a trapdoor commitment $x_i = \xi(v_i, k_i, td_i)$ for a random key k_i and a trapdoor td_i.
- V_i blinds the commitment using a blinding function χ, a random value r_i and obtains $e_i = \chi(x_i, r_i)$.
- V_i signs e_i and sends the signature $s_i = \sigma_{V_i}(e_i)$ together with e_i and his identity to the administrator A.

[1] This does not entirely cover intuitive correctness as it will be fulfilled by protocols always giving the same result independently from the votes, but it will fail for a protocol announcing a random result.

- 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 checks the signature, unblinds d_i using δ and obtains $y_i = \delta(d_i, r_i)$.

In the second phase the actual voting takes place:

- V_i sends the signed trapdoor commitment y_i to the collector C through an anonymous channel.

- C checks the administrator's signature.

- V_i sends (v_i, r_i, x_i) to the *timeliness* member T through an untappable anonymous channel.

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

- C publishes the list of correct ballots (x_i, y_i).

- T publishes a randomly shuffled list of votes v_i and a zero-knowledge proof that he knows a permutation π for which $x_{\pi(i)} = \xi(v_i, r_i)$.

4.3.2 Analysis:

The protocol is receipt-free because the trapdoor allows a voter to open the commitment in any way to fake a receipt for any candidate as formally shown by Delaune et al. [5]. Here we use a slightly modified version of their model to show that it also respects (MRF)[2], see Appendix I for details.

It is also easy to see that the protocol ensures (EQ) as votes are not weighted and the honest timeliness members will publish the correct result. We can also find that it is *modular* by analyzing the structure of the voting processes. In the case of n voters, we have the form

$$\nu\text{chT}.(V_1|\dots|V_n \mid_{i=1,\dots,n} \text{processT}),$$

where processT is the process executed by the timeliness member and chT is the private channel between voters and the timeliness member. For $k \in \{1, \dots, n-1\}$, a possible decomposition would be

$$\nu\text{chT}.(V_1|\dots|V_k \mid_{i=1,\dots,k} \text{processT})|\nu\text{chT}.(V_{k+1}|\dots|V_n \mid_{i=k+1,\dots,n} \text{processT}),$$

which is obviously bisimilar. It is easy to see that this also works for composing processes. This is because each instance contains the same private channel and as many processT as voters. Thus the protocol is modular, and using Theorem 2 we have that the protocol by Okamoto ensures (MRF).

Note that this would also hold for a variant of the protocol with weighted votes. Similarly to the first example we could implement this using multiple ballots, and the resulting protocol ensures (SRF), (MRF), (Cor) and (Mod), but neither (EQ) nor (SwRF).

5 Coercion-Resistance

After discussing Receipt-Freeness, we now define Coercion-Resistance. As before, we start with Single-Voter Coercion-Resistance.

[2]Essentially we do not use the key distribution process as no keys are required to be secret. We model them as free variables instead.

5.1 Single-Voter Coercion (SCR)

In this case, we combine (VP) with (SwCR): If two instances of a voting protocol give the same result, they should be bisimilar even if one voter interacts with the attacker in one case or only pretends to do so in the other case. The coercion is modeled by the context C that interacts with the voter and tries to force him to vote for a certain candidate.

Definition 20 (Single-Voter Coercion-Resistance (SCR))**.** *A voting protocol ensures* Single-Voter Coercion-Resistance (SCR) *if for any voting processes* $VP_A = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$, $VP_B = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *and any number* $i \in \{1,\ldots,n\}$ *there exists a process* V_i' *such that for any context* C_i *with* $C_i = \nu c_1.\nu c_2.(\_|P_i)$ *and* $\tilde{n} \cap fn(C) = \emptyset$, $VP'_A\left[C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VP'_A\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]$ *we have* $C_i\left[V_i'\right]^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$ *and*

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP'_A\left[C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VP'_B\left[C_i\left[V_i'\right]\right],$$

where VP'_A *and* VP'_B *are like* VP_A *and* VP_B*, but with a holes for the voter* $V\sigma_{id_i}$.

As above, we can easily link this definition to the existing swap-based definition using (EQ): If a protocol respects (EQ), (SCR) and (SwCR) are equivalent. The proof is similar to the (SRF) case.

5.2 Multi-Voter Coercion (MCR)

We now discuss Multi-Voter Coercion-Resistance. To model the case where multiple voters are attacked, we consider the set I of attacked voters.

Definition 21 (Multi-Voter Coercion-Resistance (MCR))**.** *A voting protocol ensures* Multi-Voter Coercion-Resistance (MCR) *if for any voting processes* $VP_A = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$, $VP_B = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ *and any subset* $I \subset \{1,\ldots,n\}$, $I \neq \{1,\ldots,n\}$ *if* $n > 1$, *there exists processes* V_i' *such that for any contexts* $C_i, i \in I$ *with* $C_i = \nu c_1.\nu c_2.(\_|P_i)$ *and* $\tilde{n} \cap fn(C) = \emptyset$, $VP'_A\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VP'_A\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]$ *we have* $\forall i \in I : C_i\left[V_i'\right]^{\setminus out(chc,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$ *and*

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP'_A\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VP'_B\left[\mid_{i\in I} C_i\left[V_i'\right]\right],$$

where VP'_A *and* VP'_B *are like* VP_A *and* VP_B*, but with holes for all voters* $V\sigma_{id_i}, i \in I$.

As for (MRF), (MCR) implies (SCR), and (MCR) resp. (SCR) is stronger than (MRF) resp. (SRF) (for the proofs, see Appendix G). We also have equivalence between (SCR) and (MCR) under the same assumptions as in the case of Receipt-Freeness using a similar proof.

5.3 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) and a voting booth.

5.3.1 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. 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 dummy or real votes) 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.

5.3.2 Analysis:

The protocol satisfies (SwRF) as the receipt contains only random numbers, and it is impossible for the attacker to know which entry corresponds to the random value generated by the RNG [8]. It also ensures (SwCR) as voting takes places inside a secured voting booth. This was formally proven in the DKR-model [8], and we use the same model to show that it satisfies (MCR).

As before, it is easy to see that the protocol ensures Equality of Votes and hence Correctness as votes are not weighted. By analyzing the structure of the voting process, we can see that the protocol also is *modular*. In the case of n voters, we have the following voting process

$$\begin{array}{c}\nu\text{privChM}_1 \dots \nu\text{privChM}_n.\nu\text{privChRM}_1 \dots \nu\text{privChRM}_n.\\ \nu\text{privChR}_1 \dots \nu\text{privChR}_n.(V_1| \dots |V_n|M_{1,\dots,n;l}|R_1| \dots |R_n)\end{array}$$

where R_i are the trusted random number generators, $M_{1,\dots,n;l}$ is the voting machine process for n voters from 1 to n and l candidates, and privChM$_i$, privChRM$_i$ and privChR$_i$ are the private channels between the voter and the voting machine, the RNG and the voting machine, and the RNG and the voter respectively. For $k \in \{1, \dots, n-1\}$, this can be rewritten as

$$\begin{array}{c}\nu\text{privChM}_1 \dots \nu\text{privChM}_k.\nu\text{privChRM}_1 \dots \nu\text{privChRM}_k.\\ \nu\text{privChR}_1 \dots \nu\text{privChR}_k.(V_1| \dots |V_k|M_{1,\dots,k;l}|R_1| \dots |R_k)|\\ \nu\text{privChM}_{k+1} \dots \nu\text{privChM}_n.\nu\text{privChRM}_{k+1} \dots \nu\text{privChRM}_n.\\ \nu\text{privChR}_{k+1} \dots \nu\text{privChR}_n.(V_{k+1}| \dots |V_n|M_{k+1,\dots,n;l}|R_{k+1}| \dots |R_n)\end{array}$$

as $M_{1,\dots,n;l} \approx_l M_{1,\dots,k;l}|M_{k+1,\dots,n;l}$. This can be easily seen from the applied π-code in Appendix J. It is easy to see that this also works for composing processes. Hence we have all necessary conditions and obtain that Bingo Voting ensures (MCR).

6 Conclusion

We presented an intuitive definition of privacy for voting protocols that generalizes to situations with weighted votes. We extended the definition to include Receipt-Freeness and Coercion-Resistance as well. We considered situations where only one voter is under attack, and others where multiple voters are attacked. We were able to show that - under the assumptions that votes are not weighted and correctly counted - the single voter case is equivalent to (SwP), (SwRF), (SwCR) as defined by Delaune et al. [5]. Moreover, we proved that the multi-voter case is equivalent to the single-voter case if the protocol is correct (Cor) and respects a modularity condition (Mod). This condition allows us to compose and decompose protocols, which expresses the fact the different parts of the protocol are independent. Figure 2 summarizes our results. Finally, we illustrated our work by analyzing two existing protocols. As future work, we would like to translate these symbolic definitions to the computational setting.

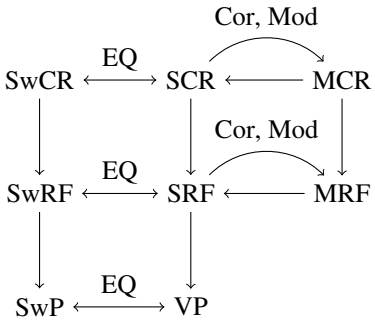


Figure 2: Relations among the notions. A $\xrightarrow{C}$ B means that under the assumption C a protocol ensuring A also ensures B.

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. 1, 2.1, 2.1, 1, 2, 3

[2] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. *Computer Security Foundations Symposium, IEEE*, 0:195–209, 2008. 1, 1, 2.3

[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. 1, 1

[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, pages 111–124. Springer Berlin / Heidelberg, 2007. 4.2, 5.3

[5] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17:435–487, December 2009. 1, 1, 1, 6, 7, 8, 2.3, 9, 10, 2.3, 11, 2.3, 4.3, 4.3.2, 6, 1, 2, C, D

[6] 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 *LNCS*, pages 289–309. Springer, May 2010. 1, 1, 2.3

[7] Jannik Dreier. The code and scripts used to automatically verify the examples is available at `http://www-verimag.imag.fr/~dreier/papers/foo-weighted-code.zip`, 2011. 3.3.2

[8] 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)*, LNCS. Springer, 2011. 1, 1, 2.3, 5.3.2

[9] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. A formal taxonomy of privacy in voting protocols. In *First IEEE International Workshop on Security and Forensics in Communication Systems (ICC'12 WS - SFCS)*, 2012. 1, 1, 2.3

[10] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. On parallel factorization of processes in the applied pi calculus. Technical Report TR-2012-3, Verimag Research Report, March 2012. Available at `http://www-verimag.imag.fr/TR/TR-2012-3.pdf`. 4

[11] Charlott Eliasson and André Zúquete. An electronic voting system supporting vote weights. *Internet Research*, 16(5):507–518, 2006. 1, 1, 3.3, 3.3.2

[12] 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 *LNCS*, pages 244–251. Springer Berlin / Heidelberg, 1992. 1, 3.3, 3.3.1, 4.2

[13] Rui Joaquim, André Zúquete, and Paulo Ferreira. Revs - a robust electronic voting system. In *IADIS International Conference e-Society 2003, Lisboa (Portugal), June 3-6*, 2003. 1, 3.3

[14] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections, 2002. Cryptology ePrint Archive, Report 2002/165, `http://eprint.iacr.org/`. 1, 1

[15] 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. ACM, 2005. 1, 1, 3.1

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

[17] 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 *LNCS*, pages 186–200. Springer, 2005. 1, 1, 2.3

[18] Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *Proceedings of the 15th European Symposium on Research in Computer Security, ESORICS 2010*, volume 6345 of *LNCS*, pages 389–404. Springer, 2010. 1, 1

[19] R. Küsters and T. Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *2009 IEEE Symposium on Security and Privacy (S&P 2009)*, pages 251–266. IEEE Computer Society, 2009. 1, 1

[20] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A game-based definition of coercion-resistance and its applications. In *Proceedings of the 2010 23rd IEEE Computer Security Foundations Symposium*, CSF '10, pages 122–136, Washington, DC, USA, 2010. IEEE Computer Society. 1

[21] Lucie Langer, Hugo Jonker, and Wolter Pieters. Anonymity and verifiability in voting: understanding (un)linkability. In *Proceedings of the 12th international conference on Information and communications security*, ICICS'10, pages 296–310. Springer-Verlag, 2010. 1

[22] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Information Security and Cryptology - ICISC 2003*, volume 2971 of *LNCS*, pages 245–258. Springer Berlin / Heidelberg, 2004. 4.2

[23] 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, 1

[24] Aybek Mukhamedov and Mark D. Ryan. Identity escrow protocol and anonymity analysis in the applied pi-calculus. *ACM Transactions on Information System Security*, 13(41):1–29, 2010. J

[25] Tatsuaki Okamoto. An electronic voting scheme. In *Proceedings of the IFIP World Conference on IT Tools*, pages 21–30, 1996. 4.2, 4.3

[26] 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/. 2.3

[27] Ben Smyth and Veronique Cortier. Attacking and fixing helios: An analysis of ballot secrecy. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium (CSF'11)*, pages 297–311. IEEE, 2011. 1, 1, 2.3, 2.3

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

[29] Roland Wen and Richard Buckland. Masked ballot voting for receipt-free online elections. In *Proceedings of the 2nd International Conference on E-Voting and Identity*, VOTE-ID '09, pages 18–36, Berlin, Heidelberg, 2009. Springer-Verlag. 4.2

A The Applied Pi Calculus

The semantics of the calculus is 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 (i.e. "normal" values).

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 proofs we will use the following two Lemmas:

Lemma 1 ([5]). *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 2 ([5]). *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.*

B Proof of Theorem 1

Proof. "$\Rightarrow$": Suppose the protocol respects (VP). We have to show that it respects (SwP), i.e. that for any voting process S and for all votes σ_{v_A} and σ_{v_B} we have

$$VP_A := S'\,[V\sigma_{id_A}\sigma_{v_A}|V\sigma_{id_B}\sigma_{v_B}] \quad \approx_l \quad S'\,[V\sigma_{id_A}\sigma_{v_B}|V\sigma_{id_B}\sigma_{v_A}] =: VP_B$$

where S' is like S but with a hole for two voters. Obviously the votes on the right hand side are a permutation of the votes of the left hand side and as the protocol ensures Equality of Votes we have

$$VP_A|_{res} \approx_l VP_B|_{res}$$

As the protocol respects Vote-Privacy, we conclude

$$VP_A \approx_l VP_B$$

"$\Leftarrow$": Suppose the protocol respects (SwP) and we want to show that it respects (VP), i.e. for any voting process $VP_A = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} | \ldots | V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$ and $VP_B = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ we have

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP_A \approx_l VP_B$$

Suppose $VP_A|_{res} \approx_l VP_B|_{res}$. As the protocol respects (EQ), we know that there exists a permutation π such that $\forall i : \sigma_{v_i^A} = \sigma_{v^B_{\pi(i)}}$. Using the fact that Swap-Privacy allows us to permute any two votes and that any permutation can be expressed as a sequence of simple permutations, we obtain $VP_A \approx_l VP_B$. □

C Proof of "(SRF) is equivalent to (SwRF) if (EQ) holds"

Proof. "$\Rightarrow$": Suppose a protocol ensuring (SRF). We will now show that the protocol ensures (SwRF), i.e. for any voting process S and for all votes σ_{v_A} and σ_{v_B} there exists a process V_i' such that

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

and

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

where S' is like S but with a hole for two voters. Consider

$$VP_A := S'\left[V\sigma_{id_A}\sigma_{v_A} | V\sigma_{id_B}\sigma_{v_B}\right]$$
$$VP_B := S'\left[V\sigma_{id_A}\sigma_{v_B} | V\sigma_{id_B}\sigma_{v_A}\right]$$

Obviously the votes on both sides are a permutation of each other, thus we we have $VP_A|_{res} \approx_l VP_B|_{res}$ (by (EQ)). We can apply (SRF) to obtain for any i the existence of a process V_i' such that

$$V_i'^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP'_A\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right] \approx_l VP'_B\left[V_i'\right]$$

We choose $i = A$ and obtain the desired property.

"$\Leftarrow$": Suppose a protocol ensuring (SwRF) and we want to show (SRF), i.e. for any voting processes $VP_A = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^A} \mid A_1 \mid \ldots \mid A_l)$, $VP_B = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B} \mid \ldots \mid V\sigma_{id_n}\sigma_{v_n^B} \mid A_1 \mid \ldots \mid A_l)$ and any number $i \in \{1, \ldots, n\}$ there exists a process V_i' such that we have

$$V_i'^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP'_A\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right] \approx_l VP'_B\left[V_i'\right]$$

where VP'_A and VP'_B are like VP_A and VP_B, but with holes for the voter $V\sigma_{id_i}$. As we suppose (EQ), the votes on the left are a permutation of the votes on the right. (SwRF) allows us the swap the vote of the targeted voter i with any other voter's vote. As (SwRF) implies (SwP) (cf. [5]), we can also swap the votes of all other voters if necessary, which gives the desired result. □

D Proof of "(SRF) implies (VP)"

Proof. The proof is similar to the proof of (SwRF) implies (SwP) in the DKR-model [5]:
By hypothesis there is a closed plain process so that

$$V_i'^{\backslash out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP'_A\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right] \approx_l VP'_B\left[V_i'\right]$$

where VP'_A and VP'_B are like VP_A and VP_B, but with holes for the voter $V\sigma_{id_i}$. We suppose $VP_A|_{res} \approx_l VP_B|_{res}$. Then we can apply the context $\nu chc_i.(\ \_\ |\ !\mathtt{in}(chc_i, x))$ on both sides, which gives

$$VP'_A\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]^{\backslash out(chc_i,\cdot)} \approx_l VP'_B\left[V_i'\right]^{\backslash out(chc_i,\cdot)}$$

By using Lemma 2 we obtain

$$VP'_A\left[(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]^{\backslash out(chc_i,\cdot)} \equiv VP'_A\left[\left(V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right)^{\backslash out(chc_i,\cdot)}\right]$$

and

$$VP'_B\left[V_i'\right]^{\backslash out(chc_i,\cdot)} \equiv VP'_B\left[V_i'^{\backslash out(chc_i,\cdot)}\right]$$

We can now apply Lemma 1, (D) and use the fact that labeled bisimilarity is closed under structural equivalence and conclude

$$VP_A = VP'_A\left[V\sigma_{id_i}\sigma_{v_i^A}\right] \approx_l VP'_B\left[V\sigma_{id_i}\sigma_{v_i^B}\right] = VP_B$$

□

E Proof of "(EQ) implies (Cor)"

Lemma 3. *If a protocol respects Equality of Votes, it also respects Correctness.*

Proof. We have to show that for any generalized voting processes $VP_A = \nu\tilde{n}_A.(V_{1,A}|\ldots|V_{n,A}|A_1|\ldots|A_l)$ and $VP_B = \nu\tilde{n}_B.(V_{1,B}|\ldots|V_{n,B}|A_1|\ldots|A_l)$ with for any i and $X \in \{A, B\}$: $V_{i,X}^{\backslash out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i}$, we have

$$VP_A|_{res} \approx_l VP_B|_{res} \tag{5}$$

To show this, we will rewrite VP_A using Lemma 2:

$$\begin{aligned}
&VP_A^{\backslash out(chc_1,\cdot)\ldots\backslash out(chc_n,\cdot)}\\
&= (\nu\tilde{n}_A.(V_{1,A}|\ldots|V_{n,A}|A_1|\ldots|A_l))^{\backslash out(chc_1,\cdot)\ldots\backslash out(chc_n,\cdot)}\\
&\equiv (\nu\tilde{n}_A.(V_{1,A}^{\backslash out(chc_1,\cdot)}|\ldots|V_{n,A}^{\backslash out(chc_n,\cdot)}|A_1|\ldots|A_l))\\
&\approx_l (\nu\tilde{n}_A.(V\sigma_{id_1}\sigma_{v_1}|\ldots|V\sigma_{id_n}\sigma_{v_n}|A_1|\ldots|A_l))
\end{aligned} \tag{6}$$

We apply the same operations on VP_B and obtain

$$VP_B^{\backslash out(chc_1,\cdot)\ldots\backslash out(chc_n,\cdot)} \approx_l \nu\tilde{n}_B.(V\sigma_{id_1}\sigma_{v_1}|\ldots|V\sigma_{id_n}\sigma_{v_n}|A_1|\ldots|A_l) \tag{7}$$

Hence we have two voting processes where the votes are a permutation of each other and obtain the result $VP_A|_{res} \approx_l VP_B|_{res}$ directly using (EQ). □

F Proof of Theorem 2

For this proof we need the following lemma.

Lemma 4 ([10]). *For any finite processes (i.e. no "!") A, B, C and D with $C \approx_l D$ we have*

$$A|C \approx_l B|D \Rightarrow A \approx_l B \tag{8}$$

Note that there are no private channels between A and C or B and D respectively.

Proof. We want to prove that the protocol ensures Multi-Voter Receipt Freeness, i.e. that for any $VP_A = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^A}|\dots|V\sigma_{id_n}\sigma_{v_n^A}|A_1|\dots|A_l)$, $VP_B = \nu\tilde{n}.(V\sigma_{id_1}\sigma_{v_1^B}|\dots|V\sigma_{id_n}\sigma_{v_n^B}|A_1|\dots|A_l)$ and any subset $I \subset \{1,\dots,n\}$ there exists processes V_i' such that we have

$$\forall i \in I : {V_i'}^{\setminus out(chc_i,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B} \tag{9}$$

and

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP'_A\left[\,|_{i\in I} V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right] \approx_l VP'_B\left[\,|_{i\in I} V_i'\right] \tag{10}$$

where VP'_A and VP'_B are like VP_A and VP_B, but with holes for all voters $V\sigma_{id_i}, i \in I$.

We suppose $VP_A|_{res} \approx_l VP_B|_{res}$ and we will show that there exists processes V_i' such that we have (9) and

$$VP'_A\left[\,|_{i\in I} V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right] \,|_{i\le n-1}\, VP_A \approx_l VP'_B\left[\,|_{i\in I} V_i'\right] \,|_{i\le n-1}\, VP_B \tag{11}$$

which gives the desired result using Lemma 4 and Vote-Privacy. The idea is that adding other voting processes allows us to decompose, mix and compose the processes so that we can apply Single-Voter Receipt Freeness and Vote-Privacy on the individual instances.

We start by decomposing the left side of the bisimilarity and recompose the processes to have $|I|$ instances with single-voter coercion and $n - |I|$ simple instances:

$$\begin{aligned} VP'_A \quad & \left[\,|_{i\in I} V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right] \,|_{i\le n-1}\, VP_A \\ \approx_l \quad & |_{i\in I}\, VP'_{A,i}\left[V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right] \,|_{i\notin I}\, VP'_{A,i}\left[V\sigma_{id_i}\sigma_{v_i^A}\right] \\ & |_{i\le n-1}\left(\,|_{j\le n}\, VP'_{A,i,j}\left[V\sigma_{id_i}\sigma_{v_i^A}\right]\right) \\ \approx_l \quad & |_{i\in I}\, VP''_{A,i}\left[V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right] \,|_{i\notin I}\, VP'''_{A,i} \end{aligned} \tag{12}$$

In the first step we break the instances down to a single voter each, i.e. $VP'_{A,i}$ and $VP'_{A,i,j}$ are voting processes with a hole for the one and only voter. In the next step, we assemble the voters and instances in such a way that we obtain complete instances (i.e. containing voters 1 to n) with at most one attacked voter per instance.

We apply the same transformations on the right side and obtain:

$$\begin{aligned} VP'_B \quad & \left[\,|_{i\in I} V_i'\right] \,|_{i\le n-1}\, VP_B \\ \approx_l \quad & |_{i\in I}\, VP'_{B,i}\,[V_i'] \,|_{i\notin I}\, VP'_{B,i}\left[V\sigma_{id_i}\sigma_{v_i^B}\right] \,|_{i\le n-1}\left(\,|_{j\le n}\, VP'_{B,i,j}\left[V\sigma_{id_i}\sigma_{v_i^B}\right]\right) \\ \approx_l \quad & |_{i\in I}\, VP''_{B,i}\,[V_i'] \,|_{i\notin I}\, VP'''_{B,i} \end{aligned} \tag{13}$$

Using Correctness (Definition 19), we have for all i

$$VP''_{A,i}\left[V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right]|_{res} = VP''_{B,i}\,[V_i']\,|_{res} \tag{14}$$

and

$$VP'''_{A,i}|_{res} = VP'''_{B,i}|_{res} \tag{15}$$

Now we can apply Single-Voter Receipt Freeness on the left side, i.e. we have for any $i \in \{1, \dots, n\}$ there exists a process V_i' such that

$$V_i'^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B} \tag{16}$$

and

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP'_A\left[V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right] \approx_l VP'_B\left[V_i'\right] \tag{17}$$

Using this and Vote-Privacy we can rewrite (12) as follows

$$\begin{aligned} &\mid_{i\in I} VP''_{A,i}\left[V\sigma_{id_i}\sigma_{v_i^A}^{chc_i}\right] \mid_{i\notin I} VP'''_{A,i} \\ \approx_l &\mid_{i\in I} VP''_{B,i}\left[V_i'\right] \mid_{i\notin I} VP'''_{A,i} \\ \approx_l &\mid_{i\in I} VP''_{B,i}\left[V_i'\right] \mid_{i\notin I} VP'''_{B,i} \end{aligned} \tag{18}$$

which gives (11) using (13). □

G Proof of "(MCR) implies (MRF)" and "(SCR) implies (SRF)"

Proof. We will only consider the Multi-Voter case, the Single-Voter case follows directly.

Let C_i be evaluation contexts such that $C_i = \nu c_1.\nu c_2.(\_|P_i)$ for some plain processes P_i which fulfill

$$VP'_A\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VP'_A\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]$$

Note that these C_i can be constructed directly from the vote process V. By hypothesis we know that there are closed plain process V_i' such that

$$\forall i \in I : C_i\left[V_i'\right]^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP'_A\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VP'_B\left[\mid_{i\in I} C_i\left[V_i'\right]\right]$$

where VP'_A and VP'_B are like VP_A and VP_B, but with holes for all voters $V\sigma_{id_i}, i \in I$. We have to find other processes V_i'' such that

$$\forall i \in I : V_i'^{\backslash out(chc,\cdot)} \approx_l V\sigma_{id_i}\sigma_{v_i^B}$$

and

$$VP_A|_{res} \approx_l VP_B|_{res} \Rightarrow VP'_A\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right] \approx_l VP'_B\left[\mid_{i\in I} V_i'\right]$$

Let $V_i'' = C_i[V_i']$. This directly fulfills the first requirement. For the second equation, we suppose $VP_A|_{res} \approx_l VP_B|_{res}$. We can then use the condition on C_i and obtain

$$VP'_A\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VP'_A\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right]$$

The second hypothesis gives

$$VP'_A\left[\mid_{i\in I} C_i\left[(V\sigma_{id_i}\sigma_{v_i^A})^{c_1,c_2}\right]\right] \approx_l VP'_B\left[\mid_{i\in I} C_i\left[V_i'\right]\right]$$

As labeled bisimilarity is transitive, we can conclude

$$VP'_A\left[\mid_{i\in I}(V\sigma_{id_i}\sigma_{v_i^A})^{chc_i}\right] \approx_l VP'_B\left[\mid_{i\in I} C_i\left[V_i'\right]\right]$$

which gives us the desired result for $V_i'' = C_i[V_i']$. □

```
processA =
  in(ch1, m1).
  let (pubkeyv, sig) = m1 in
  if pubkeyv = pubkv then
  out(ch2, sign(checksign(sig, pubkv), ska))
```

Listing 1: The administrator process

```
processC =
  synch 1.
  in(ch3, (m3, m4)).
  if checksign(m4, pka) = m3 then
  ν l.out(ch4, (l, m3, m4)).
  in(ch5, (=l, rand)).
  let voteV = open(m3, rand) in
  out(res, voteV)
```

Listing 2: The collector process

H FOO with weighted votes

We use the following model, based on 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 voting process The voter's process (Process 3) receives his private and the administrator's public key; then he votes following the protocol described informally in Section 3.3.1.

The administrator The administrator (Process 1) receives his private key and the public key of a legitimate voter. When receives the blinded commitment, he checks the signature, signs, and sends the result back.

The collector The collector (Process 2) receives the administrator's public key, which he then uses to verify the signature on incoming commitments. If the signature is correct, he creates a new bounded name l (the number in the list) and sends it together with the signed commitment back to the voter. The voter then reveals his randomness, which the collector uses to open the commitment.

Proof of (1). To show that (1) holds, we use a proof by contradiction. Suppose

$$\sum_{i=1}^{n} v_i^A * w_i \neq \sum_{i=1}^{n} v_i^B * w_i \tag{19}$$

We will show that this implies $VP_A \not\approx_l VP_B$ by showing that there exists an execution which allows the attacker to see for all i w_i times a message v_i^X on channel res. Hence he can compare messages and distinguish VP_A and VP_B.

Consider two processes

$$\begin{aligned} VP_A &= \mathop{|}_{1\leq i\leq n} (\mathtt{processV}\left\{{}^{v_i^A}/_{v}\right\}\left\{{}^{w_i}/_{w}\right\}) \mathop{|}_{1\leq i\leq n} (\mathop{|}_{1\leq k\leq w_i} \mathtt{processC}) \\ VP_B &= \mathop{|}_{1\leq i\leq n} (\mathtt{processV}\left\{{}^{v_i^B}/_{v}\right\}\left\{{}^{w_i}/_{w}\right\}) \mathop{|}_{1\leq i\leq n} (\mathop{|}_{1\leq k\leq w_i} \mathtt{processC}) \end{aligned} \tag{20}$$

```
processV =
  for i = 1 to w do
    ν blinder_i . ν r_i .
    let commitedvote_i = commit(v, r_i) in
    let blindedvote_i = blind(commitedvote_i, blinder_i) in
    out(ch1, (pk(skv), sign(blindedvote_i, skv))).
  endfor.
  for i = 1 to w do
    in(ch2, m_i).
    let result_i = checksign(m_i, pubka) in
    if result_i = blindedvote_i then
    let signedvote_i = unblind(m_i, blinder_i) in
  endfor.
  sync 1.
  for i = 1 to w do
    out(ch3, (commitedvote_i, signedvote_i)).
    in(ch4, (l, =commitedvote_i, =signedvote_i)).
  endfor.
  sync 2.
  for i = 1 to w do
    out(ch5, (l_i, r_i)).
  endfor
```

Listing 3: The voting process

In both cases, `processC` cannot synchronize and is blocked. The voter processes will output w_i times $(\mathtt{pk}(\mathtt{skv}), \mathtt{sign}(\mathtt{blindedvote_i}, \mathtt{skv}))$ (their blinded and signed vote) on channel `ch1`. The intruder can then reply with the correctly signed votes, which allows the voters to successfully verify and unblind the signed ballots. Then they will synchronize and output the $(\mathtt{commitedvote_i}, \mathtt{signedvote_i})$ on channel `ch3`. The `processC` processes can then receive the messages, verify the signatures and output $(\mathtt{l}, \mathtt{m3}, \mathtt{m4})$, i.e. $(\mathtt{l}, \mathtt{commitedvote_i}, \mathtt{signedvote_i})$ on channel `ch4`. The voters will then verify if the messages correspond to their votes, synchronize and reveal the random values of the commitments. The collectors can then open the commitments and will publish all votes on channel `res`. Hence there is an execution that reveals all votes and their weights in clear to the intruder, hence using the hypothesis on the distinctiveness of the result, $VP_A \not\approx_l VP_B$. □

I Protocol by Okamoto

We use the following equational theory:

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

The voter The voter process (Process 4) receives the necessary keys and follows the nearly the same protocol as in the case of FOO, but he has to reveal the data necessary to open the commitment over a private channel to the timeliness member T.

The administrator The administrator is exactly the same as given in Section 3.3.1 (Process 1).

```
processV =
  ν blinder . ν r . ν td .
  let commitedvote = tdcommit(v, r, td) in
  let blindedvote = blind(commitedvote, blinder) in
  out(ch1, (pk(skv), sign(blindedvote, skv))).
  in(ch2, m2).
  let result = checksign(m2, pka) in
  if result = blindedvote then
  let signedvote = unblind(m2, blinder) in
  sync 1.
  out(ch3, (commitedvote, signedvote)).
  out(chT, (v, r, commitedvote))
```

Listing 4: The voting process

```
processA =
  in(ch1, m1).
  let (pubkeyv, sig) = m1 in
  if pubkeyv = pubkv then
  out(ch2, sign(checksign(sig, pubkv), ska))
```

Listing 5: The administrator process

```
processT =
  sync 1.
  (* receiving the commitment *)
  in(chT, (vt, rt, xt)).
  sync 2.
  if open(xt, rt) = vt then out(res, vt)
```

Listing 6: The timeliness process

The timeliness member The timeliness process (Process 6) receives the vote, the corresponding commitment and the randomness over a private channel. He verifies the correctness of the data and then publishes the vote.

Lemma 5. *The protocol by Okamoto ensures Equality of Votes.*

Proof. Suppose we have two voting processes VP_A and VP_B and there exists a permutation $\pi : \forall i : \sigma_{v_i^B} = \sigma_{v^A_{\pi(i)}}$. We have to show that $VP_A|_{res} \approx_l VP_B|_{res}$. As the protocol ensures Swap-Privacy and we can write any permutation as a sequence of simple permutations, we have $VP_A \approx_l VP_B$, which gives $VP_A|_{res} \approx_l VP_B|_{res}$.

Suppose we have two voting processes VP_A and VP_B with $VP_A|_{res} \approx_l VP_B|_{res}$. Assume that there exists no permutation π such that $\forall i : \sigma_{v_i^B} = \sigma_{v^A_{\pi(i)}}$, hence there exists a vote v such that the number v_i^A with $v_i^A = v$ is different of the number of v_i^B with $v_i^B = v$, i.e.

$$\sum_{1\leq i\leq n, v_i^A = v} 1 \neq \sum_{1\leq i\leq n, v_i^B = v} 1 \tag{21}$$

We show that this allows an attacker to distinguish $VP_A|_{res}$ and $VP_B|_{res}$ which contradicts $VP_A|_{res} \approx_l VP_B|_{res}$. Consider the following voting processes:

$$\begin{aligned} VP_A &= \nu \text{chT}.\left(\mid_{1\leq i\leq n} \left((\texttt{processV}\left\{ {}^{v_i^A}/_{v} \right\} \sigma_{id_i}) \mid \texttt{processT} \right) \right) \\ VP_B &= \nu \text{chT}.\left(\mid_{1\leq i\leq n} \left((\texttt{processV}\left\{ {}^{v_i^B}/_{v} \right\} \sigma_{id_i}) \mid \texttt{processT} \right) \right) \end{aligned} \tag{22}$$

In both cases, `processT` cannot synchronize and is blocked. The voter processes will output $(\texttt{pk}(\texttt{skv}), \texttt{sign}(\texttt{blindedvote}, \texttt{skv})))$ (their blinded and signed vote) on channel `ch1`. The intruder can then reply with the correctly signed votes, which allows the voters to successfully verify and unblind the signed ballots. Then they will synchronize and output the $(\texttt{commitedvote}, \texttt{signedvote})$ on channel `ch3`, and output $(\texttt{v}, \texttt{r}, \texttt{commitedvote})$ on the private channel `chT` with the timeliness member. The `processT` processes can then receive the messages, synchronize, verify the signatures and output the votes `vt` on channel `res`. Hence there is an execution that reveals all votes in clear to the intruder, thus he can count the number of occurrences of each choice which gives $VP_A \not\approx_l VP_B$. □

Lemma 6. *The protocol by Okamoto ensures Correctness.*

Proof. By Lemmas 3 and 5. □

Lemma 7. *The protocol by Okamoto ensures Modularity.*

Proof. We analyze the structure of the voting processes. In the case of n voters, we have the form

$$\nu \text{chT}.(V_1 | \ldots | V_n \mid_{i=1,\ldots,n} \text{processT}), \tag{23}$$

where processT is the process executed by the timeliness member and chT is the private channel between voters and the timeliness member. For $k \in \{1, \ldots, n-1\}$, a possible decomposition would be

$$\nu \text{chT}.(V_1 | \ldots | V_k \mid_{i=1,\ldots,k} \text{processT}) | \nu \text{chT}.(V_{k+1} | \ldots | V_n \mid_{i=k+1,\ldots,n} \text{processT}), \tag{24}$$

which would be syntactically equivalent except for the private channel `chT`, which is now decomposed into two channels. This affects only one transition, namely the communication between the voter[3] and the timeliness member when the voter reveals his random value. Consider a communication between a V_i and

[3]Here we consider generalized voting processes, however due to the condition on the V_i's we know that there is exactly one message on this channel.

```
processRNG =
  (* generate random number *)
  ν r.
  (* output to voting machine *)
  out(privChM, r).
  (* output to voter *)
  out(privChV, r)
```

Listing 7: The random number generator (RNG)

```
processV =
  (* voting *)
  out(privChM, v).
  (* receipt *)
  for (i = 1 to l)
  in(privChM, receipt_i)
  (* random value, to verify receipt *)
  in(privChRNG, r)
```

Listing 8: The voting process

one `processT` on the left side: This can obviously be matched by the same transition on the right, and vice versa (as the `processT` are all the same, it is not important which one of them is chosen).

Hence we can decompose any process into two processes, and by applying this recursively, into n processes as in the definition. Similarly we can apply the equation the other way round and obtain composability. □

J Bingo Voting

We use the following equational theory:

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

To model the voting booth, we use private channels between the voting machine and the voter, the voter and the RNG, and between the RNG and the voting machine. To achieve better readability we use the macros `for` and `parfor` (similar to [24]) where e.g. `for (i = 1 to 2) out(ch, i)` corresponds to `out(ch, 1).out(ch, 2)` and `parfor (i = 1 to 2) out(ch, i)` corresponds to `(out(ch, 1) | out(ch, 2))`. Additionally we use a function `choose` where

$$\texttt{choose}(p_i, p_j, x, y) = \begin{cases} x \text{ if } i = j \\ y \text{ else} \end{cases}$$

Our model depends on two parameters: k (the number of voters) and l (the number of candidates).

The RNG process The RNG (Process 7) generates a random number and sends it to the voting machine and the voter over private channels.

The voter process The voter (Process 8) sends his vote to the voting machine, receives the random number from the RNG and the receipt.

```
processM' = in(privChV, v).
  in(privChR, r).
  for (j = 1 to l)
    let receipt_j = choose_l(v, r, n_{i,j}) in
    out(privChV, receipt_j);
  synch1.
  (* output result *)
  out(res, v)
  (* output receipts *)
  for (j = 1 to l)
    out(rec, receipt_j)
  (* output unused dummy votes *)
  parfor (j = 1 to l)
    if v_i ≠ p_j then out(dum, ((n_{i,j}, p_j),
      commit((n_{i,j}, p_j), r_{i,j}), r_{i,j}))

processM =
  (* prepare dummy votes *)
  for (i = 1 to k)
    for (j = 1 to l)
      νn_{i,j}.νr_{i,j}.
  parfor (i = 1 to k)
    parfor (j = 1 to l)
      out(ch, commit((n_{i,j}, p_j), r_{i,j}))
    (* voting *)
    let privChV = privChM_i in
    let privChR = privChRM_i in processM'
```

Listing 9: The voting machine

The voting machine process The voting machine (Process 9) generates the dummy votes and publishes the corresponding commitments. Then k subprocesses interact with the voters, i.e. create the receipts. After all voting has been done, they publish the result, the receipts and the unused dummies in random order.

Lemma 8. *Bingo Voting ensures Equality of Votes.*

Proof. Suppose we have two voting processes VP_A and VP_B and there exists a permutation $\pi : \forall i : \sigma_{v_i^B} = \sigma_{v^A_{\pi(i)}}$. We have to show that $VP_A|_{res} \approx_l VP_B|_{res}$. As the protocol ensures Swap-Privacy and we can write any permutation as a sequence of simple permutations, we have $VP_A \approx_l VP_B$, which gives $VP_A|_{res} \approx_l VP_B|_{res}$.

Suppose we have two voting processes VP_A and VP_B with $VP_A|_{res} \approx_l VP_B|_{res}$. Assume that there exists no permutation π such that $\forall i : \sigma_{v_i^B} = \sigma_{v^A_{\pi(i)}}$, hence there exists a vote v such that the number v_i^A with $v_i^A = v$ is different of the number of v_i^B with $v_i^B = v$, i.e.

$$\sum_{1\leq i\leq n, v_i^A = v} 1 \neq \sum_{1\leq i\leq n, v_i^B = v} 1 \tag{25}$$

We show that this allows an attacker to distinguish $VP_A|_{res}$ and $VP_B|_{res}$ which contradicts $VP_A|_{res} \approx_l VP_B|_{res}$. Consider the following voting processes:

$$\begin{aligned} VP_A = \nu\text{privChM}_1 \ldots \nu\text{privChM}_n.\nu\text{privChRM}_1 \ldots \nu\text{privChRM}_n.\\ \nu\text{privChR}_1 \ldots \nu\text{privChR}_n.\Big(\texttt{processM}\,\{^n/_k\}\\ \Big|_{1\leq i\leq n}\big(\texttt{processV}\,\{^{v_i^A}/_v\}\,\sigma_{id_i}\,\{^{\text{privChM}_i}/_{\text{privChM}}\}\,\{^{\text{privChR}_i}/_{\text{privChRNG}}\}\big)\\ \Big|_{1\leq i\leq n}(\texttt{processRNG}\,\{^{\text{privChRM}_i}/_{\text{privChM}}\}\,\{^{\text{privChR}_i}/_{\text{privChV}}\})\Big) \end{aligned}$$

and

$$\begin{aligned} VP_B = \nu\text{privChM}_1 \ldots \nu\text{privChM}_n.\nu\text{privChRM}_1 \ldots \nu\text{privChRM}_n.\\ \nu\text{privChR}_1 \ldots \nu\text{privChR}_n.\Big(\texttt{processM}\,\{^n/_k\}\\ \Big|_{1\leq i\leq n}\big(\texttt{processV}\,\{^{v_i^B}/_v\}\,\sigma_{id_i}\,\{^{\text{privChM}_i}/_{\text{privChM}}\}\,\{^{\text{privChR}_i}/_{\text{privChRNG}}\}\big)\\ \Big|_{1\leq i\leq n}(\texttt{processRNG}\,\{^{\text{privChRM}_i}/_{\text{privChM}}\}\,\{^{\text{privChR}_i}/_{\text{privChV}}\})\Big) \end{aligned}$$

In both cases the voting machine `processM` will generate the commitments $\texttt{commit}((\texttt{n}_{\texttt{i,j}}, \texttt{p}_\texttt{j}), \texttt{r}_{\texttt{i,j}})$ and publish them on channel `ch`. The random generators `processRNG` will generate a fresh nonce and output it on the private channels to the voter and the voting machine. The voter processes will output his vote `v` on channel $\texttt{privChM}_i$. The voting machine will receive both and output the receipts $\texttt{receipt}_j$ to the voters on channel $\texttt{PrivChM}_i$. After all votes have been received and the receipts have been send, the voting machine can synchronize and output the result, i.e. the votes `v` in clear on channel `res`. Hence there is an execution that reveals all votes in clear to the intruder, thus he can count the number of occurrences of each choice which gives $VP_A \not\approx_l VP_B$. □

Lemma 9. *Bingo Voting ensures Correctness.*

Proof. By Lemmas 3 and 8. □

Lemma 10. *Bingo Voting ensures Modularity.*

Proof. By analyzing the structure of the voting process, we can see that the protocol also is *modular*. In the case of n voters, we have the following voting process

$$\nu\text{privChM}_1 \dots \nu\text{privChM}_n.\nu\text{privChRM}_1 \dots \nu\text{privChRM}_n.$$
$$\nu\text{privChR}_1 \dots \nu\text{privChR}_n.(V_1| \dots |V_n|M_{1,\dots,n;l}|R_1| \dots |R_n)$$

where R_i are the trusted random number generators, $M_{1,\dots,n;l}$ is the voting machine process for n voters from 1 to n and l candidates, and privChM_i, privChRM_i and privChR_i are the private channels between the voter and the voting machine, the RNG and the voting machine, and the RNG and the voter respectively. For $k \in \{1, \dots, n-1\}$, this can be rewritten using Lemma 2 as

$$\nu\text{privChM}_1 \dots \nu\text{privChM}_k.\nu\text{privChRM}_1 \dots \nu\text{privChRM}_k.$$
$$\nu\text{privChR}_1 \dots \nu\text{privChR}_k.(V_1| \dots |V_k|M_{1,\dots,k;l}|R_1| \dots |R_k)|$$
$$\nu\text{privChM}_{k+1} \dots \nu\text{privChM}_n.\nu\text{privChRM}_{k+1} \dots \nu\text{privChRM}_n.$$
$$\nu\text{privChR}_{k+1} \dots \nu\text{privChR}_n.(V_{k+1}| \dots |V_n|M_{k+1,\dots,n;l}|R_{k+1}| \dots |R_n)$$

as $M_{1,\dots,n;l} \approx_l M_{1,\dots,k;l}|M_{k+1,\dots,n;l}$. This can be easily seen from the applied pi code (Process 9).

Hence we can decompose any process into two processes, and by applying this recursively, into n processes as in the definition. Similarly we can apply the equation the other way round and obtain composability. □