Voting: You Can't Have Privacy without Individual Verifiability

Véronique Cortier
CNRS, Loria
Nancy, France
veronique.cortier@loria.fr

Joseph Lallemand
Inria, Loria
Nancy, France
joseph.lallemand@loria.fr

ABSTRACT

Electronic voting typically aims at two main security goals: vote privacy and verifiability. These two goals are often seen as antagonistic and some national agencies even impose a hierarchy between them: first privacy, and then verifiability as an additional feature. Verifiability typically includes individual verifiability (a voter can check that her ballot is counted); universal verifiability (anyone can check that the result corresponds to the published ballots); and eligibility verifiability (only legitimate voters may vote).

We show that actually, privacy implies individual verifiability. In other words, systems without individual verifiability cannot achieve privacy (under the same trust assumptions). To demonstrate the generality of our result, we show this implication in two different settings, namely cryptographic and symbolic models, for standard notions of privacy and individual verifiability. Our findings also highlight limitations in existing privacy definitions in cryptographic settings.

1 INTRODUCTION

Electronic voting is often seen as a convenient way for running elections as it allows voters to vote from any place. Moreover, it eases the tally and it can therefore often be used for non trivial counting procedures such as Single Transferable Vote or Condorcet. Numerous voting systems have been proposed so far, like Helios [4], Belenios [15], Civitas [14], Prêt-à-voter [30], or the protocols deployed in Estonia [24] or in Australia [11] to cite a few. On the other hand, many weaknesses or even attacks have been unveiled [31, 32], from voting machines [23] to Internet voting [33].

In order to carefully analyse voting systems, security requirements have been defined. The two main security properties are:

- privacy: no one should know how I voted;
- verifiability is typically described through the three following sub-properties.
  - *individual verifiability:* a voter can check that her ballot is counted;
  - *universal verifiability:* anyone can check that the results corresponds to the published ballots;
  - *eligibility verifiability:* only legitimate voters may vote.

These two main properties seem antagonistic and an impossibility result has even been established between verifiability and unconditional privacy [13], that is, a notion of privacy that is independent of the power of the attacker.

The main contribution of this paper is to establish that, in fact, (computational) *privacy implies individual verifiability*, that is, guarantees that all the honest votes will be counted. This result holds for arbitrary primitives and voting protocols without anonymous channels. To show that this implication is not due to a choice of a very particular definition, we prove this implication in two very distinct contexts, namely symbolic and cryptographic models. In symbolic models, messages are represented by terms and the attacker's behaviour is typically axiomatised through a set of logical formulas or rewrite rules. Cryptographic models are more precise. They represent messages as bitstrings and consider attackers that can be any probabilistic polynomial time Turing machines. Proofs of security are made by reduction to well accepted security assumptions such as hardness of factorisation or discrete logarithm. In both models, we consider a standard notion of privacy, already used to analyse several protocols. In both cases, we establish that privacy implies individual verifiability for a (standard) basic notion of individual verifiability, namely that the result of the election must contain the votes of all honest voters.

We now describe the main idea of the result. Actually, we show the contrapositive implication: if there is an attack against individual verifiability, then there is an attack against privacy. To explain the idea, let's consider a very simple protocol, not at all verifiable. In this simple protocol, voters simply encrypt their votes with the public key of the election. The ballot box stores the ballots and, at the end of the election, it provides the list of recorded ballots to the talliers, who detain the private key, possibly split in shares. The talliers compute and publish the result of the election. The ballot box is not public and no proof of correct decryption is provided so voters have no control over the correctness of the result. Such a system is of course not satisfactory but it is often viewed as a "basic" system that can be used in contexts where only privacy is a concern. Indeed, it is typically believed that such a system guarantees privacy provided that the attacker does not have access to the private key of the election. In particular, the ballot box (that is, the voting server) seems powerless. This is actually *not* the case. If the ballot box aims at knowing how a particular voter, say Alice, voted, he may simply keep Alice's ballot in the list of recorded ballots and then replace all the other ballots by encryptions of valid votes of his choice, possibly following a plausible distribution, to make the attack undetected. When the result of the election is published, the ballot box will know all the votes but Alice's vote, and will therefore be able to deduce how Alice voted.

One may argue that such an attack is not realistic: the ballot box needs to be able to change *all* ballots but one. Note however that elections are often split in many small voting stations (sometimes as small as 20 voters in total [18]). Therefore changing a few ballots can be sufficient to learn how Alice voted. Maybe more importantly, this attack highlights the fact that it is not possible to require privacy without verifiability as sometimes specified by national agencies. For example, in France, only privacy is required [1]. In Switzerland, privacy is a pre-requisite and the level of verifiability depend on the percentage of voters that can vote electronically [2]. Our findings point out that if voters cannot trust some authorities *w.r.t.* the fact that their vote will be counted they cannot trust the same authorities *w.r.t.* their privacy, even for entities that do not have access to the secret keys. Beyond the attack explained on a simple (and naive) protocol, our proof that privacy implies individual verifiability shows that as soon as a protocol is not verifiable, then the adversary can take advantage of the fact that he may modify a vote without being detected in order to break privacy. Individual verifiability is only one part of verifiability. It does not account for universal nor eligibility verifiability. So our result cannot be used to conclude that a private voting scheme ensures all desirable verifiability properties. Instead, it demonstrates that there is no hope to design a private voting system if it does not include some degree of verifiability, namely individual verifiability at least.

Our results also emphasise issues in existing privacy definitions. Indeed, if privacy implies individual verifiability, how is it possible to prove Helios [8] or Civitas [5] without even modelling the verification aspects? How can a system that is not fully verifiable like the Neuchâtel protocol be proved private [22]? As already pointed out in [9], existing cryptographic definitions of privacy (see [7] for a survey) implicitly assume an honest voting ballot box: honest ballots are assumed to be properly stored and then tallied. Actually, we notice that the same situation occurs in symbolic models. Although the well adopted definition of privacy [21] does not specify how the ballot box should be modelled, most symbolic proofs of privacy (see *e.g.* [5, 18, 19, 21]) actually assume that the votes of honest voters always reach the ballot box without being modified and that they are properly tallied. The reason is that the authors were aware of the fact that if the adversary may block all ballots but Alice's ballot, he can obviously break privacy. However, to avoid this apparently systematic attack, they make a very strong assumption: the ballot box needs to be honest. This means that previous cryptographic and symbolic privacy analyses only hold assuming an honest ballot box while the corresponding voting systems aim at privacy *without trusting the ballot box*. This seriously weakens the security analysis and attacks may be missed, like the attack of P. Roenne [29] on Helios, for which there is no easy fix.

Why is it so hard to define vote privacy *w.r.t.* a dishonest ballot box? Intuitively, vote privacy tries to capture the idea that, no matter how voters vote, the attacker should not be able to see any difference. The key issue is that the result of the election *does* leak some information (typically the sum of the votes) and the adversary may notice a difference based on this. This particularity makes vote privacy differ from privacy in other contexts, where the adversary really should learn no information. Therefore, most definitions of vote privacy (roughly) say that, no matter how honest voters voted, provided that the aggregation of the corresponding votes remains the same, then the attacker should not see any difference. However, as soon as the ballot box is dishonest, it may discard some honest ballots and break privacy, as already discussed. The first definition of privacy *w.r.t.* a dishonest ballot box [9] weakens privacy by requiring that among the ballots that are ready to be tallied, the (sub-)tally of the honest ones does not change. This preliminary definition has two limitations. First, it assumes that the tallied ballots are exactly the same as the cast ones, which is not the case of all protocols (*e.g.* in ThreeBallots [28], only a part of the ballot is published; in BeleniosRF [12], ballots are re-randomised). Second, it does not model re-voting: the tally process cannot discard ballots due to some revote policy.

We propose here another approach. Instead of changing the privacy definition, we now include a model of the verification process: the ballots should be tallied only if the honest voters have successfully performed the tests specified by the protocol. We compare our definition with [9] and an original definition of privacy [6] on a selection of well-studied protocols, that have different levels of verifiability (Helios, Civitas, Belenios, Neuchâtel, and our simple - non verifiable - protocol). We show again that our notion of privacy, *w.r.t.* a dishonest ballot box, implies individual verifiability. We do not consider our new definition of privacy as final but it opens the way to a better understanding of privacy in the context of fully dishonest authorities.

Threat model. We show that privacy implies individual verifiability, *under the same trust assumptions*, that is, trusting the same group of authorities, channels, etc. In symbolic models, the privacy definition does not make prior assumptions on the threat model. Instead, the encoding of the protocol defines which parties are trusted. In particular, as already discussed, existing proofs of privacy [5, 18, 19, 21] often implicitly assume that honest ballots reach the ballot box without any modification. We show that whenever privacy holds then individual verifiability holds, for the same encoding, hence the same assumptions. In contrast, most cryptographic definitions of privacy implicitly assume an honest ballot box. Therefore, we first show that privacy implies individual verifiability, assuming an honest ballot box, considering the standard definition of privacy by Benaloh [6]. Then we show that privacy still implies individual verifiability, assuming a dishonest ballot box, considering our novel definition of privacy, that explicitly models the verification steps.

Related work. As already mentioned, [13] shows an impossibility result between universal verifiability and unconditional privacy. We show in contrast that the commonly used (computational) definitions of privacy actually imply verifiability. The discrepancy between the two results comes from the fact that [13] considers unconditional privacy while most protocols achieve only computational privacy, that is against a polynomially bounded adversary. Interestingly, the impossibility result still holds between unconditional privacy and our notion of individual verifiability. [20] establishes a hierarchy between privacy, receipt-freeness, and coercion resistance, while in a quantitative setting, [27] shows that this hierarchy does not hold anymore. [16] recasts several definition of verifiability in a common setting, providing a framework to compare them. Besides [13], none of these approaches relates privacy with verifiability. Many privacy definitions have been proposed as surveyed in [7]. However, they all assume an honest ballot box. To our knowledge, [9] is the only exception, as already discussed in details. [18] shows how to break privacy by replaying a ballot. If an attacker may replay Alice's ballot and cast it in his own name (or cast a related ballot), then he introduces a bias in the result, that leaks some information on Alice's vote. Note that this replay attack does not break individual verifiability: honest votes are correctly counted. We show here another breach for privacy: if an attacker may remove some honest votes, then he breaks privacy as well.

Roadmap. We first prove that privacy implies individual verifiability in symbolic models, in Section 3, and then in cryptographic models, in Section 4. These two parts are rather independent. In Section 6, we examine a selection of well-studied voting protocols and compare the effect of different (cryptographic) notions of privacy when the ballot box is dishonest.

2 PRELIMINARIES

Notations: The multiset of elements a, a, b, c is denoted $\{\!|a, a, b, c|\!\}$. The union of two multisets S_1 and S_2 is denoted $S_1 \uplus S_2$.

In both cryptographic and symbolic models, we assume a set $\mathcal{V}$ of votes and a set $\mathcal{R}$ of possible results, equipped with an associative and commutative operator $*$ (*e.g.* addition of vectors). A *counting function* is a function ρ that associates a result $r \in \mathcal{R}$ to a multiset of votes. We assume that counting functions have a *partial tally* property: it is always possible to count the votes in two distinct multisets and then combine the results.

$$\forall V, V' \; \rho(V \uplus V') = \rho(V) * \rho(V')$$

A vote v is said to be *neutral* if $\rho(v)$ is neutral *w.r.t.* $*$.

Example 2.1. Consider a finite set of candidates $C = \{a_1, \dots, a_k\}$. In case voters should select between k_1 and k_2 candidates or vote blank, we can represent valid votes by vectors representing the selection of candidates

$$\mathcal{V}_{k_1,k_2} = \left\{ v \in \{0,1\}^k \;\middle|\; k_1 \leq \sum_{i=0}^{k} v_i \leq k_2 \right\} \cup \{v^{\mathsf{blank}}\}$$

where v^{blank} is the null vector $(0, \dots, 0)$, representing a blank vote. In a mixnet-based tally, all the individual votes are revealed. Thus $\mathcal{R}$ is the set of multisets of votes in $\mathcal{V}_{k_1,k_2}$ and $*$ is the union of multisets. The corresponding counting function is $\rho_{mix}(V) = V$, where V is a multiset of elements of $\mathcal{V}_{k_1,k_2}$.

In an homomorphic-based tally, the votes are added together. Thus $\mathcal{R} = \mathbb{N}^k$, the set of vectors of k elements, and $*$ is the addition of vectors. The corresponding counting function is $\rho_{hom}(V) = \sum_{v \in V} v$.

Both ρ_{mix} and ρ_{hom} have the partial tally property. The vote v^{blank} is a neutral vote *w.r.t.* ρ_{hom} but not ρ_{mix}.

The result of the election r may have several representations. For example, a multiset may be represented by several lists (where the order changes). In symbolic models, the result will be represented by abstract terms and we wish our result to be independent of a particular choice of representation. Therefore, we will simply say that a *representation* R is a function that associates to a result $r \in \mathcal{R}$ a set of possible representations with an injectivity property:

$$\forall r \neq r'. \; R(r) \cap R(r') = \emptyset$$

Intuitively, a result can be associated to several representations but a given representation can correspond to at most one result.

For our proofs in a cryptographic setting, we will also assume that given an election result r and a set of votes V, one can decide efficiently (in polynomial time) whether r includes all the votes of V, that is, whether there exists V' such that $r = \rho(V \uplus V')$. This condition is satisfied by ρ_{mix} and ρ_{hom} and all standard counting functions.

3 SYMBOLIC MODEL

3.1 Model

In symbolic models, security protocols are often modelled through a process algebra, in the spirit of the applied pi-calculus [3], that offers a small, abstract language for specifying communications, where messages are represented as terms. We present here a calculus inspired from the calculus underlying the ProVerif tool [10].

3.1.1 Terms. We consider an infinite set of names $\mathcal{N}$ that model fresh values such as nonces and keys. We distinguish the set $\mathcal{FN}$ of free nonces (generated by the attacker) and the set $\mathcal{BN}$ of bound nonces (generated by the protocol agents). We also assume an infinite set of variables $\mathcal{V} = \mathcal{X} \uplus \mathcal{AX}$ where $\mathcal{X}$ contains variables used in processes (agent's memory) while $\mathcal{AX}$ contains variables used to store messages (adversary's memory). Cryptographic primitives are represented through a set of function symbols, called *signature* $\mathcal{F}$. Each function symbol has an arity, that is, the number of its arguments. We assume an infinite set $C \subseteq \mathcal{F}$ of public constants, which are functions of arity 0.

Example 3.1. The standard primitives, public keys, symmetric and asymmetric encryption, concatenation, as well as addition, can be modelled by the following signature.

$$\mathcal{F}_c = \{\mathsf{pk}/1, \mathsf{enc}/2, \mathsf{aenc}/2, \langle\cdot,\cdot\rangle/2, +/2\}$$

The companion primitives (symmetric and asymmetric decryption, projections) are then represented by the following signature:

$$\mathcal{F}_d = \{\mathsf{dec}/2, \mathsf{adec}/2, \pi_1/1, \pi_2/1\}$$

Given a signature $\mathcal{F}$, a set of names $\mathcal{N}$, a set of variables $\mathcal{V}$, the set of *terms* $\mathcal{T}(\mathcal{F}, \mathcal{V}, \mathcal{N})$ is the set inductively defined by applying functions to variables in $\mathcal{V}$ and names in $\mathcal{N}$. The set of names resp. variables) occurring in t is denoted $\mathsf{names}(t)$ (resp. $\mathsf{vars}(t)$). A term is *ground* if it does not contain any variable. The set of terms $\mathcal{T}(\mathcal{F}, \mathcal{AX}, \mathcal{FN})$ represents the *attacker terms*, that is, terms built from the messages sent on the network and stored thanks to the variables in $\mathcal{AX}$.

A *substitution* $\sigma = \{M_1/x_1, \dots, M_k/x_k\}$ maps variables $x_1, \dots, x_k \in \mathcal{V}$ to messages $M_1, \dots, M_k$. Its *domain* is denoted $\mathsf{dom}(\sigma) = \{x_1, \dots, x_k\}$. The application of σ to a term t is denoted $t\sigma$ and is defined as usual. A substitution σ is ground if its messages $M_1, \dots, M_k$ are ground.

The properties of the cryptographic primitives are modelled through an *equational theory* E, which is a finite set of equations of the form $M = N$ where $M, N \in \mathcal{T}(\mathcal{F}, \mathcal{X}, \emptyset)$ are messages without names. Equality modulo E, denoted by $=_{\mathsf{E}}$, is defined as the smallest equivalence relation on terms that is closed under context and substitution. We denote disequalities modulo E by $M \neq_{\mathsf{E}} N$.

Example 3.2. Considering the signature $\mathcal{F}_c \cup \mathcal{F}_d \cup C$ from Example 3.1, the following equational theory describes the ability to decrypt symmetrically, asymmetrically, and to project pairs. It also

Processes:

$$
\begin{array}{rlll}
P, Q & ::= & & \\
& & \mathtt{0} & \\
& | & \nu\, n.P & \text{for } n \in \mathcal{BN} \ (n \text{ bound in } P) \\
& | & \mathtt{out}(c, M).P & \\
& | & \mathtt{in}(c, x).P & \text{for } x \in \mathcal{X} \ (x \text{ bound in } P) \\
& | & \mathsf{event}(M_1, \ldots, M_n).P & \text{for event} \in \mathsf{Ev} \text{ of arity } n \\
& | & P \mid Q & \\
& | & \mathtt{let}\ x = M\ \mathtt{in}\ P & \text{for } x \in \mathcal{X} \ (x \text{ bound in } P) \\
& | & \mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q & \\
& | & !P &
\end{array}
$$

where $M, N, M_1, \ldots, M_n$ are messages and $c \in Ch$ is a channel.

Figure 1: Syntax for processes.

characterises + as an associative and commutative operator.

$$
\begin{aligned}
\mathsf{dec}(\mathsf{enc}(x, y), y) &= x \\
\mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) &= x \\
\pi_1(\langle x, y \rangle) &= x \\
\pi_2(\langle x, y \rangle) &= y \\
x + (y + z) &= (x + y) + z \\
x + y &= y + x
\end{aligned}
$$

3.1.2 Processes. The behaviour of protocol parties is described through processes. Let Ch be an infinite set of channel names, representing the channels on which the messages are exchanged. All channels will be public. We consider different channels nevertheless to model the fact that an attacker can identify the provenance of a message. We also consider a finite set Ev of event symbols, given together with their arity. Events are used to record that participants have reached a certain step, with some associated knowledge. Protocols are modelled through a process algebra, whose syntax is displayed in Figure 1.

As usual, we identify processes up to α-renaming, to avoid capture of bound names and variables.

A *configuration* of the system is a triple $(\mathsf{E}; \mathcal{P}; \phi)$ where:

- $\mathcal{P}$ is a multiset of processes that represents the current active processes;
- $\mathcal{E}$ is a set of names, which represents the private names of the processes;
- ϕ is a substitution with $\mathsf{dom}(\phi) \subseteq \mathcal{AX}$ that represents the messages sent on the network. We assume ϕ to be ground, that is for any $x \in \mathsf{dom}(\phi)$, $\phi(x)$ is a ground term.

The semantics of processes is given through a transition relation $\xrightarrow{\alpha}$ provided in Figure 2, where α is the *action* associated to the transition. τ denotes a silent action. Events are recorded but will be invisible to the attacker. Intuitively, process $\nu\, n.P$ creates a fresh nonce, stored in $\mathcal{E}$, and behaves like P. Process $\mathtt{out}(c, M).P$ emits M on c and behaves like P. Process $\mathtt{in}(c, x).P$ inputs a term computed by the attacker (that is a term built from ϕ using an attacker term) on channel c and then behaves like P. Process $\mathsf{event}(M_1, \ldots, M_n).P$ triggers the event $\mathsf{event}(M_1, \ldots, M_n)$, and then behaves like P. Process $P \mid Q$ corresponds to the parallel composition of P and Q. Process $\mathtt{let}\ x = M\ \mathtt{in}\ P$ behaves like P in which x is replaced with M. Process $\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$ behaves like P if M and N are equal modulo E, and behaves like Q otherwise. The replicated process $!P$ behaves as an unbounded number of copies of P.

We denote by $\xrightarrow{w}_*$ the reflexive transitive closure of $\xrightarrow{\alpha}$, where w is the concatenation of all actions. We also write equality up to silent actions and events $=_\tau$.

A *trace* of a process P is any possible sequence of transitions starting from P. Traces correspond to all possible executions in the presence of an attacker that may read, forge, and send messages. Formally, the set of traces $\mathsf{trace}(P)$ is defined as follows.

$$
\mathsf{trace}(P) = \{(w, \mathsf{new}\ \mathcal{E}.\phi) | (\emptyset; \{P\}; \emptyset) \xrightarrow{w}_* (\mathcal{E}; \mathcal{P}; \phi)\}
$$

A sequence of actions t is *blocking* in a process P if it cannot be executed.

$$
\mathsf{blocking}(t, P) \stackrel{\text{def}}{=} \forall \phi.\ (t, \phi) \notin \mathsf{trace}(P).
$$

Example 3.3. Helios [4] is a simple voting protocol used in several elections, like the election of the recteur of the university of Louvain-la-Neuve. A voter simply encrypts her vote with the public key of the election. This encrypted vote forms the ballot, which is sent to the ballot box. The voter may check that her ballot is on the ballot box since the ballot box is public. There are two ways for tallying, either homomorphic tally or mixnet-based tally. We model here the two options in an abstract way: given the ballots, the talliers output the aggregation of the decryption of the ballot. This aggregation could be the addition or just the votes in a random order. For simplicity, we describe here a simple version with only two honest voters A and B, a dishonest voter C, and a voting server S. This protocol can be modelled by the following process.

$$
\begin{array}{l}
P_{\mathsf{Helios}}(v_a, v_b) = \\
\quad \nu\, k_{as}, k_{bs}, k_{cs}, k_e. \\
\qquad (\mathsf{out}(c, k_{cs}).\mathsf{out}(c, \mathsf{pk}(k_e)))\ | \\
\qquad \mathsf{Voter}(A, v_a, c_a, c'_a, k_{as}, k_e)\ |\ \mathsf{Voter}(B, v_b, c_b, c'_b, k_{bs}, k_e)\ | \\
\qquad \mathsf{Tally}_{\mathsf{Helios}}(c_a, c_b, c_c, c_s, k_{as}, k_{bs}, k_{cs}, k_e))
\end{array}
$$

where $\mathsf{Voter}(a, v, c, c', k, k_e)$ represents voter a willing to vote for v using the channels c and c', the election key k_e and the credential k to authenticate to the server, while $\mathsf{Tally}_{\mathsf{Helios}}$ represents the voting server.

$\mathsf{Voter}(a, v, c, c', k, k_e)$ simply sends an encrypted vote. To model the fact that voters communicate with the ballot box through an authenticated channel, we assume that a voter first sends her ballot privately to the server (using the encryption with k) and then sends the ballot on a public channel. Note that the key k is just a modelling artefact to abstract away the underlying password-based authenticated channel.

$$
\begin{array}{l}
\mathsf{Voter}(a, v, c, c', k, k_e) = \\
\quad \nu\, r.\ \mathsf{out}(c, \mathsf{enc}(\mathsf{aenc}(\langle v, r \rangle, \mathsf{pk}(k_e)), k)).\ \mathsf{Voted}(a, v). \\
\qquad\qquad\qquad\qquad \mathsf{out}(c', \mathsf{aenc}(\langle v, r \rangle, \mathsf{pk}(k_e)))
\end{array}
$$

The voting server receives ballots from voters A, B, and C and then outputs the decrypted ballots, after some mixing, modelled

$$
\begin{array}{rclr}
(\mathcal{E}; \{P_1 \mid P_2\} \cup \mathcal{P}; \phi) & \xrightarrow{\tau} & (\mathcal{E}; \{P_1, P_2\} \cup \mathcal{P}; \phi) & \textsc{Par} \\
(\mathcal{E}; \{\mathtt{0}\} \cup \mathcal{P}; \phi) & \xrightarrow{\tau} & (\mathcal{E}; \mathcal{P}; \phi) & \textsc{Zero} \\
(\mathcal{E}; \{\nu\, n.P\} \cup \mathcal{P}; \phi) & \xrightarrow{\tau} & (\mathcal{E} \cup \{n\}; \{P\} \cup \mathcal{P}; \phi) & \textsc{New} \\
(\mathcal{E}; \{\mathtt{out}(c, M).P\} \cup \mathcal{P}; \phi) & \xrightarrow{\nu\, ax_n.\mathtt{out}(c, ax_n)} & (\mathcal{E}; \{P\} \cup \mathcal{P}; \phi \cup \{M/ax_n\}) & \textsc{Out} \\
& \multicolumn{2}{r}{\text{if } M \text{ is a ground term, } ax_n \in \mathcal{AX} \text{ and } n = |\phi| + 1} & \\
(\mathcal{E}; \{\mathtt{in}(c, x).P\} \cup \mathcal{P}; \phi) & \xrightarrow{\mathtt{in}(c, R)} & (\mathcal{E}; \{P[R\phi/x]\} \cup \mathcal{P}; \phi) & \textsc{In} \\
& \multicolumn{2}{r}{\text{if } R \text{ is an attacker term such that } \mathsf{vars}(R) \subseteq \mathsf{dom}(\phi)} & \\
(\mathcal{E}; \{\mathsf{event}(M_1, \ldots, M_n).P\} \cup \mathcal{P}; \phi) & \xrightarrow{\mathsf{event}(M_1, \ldots, M_n)} & (\mathcal{E}; \{P\} \cup \mathcal{P}; \phi) & \textsc{Event} \\
& \multicolumn{2}{r}{\text{if } \forall i.\ M_i \text{ is a ground message}} & \\
(\mathcal{E}; \{\mathtt{let}\ x = M\ \mathtt{in}\ P\} \cup \mathcal{P}; \phi) & \xrightarrow{\tau} & (\mathcal{E}; \{P[M/x]\} \cup \mathcal{P}; \phi) & \textsc{Let-In} \\
& \multicolumn{2}{r}{\text{if } M \text{ is ground}} & \\
(\mathcal{E}; \{\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q\} \cup \mathcal{P}; \phi) & \xrightarrow{\tau} & (\mathcal{E}; \{P\} \cup \mathcal{P}; \phi) & \textsc{If-Then} \\
& \multicolumn{2}{r}{\text{if } M, N \text{ are ground messages such that } M =_{\mathsf{E}} N} & \\
(\mathcal{E}; \{\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q\} \cup \mathcal{P}; \phi) & \xrightarrow{\tau} & (\mathcal{E}; \{Q\} \cup \mathcal{P}; \phi) & \textsc{If-Else} \\
& \multicolumn{2}{r}{\text{if } M, N \text{ are ground messages such that } M \neq_{\mathsf{E}} N} & \\
(\mathcal{E}; \{!P\} \cup \mathcal{P}; \phi) & \xrightarrow{\tau} & (\mathcal{E}; \{P, !P\} \cup \mathcal{P}; \phi) & \textsc{Repl}
\end{array}
$$

Figure 2: Semantics

through the + operator.

$$
\begin{array}{l}
\mathsf{Tally}_{\mathsf{Helios}}(c_a, c_b, c_c, c_s, k_{as}, k_{bs}, k_{cs}, k_e) = \\
\quad \mathtt{in}(c_a, x_1).\mathtt{in}(c_b, x_2).\mathtt{in}(c_c, x_3). \\
\quad \mathtt{let}\ y_1 = \mathsf{dec}(x_1, k_{as})\ \mathtt{in} \\
\quad \mathtt{let}\ y_2 = \mathsf{dec}(x_2, k_{bs})\ \mathtt{in} \\
\quad \mathtt{let}\ y_3 = \mathsf{dec}(x_3, k_{cs})\ \mathtt{in} \\
\quad \mathtt{if}\ x_1 \neq x_2\ \wedge\ x_1 \neq x_3\ \wedge\ x_2 \neq x_3\ \mathtt{then} \\
\qquad \mathtt{out}(c_s, \pi_1(\mathsf{adec}(y_1, k_e))\ +\ \pi_1(\mathsf{adec}(y_2, k_e)) \\
\qquad\qquad\qquad\qquad + \pi_1(\mathsf{adec}(y_3, k_e)))
\end{array}
$$

where we omit the null else-branches. $\wedge$ is syntactic sugar for a succession of tests and `if` $M \neq N$ `then` P is syntactic sugar for `if` $M = N$ `then 0 else` P.

3.1.3 Equivalence. Sent messages are stored in a substitution ϕ while private names are stored in $\mathcal{E}$. A *frame* is simply an expression of the form new $\mathcal{E}.\phi$ where $\mathsf{dom}(\phi) \subseteq \mathcal{AX}$. It represents the knowledge of an attacker. We define $\mathsf{dom}(\text{new } \mathcal{E}.\phi)$ as $\mathsf{dom}(\phi)$.

Intuitively, two sequences of messages are indistinguishable to an attacker if he cannot perform any test that could distinguish them. This is typically modelled as static equivalence [3].

Definition 3.4 (Static Equivalence). Two ground frames new $\mathcal{E}.\phi$ and new $\mathcal{E}'.\phi'$ are statically equivalent if and only if they have the same domain, and for all attacker terms R, S with variables in $\mathsf{dom}(\phi) = \mathsf{dom}(\phi')$, we have

$$(R\phi =_{\mathsf{E}} S\phi) \iff (R\phi' =_{\mathsf{E}} S\phi')$$

Two processes P and Q are in equivalence if no matter how the adversary interacts with P, a similar interaction may happen with Q, with equivalent resulting frames.

Definition 3.5 (Trace Equivalence). Let P, Q be two processes. We write $P \sqsubseteq_t Q$ if for all $(s, \psi) \in \mathsf{trace}(P)$, there exists $(s', \psi') \in \mathsf{trace}(Q)$ such that $s =_\tau s'$ and ψ and ψ' are statically equivalent. We say that P and Q are trace equivalent, and we write $P \approx_t Q$, if $P \sqsubseteq_t Q$ and $Q \sqsubseteq_t P$.

Note that this definition already includes the attacker's behaviour, since processes may input any message forged by the attacker.

Example 3.6. Ballot privacy is typically modelled as an equivalence property [21] that requires that an attacker cannot distinguish when Alice is voting 0 and Bob is voting 1 from the scenario where the two votes are swapped.

Continuing Example 3.3, ballot privacy of Helios can be expressed as follows:

$$P_{\mathsf{Helios}}(0, 1) \approx_t P_{\mathsf{Helios}}(1, 0)$$

3.2 Voting protocols

We consider two disjoint, infinite subsets of C: a set $\mathcal{A}$ of *agent names* or *identities*, and a set $\mathcal{V}$ of *votes*. We assume given a representation R of the result.

A voting protocol is modelled as a process. It is composed of:

- processes that represent honest voters;
- a process modelling the tally;
- possibly other processes, modelling other authorities.

Formally, we define a *voting process* as follows.

Definition 3.7. A *voting process* is a process of the form

$$
\begin{array}{l}
P = \nu\ \overrightarrow{cred}.\nu\ cred_1 \ldots \nu\ cred_p.\ (\\
\qquad \mathsf{Voter}(a_1, v_{a_1}, \overrightarrow{c_1}, \overrightarrow{cred}, cred_1) \mid \cdots \mid \\
\qquad\qquad \mathsf{Voter}(a_n, v_{a_n}, \overrightarrow{c_n}, \overrightarrow{cred}, cred_n) \\
\quad \mid \mathsf{Tally}_p(\overrightarrow{c}, \overrightarrow{cred}, cred_1, \ldots, cred_p) \\
\quad \mid \mathsf{Others}_p(\overrightarrow{c}', \overrightarrow{cred}, cred_1, \ldots, cred_p))
\end{array}
$$

where $a_i \in \mathcal{A}$, $v_{a_i} \in \mathcal{V}$, $\overrightarrow{c_i}$, $\overrightarrow{c}$, $\overrightarrow{c}'$ are (distinct) channels, $\overrightarrow{cred}$ and $cred_i$ are (distinct) names.

A voting process may be instantiated by various voters and vote selections. Given $A = \{b_1, \ldots, b_n\} \subseteq \mathcal{A}$ a finite set of voters, and $\alpha : A \to \mathcal{V}$ that associates a vote to each voter, we define P_α by replacing a_i by b_i and v_i by $\alpha(b_i)$ in P.

Moreover, P must satisfy the following properties.

- Process $\mathsf{Voter}(a, v_a, \vec{c}, \overrightarrow{cred}, cred)$ models an honest voter a willing to vote for v_a, using the channels $\vec{c}$, credentials $cred$ (*e.g.* a signing key) and election credentials $\overrightarrow{cred}$. It is assumed to contain an event $\mathsf{Voted}(a, v)$ that models that a has voted for v. This event is typically placed at the end of process $\mathsf{Voter}(a, v_a, \vec{c}, \overrightarrow{cred}, cred)$. This event cannot appear in process Tally_p nor Others_p.
- Process $\mathsf{Tally}_p(\vec{c}, \overrightarrow{cred}, cred_1, \ldots, cred_p)$ models the tally. It is parametrised by the total number of voters p (honest and dishonest), with $p \geq n$. It is assumed to contain exactly one output action on a reserved channel c_r. The term output on this channel is assumed to represent the final result of the election.

$$\forall \alpha.\ \forall (tr, \phi) \in \mathsf{trace}(P_\alpha).\ \mathsf{out}(c_r, r) \in tr \implies \exists V.\ \phi(r) \in R(\rho(V))$$

  Tally_p may of course contain input/output actions on other channels.
- Process $\mathsf{Others}_p(\vec{c}', \overrightarrow{cred}, cred_1, \ldots, cred_p)$ is an arbitrary process, also parametrised by p. It models the remaining of the voting protocol, for example the behaviour of other authorities. It also models the initial knowledge of the attacker by sending appropriate data (*e.g.* the public key of the election or dishonest credentials). We simply assume that it uses a set of channels disjoint from the channels used in Voter and Tally_p.

The channel c_r used in Tally_p to publish the result is called the *result channel* of P.

Example 3.8. The process modelling the Helios protocol, as defined in Example 3.3 is a voting process, where process Others_p consists in the output of the keys: $\mathsf{out}(c, k_{cs}).\mathsf{out}(c, \mathsf{pk}(k_e))$.

We can read which voters voted from a trace. Formally, given a sequence tr of actions, the set of voters $\mathsf{Voters}(tr)$ who did vote in tr is defined as follows.

$$\mathsf{Voters}(tr) = \{a \in \mathcal{A} \mid \exists v \in \mathcal{V}.\ \mathsf{Voted}(a, v) \in tr\}.$$

The result of the election is emitted on a special channel c_r. It should correspond to the tally of a multiset of votes. Formally, given a trace (t, ϕ) and a multiset of votes V, the predicate $\mathsf{result}(t, \phi, V)$ holds if the election result in (t, ϕ) corresponds to V.

$$\mathsf{result}(t, \phi, V) \overset{\mathrm{def}}{=} \exists x, t'.\ t = t'.\mathsf{out}(c_r, x) \ \wedge\ \phi(x) \in R(\rho(V)).$$

3.3 Security properties

Several definitions of verifiability have been proposed. In the lines of [15, 26], we consider a very basic notion, that says that the result should at least contain the votes from honest voters.

Definition 3.9 (symbolic individual verifiability). Let P be a voting process with result channel c_r. P satisfies *symbolic individual verifiability* if, for any trace $(t, \phi) \in \mathsf{trace}(P_\alpha)$ of the form $t'.\mathsf{out}(c_r, x)$), there exists V_c such that the result in t corresponds to $V_a \uplus V_c$, that is $\mathsf{result}(t, \phi, V_a \uplus V_c)$, where

$$V_a = \{\!\!\{ v \mid \exists a.\ \mathsf{Voted}(a, v) \in t \}\!\!\}$$

Individual verifiability typically guarantees that voters can check that their ballot will be counted. Our notion of individual verifiability goes one step further, ensuring that the corresponding votes will appear in the result, even if the tally is dishonest. One of the first definitions of verifiability was given in [25], distinguishing between individual, universal, and eligibility verifiability. Intuitively, our own notion of individual verifiability sits somewhere between individual verifiability and individual plus universal verifiability as defined in [25]. A precise comparison is difficult as individual and universal verifiability are strongly tight together in [25]. Moreover, [25] only considers the case where all voters are honest and they all vote.

We consider the privacy definition proposed in [21] and widely adopted in symbolic models: an attacker cannot distinguish when Alice is voting v_1 and Bob is voting v_1 from the scenario where the two votes are swapped.

Definition 3.10 (Privacy [21]). Let P be a voting process. P satisfies *privacy* if, for any subtitution α from voters to votes, for any two voters $a, b \in \mathcal{A} \backslash \mathsf{dom}(\alpha)$ and any two votes $v_1, v_2 \in \mathcal{V}$, we have

$$P_{\alpha \cup \{a \mapsto v_1, b \mapsto v_2\}} \approx P_{\alpha \cup \{a \mapsto v_2, b \mapsto v_1\}}$$

3.4 Privacy implies verifiability

We show that privacy implies verifiability under a couple of assumptions, typically satisfied in practice.

First, we assume a light form of determinacy: two traces with the same observable actions yield the same election result. This excludes for example cases for voters chose non deterministically how they vote. Formally, we say that a voting process P with election channel c_r is *election determinate* if, for any substitution α from voters to votes, for any two traces t, t' such that $t =_\tau t'$, $(t.\mathsf{out}(c_r, x), \phi) \in \mathsf{trace}(P_\alpha)$, and $(t'.\mathsf{out}(c_r, x), \phi') \in \mathsf{trace}(P_\alpha)$, then

$$\phi(x) \in R(\rho(V))) \Rightarrow \phi'(x) \in R(\rho(V))$$

This assumption still supports some form of non determinism but may not hold for example in the case where voters use anonymous channels that even hide who participated in the election.

Second, we assume that it is always possible for a new voter to vote (before the tally started) without modifying the behaviour of the protocol.

Formally, a voting proces P is *voting friendly* if for all voter $a \in \mathcal{A}$, there exists t'' (the honest voting trace) such that for all α satisfying $a \notin \mathsf{dom}(\alpha)$,

- for all $(t, \phi) \in \mathsf{trace}(P_\alpha)$, such that $t = t'.\mathsf{out}(c_r, x)$ for some t', x, for all v, there exists tr, ψ such that $tr =_\tau t''$, $\mathsf{Voted}(a, v) \in tr$, $(t'.tr.\mathsf{out}(c_r, x), \psi) \in \mathsf{trace}(P_{\alpha \cup \{a \mapsto v\}})$, and $\forall V.\ \phi(x) \in R(\rho(V)) \Rightarrow \psi(x) \in R(\rho(V \cup \{v\}))$. Intuitively, if a votes normally, her vote will be counted as expected, no matter how the adversary interfered with the other voters.
- for all t', x such that $\mathsf{blocking}(t'.\mathsf{out}(c_r, x), P_\alpha)$, for all v, tr, ψ such that $tr =_\tau t''$, we have $\mathsf{blocking}(t'.tr.\mathsf{out}(c_r, x), P_{\alpha \cup \{a \mapsto v\}})$. Intuitively, the fact that a voted does not suddenly unlock the tally.

In practice, most voting systems are voting friendly since voters vote independently. In particular, process P_{Helios} modelling Helios, as defined in Example 3.3, is voting friendly (assuming an honest tally). The voting friendly property prevents a fully dishonest tally since the first item requires that unmodified honest ballots are correctly counted. However, we can still consider a partially dishonest tally that, for example, discards or modifies ballots that have been flagged by the attacker.

Moreover, we assume that there exists a neutral vote, which is often the case in practice. Actually, this is a simplified (sufficient) condition. Our result also holds as soon as there is a vote that can be counted separately from the other votes (as formally defined in appendix).

Theorem 3.11 (Privacy implies individual verifiability). *Let P be a voting friendly, election determinate voting process. If P satisfies privacy then P satisfies individual verifiability.*

The proof of this result intuitively relies on the fact that in order to satisfy privacy *w.r.t.* two voters Alice and Bob, a voting process has to guarantee that the vote of Alice is, if not correctly counted, at least taken into account to some extent. Indeed, if an attacker, trying to distinguish whether Alice voted for 0 and Bob for 1, or Alice voted for 1 and Bob for 0, is able to make the tally ignore completely the vote of Alice, the result of the election is then Bob's choice. Hence the attacker learns how Bob voted, which breaks privacy.

Therefore, we first we prove that if a protocol satisfies privacy, then if we compare an execution (*i.e.* a trace) where Alice votes 0 with the corresponding execution where Alice votes 1, the resulting election results must differ by exactly a vote for 0 and a vote for 1. Formally, we show the following property.

Lemma 3.12. *If a voting friendly, election determinate voting process P satisfies privacy, then it satisfies*

$$\begin{aligned}&[\ t =_\tau t' \wedge (t,\phi) \in \mathsf{trace}(P_{\alpha\cup\{a\mapsto v_1\}}) \wedge \\ &\quad (t',\phi') \in \mathsf{trace}(P_{\alpha\cup\{a\mapsto v_2\}}) \wedge \\ &\quad \mathsf{result}(t,\phi,V) \ \wedge\ \mathsf{result}(t',\phi',V')\ ] \quad \Longrightarrow \\ &\rho(V' \uplus \{\!\!\{v_1\}\!\!\}) = \rho(V \uplus \{\!\!\{v_2\}\!\!\}).\end{aligned}$$

This lemma is used as a central property to prove the theorem. Intuitively, we apply this lemma repeatedly, changing one by one all the votes from honest voters into neutral votes. Let r denote the result before this operation, and r' the result after. Let V_a denote the multiset of honest votes, and V_b the multiset containing the same number of neutral votes. Thanks to Lemma 3.12, we can show that $r * \rho(V_b) = r' * \rho(V_a)$. Since V_b only contains neutral votes, we have $r = r' * \rho(V_a)$. This means that r contains all honest votes, hence the voting process satisfies individual verifiability.

The detailed proof of this theorem can be found in appendix.

4 COMPUTATIONAL MODEL

Computational models define protocols and adversaries as probabilistic polynomial-time algorithms.

Notation: We may write $(id, *) \in \mathsf{L}$ as a shorthand, meaning that there exists an element of the form (id, x) in L. If V is a multiset of elements of the form (id, v), we define $\rho(V) = \rho(\{\!\!\{v \mid (id, v) \in V\}\!\!\})$.

4.1 Voting system

We assume that the ballot box displays a board BB, that is a list of ballots. The nature of the ballots depend on the protocol we consider.

Definition 4.1. A *voting scheme* consists in six algorithms

$$(\mathsf{Setup}, \mathsf{Credential}, \mathsf{Vote}, \mathsf{VerifVoter}, \mathsf{Tally}, \mathsf{Valid})$$

- $\mathsf{Setup}(1^\lambda)$, given a security parameter λ, returns a pair of *election keys* (pk, sk).
- $\mathsf{Credential}(1^\lambda, id)$ creates a *credential cred* for voter *id*, for example a signing key. The credential may be empty as well. Registered voters are stored in a list U.
- $\mathsf{Vote}(id, cred, \mathsf{pk}, v)$ constructs a ballot containing the vote v for voter *id* with credential *cred*, using the election public key pk.
- $\mathsf{VerifVoter}(id, cred, \mathsf{L}, \mathsf{BB})$ checks whether the local knowledge L of voter *id* is consistent with the board BB. For example, a voter may check that her (last) ballot appears on the bulletin board.
- $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U})$ computes the *tally* of the ballots on the board BB, using the election secret key sk, assuming a list of registered voter identities and credentials U. The Tally algorithm first runs some test $\mathsf{ValidTally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U})$ that typically checks that the ballots of BB are valid. Tally may return $\perp$ if the tally procedure fails (invalid board or decryption failure for example). If $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) \neq \perp$ then it must correspond to a valid result, that is, there exists V such that $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \rho(V)$.
- $\mathsf{Valid}(id, b, \mathsf{BB}, \mathsf{pk})$ checks that a ballot b cast by a voter *id* is valid with respect to the board BB using the election public key pk. For example, the ballot b should have a valid signature or valid proofs of knowledge. The ballot b will be added to BB only if $\mathsf{Valid}(id, b, \mathsf{BB}, \mathsf{pk})$ succeeds.

We will always assume a *correct* voting scheme, that is, tallying honestly generated ballots yields the expected result. Formally, for all distinct identities $\mathsf{U} = id_1, \ldots, id_n$, and credentials $cred_1, \ldots, cred_n$, for all votes $v_1, \ldots, v_n$, for all election keys (pk, sk), if $\mathsf{BB} = [\mathsf{Vote}(id_i, cred_i, \mathsf{pk}, v_i) | i \in [\![1, n]\!]]$, then

$$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \rho(\{\!\!\{v_1, \ldots, v_n\}\!\!\})$$

The tally algorithm typically applies a revote policy. Indeed, if voters may vote several times, the revote policy states which vote should be counted. The two main standard revote policies are 1. the last vote counts or 2. the first vote counts (typically when revote is forbidden). In what follows, our definitions are written assuming the last ballot revote policy. However, they can easily be adapted to the first ballot revote policy and all our results hold in both cases (as shown in appendix).

The revote policy is either based on the identities or the credentials. We say that a voting system is *id-based* if there exists a a function open_{id} which, given a ballot b, retrieves the associated identity. Formally, for any $id, cred, \mathsf{pk}, v$,

$$\mathsf{open}_{id}(\mathsf{Vote}(id, cred, \mathsf{pk}, v)) = id$$

Similarly, we say that a voting system is *cred-based* if there exists a function open_{cred} which, given a ballot b, the election secret key sk, and a list U of registered voters and credentials, retrieves the credential *cred* used by the voter to create the ballot. Formally, for any $id, cred, \mathsf{sk}, \mathsf{pk}, v$,

$$\mathsf{open}_{cred}(\mathsf{Vote}(id, cred, \mathsf{pk}, v), \mathsf{sk}, U) = cred$$

$\mathcal{O}_{\text{reg}}(id)$	$\mathcal{O}_{\text{corr}}(id)$
if $(id, *) \in \mathsf{U}$ **then**	**if** $(id, *) \notin \mathsf{U} \ \vee \ (id, *) \in \mathsf{CU}$ **then**
stop	stop
else	**else**
$cred_{id} \leftarrow \mathsf{Credential}(1^\lambda, id)$	$\mathsf{CU} \leftarrow \mathsf{CU} \| (id, cred_{id})$
$\mathsf{U} \leftarrow \mathsf{U} \| (id, cred_{id})$	**return** $cred_{id}$
	where $(id, cred_{id}) \in \mathsf{U}$

Figure 3: Registration and corruption oracles

Note that some schemes are neither id-based nor cred-based, in particular when the ballots contain no identifier. Such schemes typically assume that voters do not revote since there is no means to identify whether two ballots originate from the same voter.

4.2 Security properties

As usual, an *adversary* is any probabilistic polynomial time Turing machine (PPTM). We define verifiability and privacy through game-based properties.

4.2.1 Verifiability. For verifiability, we propose a simple definition, inspired from [15, 26]. Intuitively, we require that the election result contains at least the votes of all honest voters. This notion was called weak verifiability in [15] but we will call it individual verifiability to match the terminology used in symbolic settings. More sophisticated and demanding definitions have been proposed, for example controlling how many dishonest votes can be inserted [15] or tolerating some variations in the result [26]. The main missing part (in terms of security) is that our definition does not control ballot stuffing: arbitrarily many dishonest votes may be added to the result. The reason is that ballot stuffing seems unrelated to privacy. Moreover, our definition assumes an honest tally, and thus does not capture universal verifiability aspects. The main reason is that existing privacy definitions in computational settings assume an honest tally and we compare the two notions under the same trust assumptions. We leave as future work to determine how to extend these two definitions to a dishonest tally, and whether the implication still holds.

Verifiability is defined through the game $\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda)$ displayed on Figure 4. In a first step, the adversary may use oracles $\mathcal{O}_{\text{reg}}(id)$ and $\mathcal{O}_{\text{corr}}(id)$ (defined on Figure 3) to respectively register a voter and get her credential (in this case, the voter is said to be corrupted). Then the adversary may ask an honest voter id to vote for a given vote v through oracle $\mathcal{O}^{v}_{\text{vote}}(id, v)$. In this case, the adversary sees the corresponding ballot and the fact that id voted for v is registered in the list Voted. The adversary may also cast an arbitrary ballot b in the name of a dishonest voter id through oracle $\mathcal{O}_{\text{cast}}(id, b)$. Finally, the adversary wins if the election result does not contain all the honest votes registered in Voted (where only the last vote is counted).

Definition 4.2 (Individual verifiability). A voting system is *individually verifiable* if for any adversary $\mathcal{A}$,

$$\mathrm{P}\left[\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda) = 1\right] \text{ is negligible.}$$

As mentioned in introduction, [13] shows an impossibility result between (unconditional) privacy and verifiability. [13] considers another aspect of verifiability, namely universal verifiability, that is, the guarantee that the result corresponds to the content of the ballot, even in presence of a dishonest tally. Interestingly, the same incompatibility result holds between individual verifiability and unconditional privacy, for the same reasons. Exactly like in [13], a powerful adversary (*i.e.* not polynomial) could tally BB and BB′ where BB′ is the ballot box from which Alice's ballot has been removed and infer Alice's vote by difference. More generally, unconditional privacy is lost as soon as there exists a tally function that is meaningfully related to the result, which is implied by individual verifiability.

4.2.2 Privacy. For privacy, we consider the old, well established definition of Josh Benaloh [6]. More sophisticated definitions are been proposed later (see [7] for a survey and a unifying definition). They aim in particular at getting rid of the partial tally assumption (needed in [6]). Note however that they all assume an honest ballot box. Since we also assume partial tally, the original Benaloh definition is sufficient for our needs. In particular, we do not know if privacy implies verifiability for counting functions that do not have the partial tally property. This is left as future work.

Intuitively, a voting system is private if, no matter how honest voters vote, the adversary cannot see any difference. However, the adversary always sees the election result, that leaks how the group of honest voters voted (altogether). Therefore, the election result *w.r.t.* the honest voters has to remain the same. More formally, in a first step, the adversary uses oracles $\mathcal{O}_{\text{reg}}(id)$ and $\mathcal{O}_{\text{corr}}(id)$ to respectively register a voter and get her credential. Then the adversary may request an honest voter id to vote either for v_0 or v_1 through oracle $\mathcal{O}^{p}_{\text{vote}}(id, v_0, v_1)$. Voter id will vote v_β depending on the bit β. The adversary may also cast an arbitrary ballot b in the name of a dishonest voter id through oracle $\mathcal{O}_{\text{cast}}(id, b)$. The election will be tallied, only if the set V_0 of votes v_0 yields the same result than the set V_1 of votes v_1 (where only the last vote is counted). Finally, the adversary wins if he correctly guesses β. Formally, privacy is defined through the game $\mathsf{Exp}^{\mathsf{priv},\beta}_{\mathcal{A}}(\lambda)$ displayed on Figure 5.

Definition 4.3 (Privacy [6]). A voting system is private if for any adversary $\mathcal{A}$,

$$\left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{A}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{A}}(\lambda) = 1\right]\right| \text{ is negligible.}$$

4.3 Privacy implies individual verifiability

We show that privacy implies individual verifiability and we first list here our assumptions. As for the symbolic case, we assume the existence of a neutral vote. We also require that the tally can be performed piecewise, that is, informally, as soon as two boards BB_1, BB_2 are independant then $\mathsf{Tally}(\mathsf{BB}_1 \uplus \mathsf{BB}_2) = \mathsf{Tally}(\mathsf{BB}_1) * \mathsf{Tally}(\mathsf{BB}_2)$. This property is satisfied by most voting schemes. Formally, we characterize this notion of "independence" depending on whether a scheme is id-based or cred-based.

An id-based voting scheme has the *piecewise tally property* if for any two boards BB_1 and BB_2 that contain ballots registered for

$\underline{\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda)}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda})$
$\mathsf{U}, \mathsf{CU} \leftarrow []$
$state \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{reg}}, \mathcal{O}_{\mathsf{corr}}}(\mathsf{pk})$
$\mathsf{BB}, \mathsf{Voted} \leftarrow []$
$\mathcal{A}_2^{\mathcal{O}^{v}_{\mathsf{vote}}, \mathcal{O}_{\mathsf{cast}}}(state, \mathsf{pk})$
$r \leftarrow \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U})$
if $r \neq \bot \ \wedge \ \forall V_c$ (finite) . $r \neq \rho(\{v_i\}_{1 \leq i \leq k} \uplus V_c)$ **then**
  return 1
where $\mathsf{Voted} = \{(id_1, v_1), \dots, (id_k, v_k)\}$

$\underline{\mathcal{O}^{v}_{\mathsf{vote}}(id, v)}$

if $(id, *) \in \mathsf{U} \backslash \mathsf{CU}$ **then**
  $b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v)$
  $\mathsf{BB} \leftarrow \mathsf{BB} \| b$
  $\mathsf{Voted} \leftarrow \mathsf{Voted}' \| (id, cred_{id}, v)$
  return b
    where $(id, cred_{id}) \in \mathsf{U}$
    and Voted' is obtained from Voted
    by removing all previous instances of $(id, *)$

$\underline{\mathcal{O}_{\mathsf{cast}}(id, b)}$

if $(id, *) \in \mathsf{CU} \ \wedge$
  $\mathsf{Valid}(id, b, \mathsf{BB}, \mathsf{pk})$
 then
  $\mathsf{BB} \leftarrow \mathsf{BB} \| (id, b)$

Figure 4: Verifiability

$\underline{\mathsf{Exp}^{\mathsf{priv}, \beta}_{\mathcal{A}}(\lambda)}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda})$
$\mathsf{U}, \mathsf{CU} \leftarrow []$
$state_1 \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{reg}}, \mathcal{O}_{\mathsf{corr}}}(\mathsf{pk})$
$\mathsf{BB}, \mathsf{V}_0, \mathsf{V}_1 \leftarrow []$
$state_2 \leftarrow \mathcal{A}_2^{\mathcal{O}^{p}_{\mathsf{vote}}, \mathcal{O}_{\mathsf{cast}}}(state_1, \mathsf{pk})$
if $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$ **then**
  $r \leftarrow \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U})$
  $\beta' \leftarrow \mathcal{A}_3(state_2, \mathsf{pk}, r)$
  return β'

$\underline{\mathcal{O}^{p}_{\mathsf{vote}}(id, v_0, v_1)}$

if $(id, *) \in \mathsf{U} \backslash \mathsf{CU}$ **then**
  $b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v_{\beta})$
  $\mathsf{BB} \leftarrow \mathsf{BB} \| b$
  $\mathsf{V}_0 \leftarrow \mathsf{V}'_0 \| (id, v_0)$
  $\mathsf{V}_1 \leftarrow \mathsf{V}'_1 \| (id, v_1)$
  return b
    where $(id, cred_{id}) \in \mathsf{U}$
    and V'_0 (resp. V'_1) is obtained from V_0 (resp. V_1)
    by removing all instances of $(id, *)$

$\underline{\mathcal{O}_{\mathsf{cast}}(id, b)}$

if $(id, *) \in \mathsf{CU} \ \wedge$
  $\mathsf{Valid}(id, b, \mathsf{BB}, \mathsf{pk})$
 then
  $\mathsf{BB} \leftarrow \mathsf{BB} \| (id, b)$

Figure 5: Privacy

different agents and such that $\mathsf{BB}_1 \uplus \mathsf{BB}_2$ is valid, that is, if

$$\mathsf{ValidTally}(\mathsf{BB}_1 \uplus \mathsf{BB}_2, \mathsf{sk}, \mathsf{U}) \wedge \\ \forall b \in \mathsf{BB}_1.\ \forall b' \in \mathsf{BB}_2.\ \mathsf{open}_{id}(b) \neq \mathsf{open}_{id}(b'),$$

then their tally can be computed separately:

$$\mathsf{Tally}(\mathsf{BB}_1 \uplus \mathsf{BB}_2, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}_1, \mathsf{sk}, \mathsf{U}) * \mathsf{Tally}(\mathsf{BB}_2, \mathsf{sk}, \mathsf{U}). \quad (*)$$

We also assume that the tally only counts ballots cast with registered ids, *i.e.* $\forall \mathsf{BB}, \mathsf{sk}, \mathsf{U}.\ \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}', \mathsf{sk}, \mathsf{U})$ where $\mathsf{BB}' = [b \in \mathsf{BB} \mid (\mathsf{open}_{id}(b), *) \in \mathsf{U}]$; and that registering more voters does not change the tally: if U, U' have no id in common and $\forall b \in \mathsf{BB}.\ (\mathsf{open}_{id}(b), *) \notin \mathsf{U}'$, then $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U} \cup \mathsf{U}')$.

Similarly, a cred-based voting scheme has the *piecewise tally property* if for any two boards BB_1 and BB_2 that contain ballots associated to different credentials, that is

$$\forall b \in \mathsf{BB}_1.\ \forall b' \in \mathsf{BB}_2.\ \mathsf{open}_{cred}(b, \mathsf{sk}, \mathsf{U}) \neq \mathsf{open}_{cred}(b', \mathsf{sk}, \mathsf{U})$$

then their tally can be computed separately (Property (*)).

We also assume that registering more voters does not change the tally: if U, U' share no credentials and $\forall b \in \mathsf{BB}.\ (*, \mathsf{open}_{cred}(b, \mathsf{sk}, \mathsf{U} \cup \mathsf{U}')) \notin \mathsf{U}'$, then $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U} \cup \mathsf{U}')$.

We say that a (id-based) voting scheme is *strongly correct* if whatever valid board the adversary may produce, adding a honestly generated ballot still yields a valid board. This property is formally defined through the game $\mathsf{Exp}^{\mathsf{ValidTally}}_{\mathcal{A}}(\lambda)$ displayed in Figure 6. A similar assumption was introduced in [7]. For example, Helios is strongly correct.

$\underline{\mathsf{Exp}^{\mathsf{ValidTally}}_{\mathcal{A}}(\lambda)}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda})$
$\mathsf{U}, \mathsf{CU} \leftarrow []$
$state \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{reg}}, \mathcal{O}_{\mathsf{corr}}}(\mathsf{pk})$
$(\mathsf{BB}, id, v) \leftarrow \mathcal{A}_2^{\mathcal{O}^{vt}_{\mathsf{vote}}}(state, \mathsf{pk})$
$b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v)$
    where $(id, cred_{id}) \in \mathsf{U}$
if $(id, *) \in \mathsf{U} \backslash \mathsf{CU} \ \wedge$
  $(\forall b' \in \mathsf{BB}.\ \mathsf{open}_{id}(b') \neq id) \ \wedge$
  $\mathsf{ValidTally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) \ \wedge$
  $\neg \mathsf{ValidTally}(\mathsf{BB} || b, \mathsf{sk}, \mathsf{U})$ **then**
  return 1

$\underline{\mathcal{O}^{vt}_{\mathsf{vote}}(id, v)}$

if $\exists cred_{id}.(id, cred_{id}) \in \mathsf{U} \backslash \mathsf{CU}$ **then**
  return $\mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v)$

Figure 6: $\mathsf{ValidTally}$ **game**

A voter credential typically includes a private part used to generate a signing key for example. It should not be possible for an adversary to forge a ballot with an honest credential. Formally, we say that a voting scheme has *non-malleable credentials*, if for any

$\mathsf{Exp}^{\mathsf{NM}}_{\mathcal{A}}(\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{U}, \mathsf{CU} \leftarrow []$
$state \leftarrow \mathcal{A}_1^{O_{\mathrm{reg}}, O_{\mathrm{corr}}}(\mathsf{pk})$
$\mathsf{L} \leftarrow []$
$b \leftarrow \mathcal{A}_2^{O_{\mathrm{c}}}(state, \mathsf{pk})$
if $b \notin \mathsf{L} \wedge$
$\quad \exists (id, cred_{id}) \in \mathsf{U} \backslash \mathsf{CU}.$
$\quad \mathsf{open}_{cred}(b, \mathsf{sk}, U) = cred_{id}$ **then**
$\quad$ **return** 1

$O_{\mathrm{c}}(id, v)$

if $(id, *) \in \mathsf{U} \backslash \mathsf{CU}$ **then**
$\quad b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v)$
$\quad \mathsf{L} \leftarrow \mathsf{L} \| b$
$\quad$ **return** b
$\quad\quad$ where $(id, cred_{id}) \in \mathsf{U}$

Figure 7: Credential non-malleability

$\mathsf{Exp}^{\mathsf{dis}, \beta}_{\mathcal{A}}(\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{U}, \mathsf{CU} \leftarrow []$
$state_1 \leftarrow \mathcal{A}_1^{O_{\mathrm{reg}}, O_{\mathrm{corr}}}(\mathsf{pk})$
$\mathsf{V}_0, \mathsf{V}_1 \leftarrow []$
$state_2, \mathsf{BB} \leftarrow \mathcal{A}_2^{O^d_{\mathrm{vote}}}(state_1, \mathsf{pk})$
if $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$ **then**
$\quad r \leftarrow \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U})$
$\quad \beta' \leftarrow \mathcal{A}_3(state_2, \mathsf{pk}, r)$
$\quad$ **return** β'

$O^d_{\mathrm{vote}}(id, v_0, v_1)$

if $(id, *) \in \mathsf{U} \backslash \mathsf{CU}$ **then**
$\quad b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v_\beta)$
$\quad \mathsf{V}_0 \leftarrow \mathsf{V}'_0 \| (id, v_0)$
$\quad \mathsf{V}_1 \leftarrow \mathsf{V}'_1 \| (id, v_1)$
$\quad$ **return** b
$\quad\quad$ where $(id, cred_{id}) \in \mathsf{U}$
$\quad\quad$ and V'_0 (resp. V'_1)
$\quad\quad$ is obtained from V_0 (resp. V_1)
$\quad\quad$ by removing all instances
$\quad\quad$ of $(id, *)$

Figure 8: Privacy against a dishonest board (PrivDis-Naive)

adversary $\mathcal{A}$,

$$\mathrm{P}\left[\mathsf{Exp}^{\mathsf{NM}}_{\mathcal{A}}(\lambda) = 1\right] \text{ is negligible}$$

where $\mathsf{Exp}^{\mathsf{NM}}_{\mathcal{A}}(\lambda)$ is defined on Figure 7. For example, Belenios and Civitas have non-malleable credentials.

Theorem 4.4 (Privacy implies individual verifiability). *Let V be an id-based, strongly correct, voting scheme that has the piecewise tally property. If V is private, then V is individually verifiable.*

Similarly, let V be a cred-based voting scheme that has the piecewise tally property and non-malleable credentials. If V is private, then V is individually verifiable.

The proof of this theorem is inspired by the same intuition as in the symbolic case: if an attacker manages to break verifiability, that is, to obtain that not all votes from the honest voters are counted correctly, then there also exists an attack against privacy. Indeed, consider a scenario with additional, new voters, whose votes should compensate those cast by the initial voters. By performing the attack on verifiability for the initial voters, the attacker reaches a state where, in the result of the election, they are no longer compensated by the new votes. This allows the attacker to break privacy.

More precisely, the general idea of the proof is as follows. Consider an attacker $\mathcal{A}$ that breaks individual verifiability, *i.e.* wins the game $\mathsf{Exp}^{\mathsf{verif}}$ with non negligible probability. We construct an attacker $\mathcal{B}$ that breaks privacy, *i.e.* wins $\mathsf{Exp}^{\mathsf{priv}, \beta}$. $\mathcal{B}$ starts by registering, and corrupting, the same voters as $\mathcal{A}$, using oracles O_{reg} and O_{corr}. Let $id_1, \dots, id_n$ be this first set of voters. $\mathcal{B}$ then registers another set of n voters $id'_1, \dots, id'_n$, where the id'_i are fresh identities, that $\mathcal{A}$ does not use.

$\mathcal{B}$ then simulates $\mathcal{A}$, using the oracle O^p_{vote} to simulate $\mathcal{A}$'s calls to O^v_{vote}. Specifically, when $\mathcal{A}$ calls $O^v_{\mathrm{vote}}(id, v)$, $\mathcal{B}$ calls the oracle $O^p_{\mathrm{vote}}(id, v, v^{\mathrm{blank}})$, where v^{blank} is a neutral vote. Once $\mathcal{B}$ is done simulating $\mathcal{A}$, it triggers the new voters id'_i to vote, by calling the oracle $O^p_{\mathrm{vote}}(id'_i, v^{\mathrm{blank}}, v_i)$, where v_i is the (last) vote cast by id_i. The vote of each id'_i compensates the vote of id_i, so that the condition $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$ from $\mathsf{Exp}^{\mathsf{priv}}$ holds. $\mathcal{B}$ then obtains the result r of the election, which is equal to $r_1 * r_2$, where r_1 is the tally of the ballots cast by $\mathcal{A}$, and r_2 the tally of the additional ballots cast by $\mathcal{B}$. Then:

- if $\beta = 0$: then all the votes cast by the id'_i were v^{blank}, and the result is thus $r = r_1$. Since $\mathcal{A}$ breaks individual verifiability, r_1 does not contain the honest votes, *i.e.*, for all multiset V_c of votes, $r \neq \rho(v_1, \dots, v_n) * \rho(V_c)$.
- if $\beta = 1$ however, the votes cast by the id'_i were the v_i, and the partial tally r_2 is therefore $r_2 = \rho(v_1, \dots, v_n)$. Hence, the result r does contain the honest votes.

Therefore, by observing whether the final result of the election contains the honest votes, $\mathcal{B}$ is able to guess β, and wins $\mathsf{Exp}^{\mathsf{priv}}$.

5 PRIVACY WITH A DISHONEST BOARD

Our main theorem states that privacy implies individual verifiability. However, the privacy definition introduced by Benaloh assumes an honest ballot box, as most existing privacy definitions of the literature [7]. Therefore, our main theorem shows that whenever a voting scheme is private *w.r.t.* an honest ballot box, then it is also individually verifiable *w.r.t. an honest ballot box*, which is of course a rather weak property. However, intuitively, our proof technique does not rely on the trust assumptions.

As pointed out in introduction, extending cryptographic privacy definitions to a dishonest ballot box is difficult. Consider the natural extension of privacy as displayed in Figure 8: the game is the same than $\mathsf{Exp}^{\mathsf{priv}, \beta}_{\mathcal{A}}(\lambda)$ except that the adversary arbitrarily controls the ballot box. Unfortunately, an adversary can always win this new game. Indeed, he may simply query $O^d_{\mathrm{vote}}(id_1, 0, 1)$ and $O^d_{\mathrm{vote}}(id_2, 1, 0)$, yielding respectively ballots b_{id_1} and b_{id_2}. Then the adversary choses $\mathsf{BB} = b_{id_1}$. The tally will return β, hence the adversary wins. This corresponds to the fact that an adversary may always isolate a voter and break her privacy.

5.1 Privacy with careful voters

To solve this issue, we choose another approach, which consists in explicitly modelling the verification steps made by voters: the tally will be performed only if honest voters have successfully run their checks (*e.g.* checking that their ballot belongs to the bulletin board). Therefore, we extend the privacy game as follows. The adversary arbitrarily controls the ballot box and may request honest voters to vote through $O^{p,c}_{\mathrm{vote}}(id, v_0, v_1)$ as before. Note that there is no need for the O_{cast} oracle since the adversary may add directly his own

$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},\beta}(\lambda)}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{U}, \mathsf{CU} \leftarrow []$
$state_1 \leftarrow \mathcal{A}_1^{O_{\mathsf{reg}}, O_{\mathsf{corr}}}(\mathsf{pk})$
$\mathsf{V}_0, \mathsf{V}_1, \mathsf{L}_{id}$ (for all id in U) $\leftarrow []$
$state_2, \mathsf{BB} \leftarrow \mathcal{A}_2^{O_{\mathsf{vote}}^{p,c}}(state_1, \mathsf{pk})$
$\mathsf{H} \leftarrow []$
$state_3 \leftarrow \mathcal{A}_3^{O_{\mathsf{happy\,BB}}}(state_2)$
if $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \Rightarrow id \in \mathsf{H}$
  and $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$ **then**
  $r \leftarrow \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U})$
else
  $r \leftarrow \perp$
$\beta' \leftarrow \mathcal{A}_4(state_3, \mathsf{pk}, r)$
return β'

$\underline{O_{\mathsf{vote}}^{p,c}(id, v_0, v_1)}$

if $(id, *) \in \mathsf{U} \backslash \mathsf{CU}$ **then**
  $b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v_\beta)$
  $\mathsf{V}_0 \leftarrow \mathsf{V}_0' \| (id, v_0)$
  $\mathsf{V}_1 \leftarrow \mathsf{V}_1' \| (id, v_1)$
  $\mathsf{L}_{id} \leftarrow \mathsf{L}_{id} \| (b, v_\beta)$
  return b
    where $(id, cred_{id}) \in \mathsf{U}$
    and V_0' (resp. V_1') is obtained
    from V_0 (resp. V_1) by
    removing all instances
    of $(id, *)$

$\underline{O_{\mathsf{happy\,BB}}(id)}$

if $(id, cred_{id}) \in \mathsf{U} \backslash \mathsf{CU}$ **then**
  if $\mathsf{VerifVoter}(id, cred_{id}, \mathsf{L}_{id}, \mathsf{BB})$
  then $\mathsf{H} \leftarrow \mathsf{H} \| id$

Figure 9: Privacy game against a dishonest board with careful voters (Priv-careful)

ballots in the ballot box. He triggers voters to run their verification tests through the oracle $O_{\mathsf{happy\,BB}}(id)$. To run her verification test (using algorithm VerifVoter), the voter has access to the ballot box BB forged by the adversary as well as her local state L_{id} that contains in particular her previously generated ballots. The tally is performed only if all honest voters have successfully performed their test and if, as previously, the set V_0 of left votes v_0 yields the same result than the set V_1 of right votes v_1. Formally, privacy with careful voters is defined through the game $\mathsf{Exp}^{\mathsf{priv-careful}}$ displayed on Figure 9.

Definition 5.1 (Privacy with careful voters). A voting system is *private against a dishonest board with careful voters* if for any adversary $\mathcal{A}$,

$$\left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},1}(\lambda) = 1\right]\right|$$

is negligible.

While this definition models a dishonest ballot box, it implicitly assumes that all voters see the *same* (possibly dishonest) ballot box. This is a very common assumption in voting, that needs to be achieved by external means.

Similarly, we extend individual verifiability to *individual verifiability against a dishonest board* as expected, assuming that the tally is performed only if all honest voters have successfully performed their test. The formal definition of individual verifiability against a dishonest board can be found in appendix.

5.2 Privacy implies individual verifiability against a dishonest box too

We need to assume that the verification test run by honest voters (VerifVoter) is consistent with how the voter voted. Namely, if the voter's intended ballot is the one that is selected from the board by the revote policy (*e.g.* appears last *w.r.t.* this voter), then this voter must be satisfied with the board (that is, VerifVoter passes). Conversely, if the test VerifVoter fails for voter id then adding ballots unrelated to id (or her credential) will not change this fact (VerifVoter will still fail). These assumptions are formally stated in appendix.

$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{BS},\beta}(\lambda)}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{U}, \mathsf{CU} \leftarrow []$
$state_1 \leftarrow \mathcal{A}_1^{O_{\mathsf{reg}}, O_{\mathsf{corr}}}(\mathsf{pk})$
$\mathsf{L} \leftarrow []$
$state_2, \mathsf{BB} \leftarrow \mathcal{A}_2^{O_{\mathsf{vote}}^{bs}}(state_1, \mathsf{pk})$
if $\forall v.$
$|\{b | b \in \mathsf{BB} \wedge \exists v'.\ (b, v, v') \in \mathsf{L}\}| =$
$|\{b | b \in \mathsf{BB} \wedge \exists v'.\ (b, v', v) \in \mathsf{L}\}|$
then
  $r \leftarrow \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U})$
  $\beta' \leftarrow \mathcal{A}_3(state_2, \mathsf{pk}, r)$
  return β'

$\underline{O_{\mathsf{vote}}^{bs}(id, v_0, v_1)}$

if $(id, *) \in \mathsf{U} \backslash \mathsf{CU}$ **then**
  $b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v_\beta)$
  $\mathsf{L} \leftarrow \mathsf{L} \| (b, v_0, v_1)$
  return b
    where $(id, cred_{id}) \in \mathsf{U}$

Figure 10: PrivacyBS [9]

Theorem 5.2 (Privacy implies individual verifiability against a dishonest board). *Let V be an id-based, strongly correct voting scheme that has the piecewise tally property. If V is private against a dishonest board with careful voters, then V is individually verifiable against a dishonest board with careful voters.*

Similarly, let V be a cred-based voting scheme that has the piecewise tally property and non-malleable credentials. If V is private against a dishonest board with careful voters, then V is individually verifiable against a dishonest board with careful voters.

6 COMPARING PRIVACY

We compare different notions of privacy, with and without an honest ballot box, on four standard protocols (Helios, Belenios, Civitas, and Neuchâtel) as well as on our simple protocol, sketched in introduction.

To our knowledge, the only other definition of privacy with a dishonest ballot box is the privacy notion introduced by Bernhard and Smyth [9]. We first start by discussing this definition.

6.1 PrivacyBS

The privacy notion introduced by Bernhard and Smyth [9] is recalled in Figure 10 (PrivacyBS). The adversary may request a voter id to vote for v_0 or v_1 (depending on the bit β) through the oracle $O_{\mathsf{vote}}^{bs}(id, v_0, v_1)$. He produces an arbitrary ballot box BB and the tally will be performed provided that, looking at honest ballots that appear in BB, counting the corresponding left and right votes yields the same result.

The main interest of [9] is to highlight the fact that previous definitions implicitly assume an honest ballot box. The first attempt at defining privacy *w.r.t.* a dishonest ballot box (PrivacyBS) has several limitations. First, it strongly assumes that the ballots that appear in the ballot box are exactly the same than the cast ballots. This is not the case for example of the ThreeBallots protocol [28] where the ballot box only contains two shares (out of three) of the original ballot. It is not applicable either to a protocol like BeleniosRF [12] where ballots are re-randomised before their publication. Second, it requires ballots to be non-malleable [9]. This means that, as soon as a ballot includes a malleable part (for example the voter's id like in Helios, or a timestamp), privacy cannot be satisfied. This severely restricts the class of protocols that can be considered. Third, PrivacyBS does not account for a revote policy. As soon as revote is allowed (for example in Helios), then PrivacyBS is broken since some ballots may not be counted. Indeed, an attacker may call $\mathcal{O}^{bs}_{\mathsf{vote}}(id_1, 1, 0)$, followed by $\mathcal{O}^{bs}_{\mathsf{vote}}(id_1, 0, 1)$, obtaining ballots b_1, b'_1, and return the board $\mathsf{BB} = [b_1, b'_1]$. The equality condition on the number of ballots in BB produced by $\mathcal{O}^{bs}_{\mathsf{vote}}$ holds, since for $v = 0, 1$:

$$|\{b \in \mathsf{BB} | \exists v'.\ (b, v, v') \in \mathsf{L}\}| = |\{b \in \mathsf{BB} | \exists v'.\ (b, v', v) \in \mathsf{L}\}| = 1$$

where $\mathsf{L} = [(b_1, 1, 0), (b'_1, 0, 1)]$. Hence the tally is computed. According to the revote policy, only b'_1 is counted, and the result is β, which lets the attacker win $\mathsf{Exp}^{\mathsf{BS}}$.

6.2 Protocols

We consider four standard protocols (Helios, Belenios, Civitas, and Neuchâtel) as well as our simple protocol, presented in introduction. We briefly explain each of them in this section. In what follows $\mathcal{E} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ denotes an encryption algorithm.

Simple. We detail the simple protocol sketched in introduction. Recall that voters simply send their encrypted votes to the ballot box, and, at the end of the voting phase, the tally computes and publishes the result of the election. No revote is allowed, and the voters do not have any means of verifying that their vote is taken into account. Identities and credentials are not used in this protocol. The corresponding algorithms of this protocol are:

- $\mathsf{Vote}(id, cred, \mathsf{pk}, v) = \mathsf{enc}(\mathsf{pk}, v)$
- $\mathsf{VerifVoter}(id, cred, \mathsf{L}, \mathsf{BB}) = \mathbf{true}$ (voters do not make any checks)
- $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U})$ checks that all the ballots in BB are distinct, and returns $\perp$ if not. The tally performs a random permutation of the ballots, decrypts all of them and returns the multiset of the votes they contain.
- $\mathsf{Valid}(id, b, \mathsf{BB}, \mathsf{pk})$ checks that b does not already occur in BB.

Helios [4] is similar to Simple, except that revote is allowed, and the last vote cast by each *id* is counted. To make this revote policy possible, the ballots contain the *id* of the voter: $\mathsf{Vote}(id, cred, \mathsf{pk}, v) = (id, \mathsf{enc}(\mathsf{pk}, v))$. $\mathsf{enc}(\mathsf{pk}, v)$ here also includes a proof that v is a valid vote. Credentials are unused. The tally computes the result of the election similarly to Simple except that it also features an homomorphic mode, where the tally homomorphically computes the sum of the ballots in BB, decrypts the resulting ciphertext and returns the result. Moreover, the tally returns a proof of correct decryption. In addition, the board which will be tallied is made public, allowing the voters to check that their last ballot is indeed the last ballot with their *id* on the board:

$$\begin{aligned}&\mathsf{VerifVoter}(id, cred, \mathsf{L}_{id}, \mathsf{BB}) = \\ &\quad \text{the last element in } \mathsf{L}_{id} \text{ is the last ballot} \\ &\quad \text{registered for } id \text{ in } \mathsf{BB}.\end{aligned}$$

Similarly to Simple, the Valid function checks that there is no duplicated ciphertext and also checks that the ballot is submitted under the right *id*.

$$\mathsf{Valid}(id, (id', c), \mathsf{BB}, \mathsf{pk}) = (id = id') \ \wedge\ c \text{ does not occur in } \mathsf{BB}$$

This models an authenticated channel between the ballot box and each voter: a voter *id* may not cast a vote in the name of id'.

Belenios [15] is similar to Helios, except that voters sign their encrypted vote thanks to their credential:

$$\mathsf{Vote}(id, k, \mathsf{pk}, v) = (id, \mathsf{signElGamal}(v, \mathsf{pk}, k))$$

where $\mathsf{signElGamal}(\cdot, \cdot, \cdot)$ denotes the combination of the (ElGamal) encryption and the signature. As for Helios, it also includes a proof that v is a valid vote. Tally checks that there exists a bijection between the *id*s and the credentials in the final board, *i.e.* that the same *id* is always associated with the same signature, and vice versa. The revote policy counts the last ballot corresponding to a given credential. Voters can verify that their last ballot is indeed the last one signed by their key on the board.

Civitas [14]. In Civitas, voters privately receive a credential, that is published encrypted on the bulletin board. To cast a vote, a voter encrypts her vote, also encrypts her credential, and produces a proof π of well-formedness that links the two ciphertexts together. The corresponding ballot is of the form

$$\mathsf{Vote}(id, cred, \mathsf{pk}, v) = (\mathsf{enc}(\mathsf{pk}, cred), \mathsf{enc}(\mathsf{pk}, v), \pi).$$

The voters can verify that their vote will be taken into account by checking that it is present on the board that will be tallied.

$$\mathsf{VerifVoter}(id, cred, \mathsf{L}_{id}, \mathsf{BB}) = b \in \mathsf{BB}$$

where b is the ballot in L_{id}. In theory, revote is allowed. However, we unveil a small discrepancy in how revote should be performed. Assume for example that the last ballot should be counted. Since an adversary may recast old ballots generated by an honest voter, a voter should memorise all the ballots he generated and check that they appear in the right order on the ballot box. Such a check seems highly cumbersome for an average voter and we could not find its description in [14]. Therefore, we simply assume here that honest voters do not revote.

Neuchâtel [22]. Voters privately receive a code sheet, where each candidate is associated to a (short) code. To cast a vote, voters send their encrypted votes to the ballot box, similarly to Simple or Helios. The ballot box then provides a return code allowing the voter to check that the ballot has been received and that it encrypts their candidate, as intended. This offers a protection against a dishonest voting client (*e.g.* if the voter's computer is corrupted). No revote is allowed. Since the bulletin board is not published, voters cannot check that their ballots really belong to the final board (used for tally), which we model by $\mathsf{VerifVoter}(id, cred, \mathsf{L}, \mathsf{BB}) = \mathbf{true}$. Voters have to trust the voting server (or other internal components) on this aspect.

Protocol	Honest board [6]	PrivDis-Naive	PrivacyBS [9]	Priv-careful
Simple (no revote)	✓	✗	✓	✗
Helios	✓	✗	✗	✗
Belenios	✓	✗	✗	✓
Civitas (no revote)	✓	✗	✓	✓
Neuchâtel (no revote)	✓	✗	✓	✗

Figure 11: Comparison of several privacy definitions
(✓: the protocol is private, ✗: there exists an attack on privacy)

6.3 Attacks

Simple. As described in introduction, a dishonest ballot box may break ballot privacy of any voter by simply replacing the other votes by votes of its choice. In other words, even if the ballot box does not detain any decryption key, it can learn how Alice's voted.

Neuchâtel. Exactly like the Simple protocol, a dishonest ballot box may break ballot privacy of any voter by simply replacing the other votes. This is due to the fact that voters have no control over the ballots that are actually tallied. Note that the Neuchâtel protocol actually includes internal mechanisms that render such an attack difficult. However, from the point of view of a voter, if the ballot box is compromised, her privacy is no longer guaranteed.

Helios. Helios is also vulnerable to an attack when the ballot box is compromised. This attack is due to P. Roenne [29]. It involves two honest voters id_1, id_2, and a dishonest voter id_3. The attacker may call $O^{p,c}_{\mathsf{vote}}(id_1, 0, 1)$ twice and $O^{p,c}_{\mathsf{vote}}(id_2, 1, 0)$ once, obtaining ballots (id_1, b_1), (id_1, b'_1), (id_2, b_2). The adversary then returns the board $[(id_1, b'_1), (id_2, b_2), (id_3, b_1)]$. All ballots are different, hence no weeding is needed. The result of the tally is then $\rho(\{\!\!\{0, 1, 0\}\!\!\})$ if $\beta = 0$ and $\rho(\{\!\!\{1, 0, 1\}\!\!\})$ if $\beta = 1$. The attacker can therefore observe the difference in the result, which breaks privacy.

Belenios and *Civitas* remain private against a dishonest board as long as voters perform their verification checks. We formally prove privacy according to our definition Priv-careful.

6.4 Comparison

We summarise our findings in Figure 11. As explained in Section 5, the naive extension of the privacy definition to a dishonest board (PrivDis-Naive) is immediately false for any protocol.

All of our five protocols satisfy privacy against an honest ballot box. We rely here on previous results of the literature, except for Civitas (and of course the Simple protocol). Indeed, Civitas has been proved to be coercion-resistant [14] in a rather different setting. Therefore we show here that it satisfies the Benaloh definition.

PrivacyBS fails to detect the attack on the Neuchâtel protocol and the Simple protocol since it requires that the tally of the honest ballots present on the final board does not leak information. Conversely, it cannot prove Belenios private as it does not properly handle revoting as explained in Section 6.1.

7 CONCLUSION

We show a subtle relation between privacy and verifiability, namely that privacy implies individual verifiability, which is rather counter-intuitive. Our result holds in a cryptographic as well as a symbolic setting, for various trust assumptions. In contrast, privacy does not seem to imply universal verifiability nor eligibility verifiability. To show that there is indeed no implication, we plan to exhibit counter-examples, as simple as possible.

Our result assumes counting functions that have the partial tally property. Our proof technique does not extend immediately to more complex counting functions such as STV or Condorcet. We plan to study how privacy and individual verifiability are related in this context. Also, our results implicitly discard anonymous channels: computational models do not account for anonymous channels while our election determinism assumption discards at least some use of anonymous channels. Intuitively, in presence of anonymous channels, an attacker may be able to modify a ballot without being able to tell which one, hence breaking verifiability without breaking privacy. It would be interesting to identify which kind of anonymous channels and more generally, which form of non determinism, can still be tolerated.

Our findings also highlight a crucial need for a ballot privacy definition in the context of a dishonest ballot box, in a cryptographic setting. So far, privacy has only been proved assuming an honest ballot box, which forms a very strong trust assumption that was probably never made clear to voters nor election authorities.

We propose a first attempt at modelling privacy against a dishonest board, assuming that honest voters checks their ballots as expected by the voting protocol. We do not see our definition as final. In particular, assuming that *all* voters check their vote is highly unrealistic. In a realistic setting, it is more likely that a (small) fraction of honest voters perform the required tests while the others stop after casting their vote. We plan to explore how to adapt our definition to a quantitative setting, in the lines of [27].

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers as well as the shepherd, whose helpful comments and suggestions greatly contributed to clarify the notions presented in the paper.

This work has been partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research (grant agreement No 645865-SPOOC).

REFERENCES

[1] 2010. Délibération n° 2010-371 du 21 octobre 2010 portant adoption d'une recommandation relative à la sécurité des systèmes de vote électronique. French

National recommendation on e-voting.
[2] 2013. Ordonnance de la ChF sur le vote électronique (OVotE) du 13 décembre 2013 (Etat le 15 janvier 2014). Chancellerie fédérale ChF. Swiss recommendation on e-voting.
[3] Martín Abadi and Cédric Fournet. 2001. Mobile Values, New Names, and Secure Communication. In *28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'01)*. ACM, 104–115. https://doi.org/10.1145/360204.360213
[4] Ben Adida. 2008. Helios: Web-based Open-Audit Voting. In *17th USENIX Security Symposium (Usenix'08)*. 335–348.
[5] Michael Backes, Catalin Hritcu, and Matteo Maffei. 2008. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-Calculus. In *21st IEEE Computer Security Foundations Symposium, (CSF 2008)*. 195–209.
[6] J. Benaloh. 1987. *Verifiable secret-ballot elections.* Ph.D. Dissertation. Yale University.
[7] David Bernhard, Veronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. 2015. A comprehensive analysis of game-based ballot privacy definitions. In *Proceedings of the 36th IEEE Symposium on Security and Privacy (S&P'15)*. IEEE Computer Society Press, 499–516.
[8] David Bernhard, Olivier Pereira, and Bogdan Warinschi. 2012. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In *Advances in Cryptology - ASIACRYPT 2012 (LNCS)*, Vol. 7658. Springer, 626–643.
[9] David Bernhard and Ben Smyth. 2014. Ballot secrecy with malicious bulletin boards. Cryptology ePrint Archive, Report 2014/822.
[10] Bruno Blanchet. 2016. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. *Foundations and Trends in Privacy and Security* 1, 1–2 (Oct. 2016), 1–135.
[11] Ian Brightwell, Jordi Cucurull, David Galindo, and Sandra Guasch. 2015. An overview of the iVote 2015 voting system. Available at https://www.elections.nsw.gov.au.
[12] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. 2016. BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme. In *23rd ACM Conference on Computer and Communications Security (CCS'16)*. Vienna, Austria, 1614–1625.
[13] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. 2010. On Some Incompatible Properties of Voting Schemes. In *Towards Trustworthy Elections 2010*. 191–199.
[14] M. R. Clarkson, S. Chong, and A. C. Myers. 2008. Civitas: Toward a Secure Voting System. In *IEEE Symposium on Security and Privacy (S&P'08)*. IEEE Computer Society, 354–368.
[15] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachene. 2014. Election Verifiability for Helios under Weaker Trust Assumptions. In *19th European Symposium on Research in Computer Security (ESORICS'14) (LNCS)*, Vol. 8713. Springer, 327–344.
[16] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. 2016. SoK: Verifiability Notions for E-Voting Protocols. In *36th IEEE Symposium on Security and Privacy (S&P'16)*. San Jose, USA, 779–798.
[17] Véronique Cortier, Niklas Grimm, Joseph Lallemand, and Matteo Maffei. 2017. A type system for privacy properties. In *24th ACM Conference on Computer and Communications Security (CCS'17)*. ACM, Dallas, USA, 409–423.
[18] Véronique Cortier and Ben Smyth. 2013. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security* 21, 1 (2013), 89–148.
[19] Véronique Cortier and Cyrille Wiedling. 2012. A formal analysis of the Norwegian E-voting protocol. In *Proceedings of the 1st International Conference on Principles of Security and Trust (POST'12) (Lecture Notes in Computer Science)*, Vol. 7215. Springer, 109–128.
[20] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. 2006. Coercion-Resistance and Receipt-Freeness in Electronic Voting. In *19th IEEE Computer Security Foundations Workshop (CSFW'06)*. IEEE Computer Society Press, Venice, Italy, 28–39.
[21] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. 2009. Verifying Privacy-type Properties of Electronic Voting Protocols. *Journal of Computer Security* 17, 4 (2009), 435–487. https://doi.org/10.3233/JCS-2009-0340
[22] David Galindo, Sandra Guasch, and Jordi Puiggali. 2015. 2015 Neuchâtel's Cast-as-Intended Verification Mechanism. In *5th International Conference on E-Voting and Identity, (VoteID'15)*. 3–18.
[23] Rop Gonggrijp and Willem-Jan Hengeveld. 2007. Studying the Nedap/Groenendaal ES3B Voting Computer: A Computer Security Perspective. In *USENIX Workshop on Accurate Electronic Voting Technology (EVT'07)*.
[24] Sven Heiberg, Tarvi Martens, Priit Vinkel, and Jan Willemson. 2017. Improving the Verifiability of the Estonian Internet Voting Scheme. In *E-Vote-ID 2016 (LNCS)*, Vol. 10141. Springer, 92–107.
[25] Steve Kremer, Mark D. Ryan, and Ben Smyth. 2010. Election verifiability in electronic voting protocols. In *15th European Symposium on Research in Computer Security (ESORICS'10) (LNCS)*, Vol. 6345. Springer. https://doi.org/10.1007/978-3-642-15497-3_24
[26] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. 2010. Accountabiliy: Definition and Relationship to Verifiability. In *17th ACM Conference on Computer and Communications Security (CCS'10)*. 526–535.
[27] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. 2011. Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In *32nd IEEE Symposium on Security and Privacy (S&P 2011)*. IEEE Computer Society, 538–553.
[28] R. L. Rivest and W. D. Smith. 2007. Three Voting Protocols: ThreeBallot, VAV and Twin. In *USENIX/ACCURATE Electronic Voting Technology (EVT 2007)*.
[29] Peter Roenne. [n. d.]. Private communication. ([n. d.]). The attack has been discovered by Peter Roenne and then described in the paper [17].
[30] Peter Ryan. 2008. Prêt à Voter with Paillier encryption. *Mathematical and Computer Modelling* 48, 9–10 (2008), 1646–1662.
[31] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. 2014. Security Analysis of the Estonian Internet Voting System. In *2014 ACM SIGSAC Conference on Computer and Communications Security*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM, 703–715.
[32] Scott Wolchok, Eric Wustrow, J. Alex Halderman, Hari K. Prasad, Arun Kankipati, Sai Krishna Sakhamuri, Vasavya Yagati, and Rop Gonggrijp. 2010. Security Analysis of India's Electronic Voting Machines. In *17th ACM Conference on Computer and Communications Security (CCS'10)*. Chicago, IL.
[33] Scott Wolchok, Eric Wustrow, Dawn Isabel, and J. Alex Halderman. 2012. Attacking the Washington, D.C. Internet Voting System. In *Financial Cryptography and Data Security (FC'12)*.

Appendix A SYMBOLIC PROOF

A.1 Assumptions summary

We simply recall here the assumptions described in the core of the paper (Sections 2 and 3.4). The numbers will be useful to refer to the assumptions in the proofs. We also formally define alternative assumptions only sketched in the core of the paper, like the possibility to assume a special "independent" vote instead of a neutral vote.

Notations: if $k \in \mathbb{N}$ and $v \in \mathcal{V}$, $k \cdot v$ denotes the multiset containing k instances of v. If V is a multiset of votes, and v a vote, we denote $V(v)$ the number of instances of v in V.

(1) The tallying process is assumed to output terms representing the result of the election on a channel c_r, *i.e.*

$$\forall \alpha.\ \forall (tr, \phi) \in \mathsf{trace}(P_\alpha).\ \mathsf{out}(c_r, r) \in tr \implies \exists V.\ \phi(r) \in R(\rho(V)).$$

(2) The representation function is assumed to be injective: $\forall r \neq r'.\ R(r) \cap R(r') = \emptyset$.
(3) The counting function is assumed to have the *partial tally* property: $\forall V, V'.\ \rho(V \uplus V') = \rho(V) * \rho(V')$. $*$ is an associative, commutative operation.
(4) The voting processes must be *election determinate*, *i.e.* satisfy

$$\begin{aligned}&\forall \alpha, t, t', \phi, \phi', x, V.\\ &\quad (t =_\tau t' \ \wedge\ (t.\mathsf{out}(c_r, x), \phi) \in \mathsf{trace}(P_\alpha) \ \wedge\ (t'.\mathsf{out}(c_r, x), \phi') \in \mathsf{trace}(P_\alpha) \ \wedge\\ &\quad\ \phi(x) \in R(\rho(V))) \Rightarrow \phi'(x) \in R(\rho(V))\end{aligned}$$

(5) The voting processes are assumed to be *voting friendly*, *i.e.* we assume that for all voter $a \in \mathcal{A}$, there exists t'' such that for all α satisfying $a \notin \mathsf{dom}(\alpha)$,
   - for all $(t, \phi) \in \mathsf{trace}(P_\alpha)$, such that $t = t'.\mathsf{out}(c_r, x)$ for some t', x, for all v, there exists tr, ψ such that $tr =_\tau t''$, $\mathsf{Voted}(a, v) \in tr$, $(t'.tr.\mathsf{out}(c_r, x), \psi) \in \mathsf{trace}(P_{\alpha \cup \{a \mapsto v\}})$, and $\forall V.\ \phi(x) \in R(\rho(V)) \Rightarrow \psi(x) \in R(\rho(V \uplus \{\!|v|\!\}))$.
   - and for all t', x such that $\mathsf{blocking}(t'.\mathsf{out}(c_r, x), P_\alpha)$, for all v, tr, ψ such that $tr =_\tau t''$, $\mathsf{blocking}(t'.tr.\mathsf{out}(c_r, x), P_{\alpha \cup \{a \mapsto v\}})$.

Our result holds provided one of the two following assumptions is true:

(6) There exists a neutral vote $v_{\mathsf{neutral}} \in \mathcal{V}$, such that $\rho(\{\!|v_{\mathsf{neutral}}|\!\})$ is neutral for $*$.
(7) There exists a special vote $v_{\mathsf{special}} \in \mathcal{V}$, which is counted separately in the result, as briefly sketched in the core of the paper. Formally, v_{special} must enjoy the following properties.
   - the result associated with a multiset determines the number of instances of v_{special} in it

$$\forall V, V'.\ \rho(V) = \rho(V') \implies V(v_{\mathsf{special}}) = V'(v_{\mathsf{special}}).$$

   - the count of v_{special} can be simplified

$$\forall V, V', k.\ \rho(V \uplus k \cdot v_{\mathsf{special}}) = \rho(V' \uplus k \cdot v_{\mathsf{special}}) \implies \rho(V) = \rho(V').$$

   For example, for ρ_{hom}, all the votes are special, therefore this property always holds. For ρ_{mix}, it depends on the set of valid votes. In the standard case where a vote is a selection of candidates (for example between k_1 and k_2 candidates), then a special vote is, for instance, a vote that includes the selection of an extra candidate, not used before.

A.1.1 Properties. We recall here the definitions of the individual verifiability and privacy properties (presented in 3.3), and we define two properties used as pivots in the proof.

- Individual verifiability:

$$V_1(t, \phi) \stackrel{\text{def}}{=} \forall t', x.\ (t = t'.\mathsf{out}(c_r, x)) \implies \exists V_c.\ \phi(x) \in R(\rho(\{\!|v \mid \exists a.\ \mathsf{Voted}(a, v) \in t|\!\} \uplus V_c))$$

$$V \stackrel{\text{def}}{=} \forall \alpha.\ \forall (t, \phi) \in \mathsf{trace}(P_\alpha).\ V_1(t, \phi).$$

- Privacy:

$$P = \forall \alpha.\ \forall a, b \in \mathcal{A} \backslash \mathsf{dom}(\alpha).\ \forall v_1, v_2.\ P_{\alpha \cup \{a \mapsto v_1, b \mapsto v_2\}} \approx_t P_{\alpha \cup \{a \mapsto v_2, b \mapsto v_1\}}$$

- First pivot property:

$$\begin{aligned}F \stackrel{\text{def}}{=}\ &\forall \alpha.\ \forall a \in \mathcal{A} \backslash \mathsf{dom}(\alpha).\ \forall v_1, v_2 \in \mathcal{V}.\ \forall t, t', \phi, \phi', V, V'.\\ &\quad [\ t =_\tau t' \ \wedge\\ &\quad (t, \phi) \in \mathsf{trace}(P_{\alpha \cup \{a \mapsto v_1\}}) \ \wedge\\ &\quad (t', \phi') \in \mathsf{trace}(P_{\alpha \cup \{a \mapsto v_2\}}) \ \wedge\\ &\quad \mathsf{result}(t, \phi, V) \ \wedge\ \mathsf{result}(t', \phi', V')\ ] \implies\\ &\quad \rho(V' \uplus \{\!|v_1|\!\}) = \rho(V \uplus \{\!|v_2|\!\}).\end{aligned}$$

- Second pivot property:

$$\begin{aligned}FF \stackrel{\text{def}}{=}\ & \forall \alpha.\ \forall (t,\phi) \in \mathsf{trace}(P_\alpha).\ \forall V, V_{\mathsf{change}}.\ \forall V_{\mathsf{wanted}} \subseteq \{\!\!\{ v \mid \exists a.\ \mathsf{Voted}(a,v) \in t \}\!\!\}.\\ & \mathsf{result}(t,\phi,V) \implies \\ & |V_{\mathsf{change}}| = |V_{\mathsf{wanted}}| \implies \\ & \exists V_c.\ \rho(V) * \rho(V_{\mathsf{change}}) = \rho(V_{\mathsf{wanted}}) * \rho(V_c)\end{aligned}$$

A.2 Theorem

Lemma A.1 (Privacy implies F). *Under assumptions 1, 2, 3, 4, and 5,*

$$P \Rightarrow F$$

Proof. We prove this by contradiction: assuming F does not hold, we construct an attack on privacy.

Assume F is false. Hence there exists a scenario where changing the vote of one agent does not change the result by one. That is to say, there exist an affectation of votes α, an agent $a \notin \mathsf{dom}(\alpha)$, votes $v_1, v_2 \in \mathcal{V}$, traces $(t,\phi) \in \mathsf{trace}(P_{\alpha\cup\{a\mapsto v_1\}})$ and $(t',\phi') \in \mathsf{trace}(P_{\alpha\cup\{a\mapsto v_2\}})$, such that $t =_\tau t'$,and two multisets V, V', such that $\mathsf{result}(t,\phi,V)$, $\mathsf{result}(t',\phi',V')$, and $\rho(V' \uplus \{\!\!\{v_1\}\!\!\}) \neq \rho(V \uplus \{\!\!\{v_2\}\!\!\})$.

Since $\mathsf{result}(t,\phi,V)$, there exist x, t_1 such that $t = t_1.\mathtt{out}(c_r,x)$ and $\phi(x) \in R(\rho(V))$. Similarly, there exist y, t_1' such that $t' = t_1'.\mathtt{out}(c_r,y)$ and $\phi'(y) \in R(\rho(V'))$. Since $t =_\tau t'$, $x = y$.

Note that we necessarily have $v_1 \neq v_2$: indeed, if $v_1 = v_2$ then (t,ϕ) and (t,ϕ') are traces of the same process. Since $\phi(x) \in R(\rho(V))$, by assumption 4, this implies that $\phi'(x) \in R(\rho(V))$. Since we already know that $\phi'(x) \in R(\rho(V'))$, by assumption 2, we have $\rho(V) = \rho(V')$. Thus, as $v_1 = v_2$, we have $\rho(V) * \rho(\{\!\!\{v_2\}\!\!\}) = \rho(V') * \rho(\{\!\!\{v_1\}\!\!\})$, which is contradictory. Hence $v_1 \neq v_2$.

The attack on privacy consists in the fact that, since changing a's vote does not produce a change of exactly one in the result, even in presence of another agent b whose vote is the opposite of a's, the result will be different depending on the vote of a.

Formally, let $b \notin \mathsf{dom}(\alpha) \cup \{a\}$. By assumption 5, there exist sequences of actions t_b, t_b', and frames ψ, ψ', such that

$$\begin{gathered}(t_1.t_b.\mathtt{out}(c_r,x), \psi) \in \mathsf{trace}(P_{\alpha\cup\{a\mapsto v_1, b\mapsto v_2\}}),\\ (t_1'.t_b'.\mathtt{out}(c_r,x), \psi') \in \mathsf{trace}(P_{\alpha\cup\{a\mapsto v_2, b\mapsto v_1\}}),\\ \mathsf{Voted}(b,v_2) \in t_b,\ \mathsf{Voted}(b,v_1) \in t_b',\ t_b =_\tau t_b',\\ \psi(x) \in R(\rho(V \uplus \{\!\!\{v_2\}\!\!\})),\ \text{and}\ \psi'(x) \in R(\rho(V' \uplus \{\!\!\{v_1\}\!\!\})).\end{gathered}$$

Since $\rho(V' \uplus \{\!\!\{v_1\}\!\!\}) \neq \rho(V \uplus \{\!\!\{v_2\}\!\!\})$, by assumption 2, we have $\psi(x) \neq \psi'(x)$.

We have constructed two frames, obtained by the same actions in $P_{\alpha\cup\{a\mapsto v_1, b\mapsto v_2\}}$ and $P_{\alpha\cup\{a\mapsto v_2, b\mapsto v_1\}}$, which yield different results for the election. Using assumption 4, this lets us prove that these two processes are not $\approx_t$-equivalent.

Indeed, let us denote $t_2 \stackrel{\text{def}}{=} t_1.t_b.\mathtt{out}(c_r,x)$ and $t_2' \stackrel{\text{def}}{=} t_1'.t_b'.\mathtt{out}(c_r,x)$. We have $(t_2,\psi) \in \mathsf{trace}(P_{\alpha\cup\{a\mapsto v_1, b\mapsto v_2\}})$. For any trace $(t_2'',\psi'') \in \mathsf{trace}(P_{\alpha\cup\{a\mapsto v_2, b\mapsto v_1\}})$, if $t_2'' =_\tau t_2 =_\tau t_2'$, then by assumption 4 we have $\psi''(x) \in R(\rho(V' \uplus \{\!\!\{v_1\}\!\!\}))$. Hence, by assumption 2 as $\rho(V \uplus \{\!\!\{v_2\}\!\!\}) \neq \rho(V' \uplus \{\!\!\{v_1\}\!\!\})$, we have $\psi''(x) \notin R(\rho(V \uplus \{\!\!\{v_2\}\!\!\}))$, which implies that ψ, ψ'' are not statically equivalent.

Thus, $P_{\alpha\cup\{a\mapsto v_1, b\mapsto v_2\}} \not\approx_t P_{\alpha\cup\{a\mapsto v_2, b\mapsto v_1\}}$. This violates P, which concludes the proof. ☐

Lemma A.2 (Privacy and F imply FF). *Under assumptions 1, 2, 3, 4, and 5,*

$$(P\ \wedge\ F) \Rightarrow FF$$

Proof. Assume that both P and F hold. Let α be an affectation of votes, let $(t,\phi) \in \mathsf{trace}(P_\alpha)$, let V be such that $\mathsf{result}(t,\phi,V)$, *i.e.* there exist t', x such that $t = t'.\mathtt{out}(c_r,x)$ and $\phi(x) \in R(\rho(V))$. Let $V_{\mathsf{wanted}} \subseteq \{\!\!\{ v \mid \exists a.\ \mathsf{Voted}(a,v) \in t\}\!\!\}$, and V_{change} such that $|V_{\mathsf{change}}| = |V_{\mathsf{wanted}}|$.

To prove FF, we need to show that the result in this trace augmented with V_{change} contains at least the subset V_{wanted} of the (intended) votes of the honest voters. That is to say, we must show that there exists V_c such that $\rho(V) * \rho(V_{\mathsf{change}}) = \rho(V_{\mathsf{wanted}}) * \rho(V_c)$.

The idea of the proof is to compare $\rho(V)$ to the result $\rho(V')$ obtained by turning, one by one, all votes from V_{wanted} into the votes from V_{change}, and performing the same sequence of actions. As we will see, this is possible, otherwise P would break; and $V \uplus V_{\mathsf{change}}$ contains more instances of the honest votes than $V' \uplus V_{\mathsf{wanted}}$, since F holds.

Let us denote the Voted events appearing in t by

$$\mathsf{Voted}(a_1,v_1), \ldots, \mathsf{Voted}(a_l,v_l)$$

for some pairwise distinct agents $a_1,\ldots,a_l \in \mathsf{Voters}(t)$, and some $l \in \mathbb{N}$.

By definition, each element of V_{wanted} is associated with one of these Voted events. Let $m \stackrel{\text{def}}{=} |V_{\mathsf{wanted}}|$. Without loss of generality, we may assume that $V_{\mathsf{wanted}} = \{\!\!\{v_1,\ldots,v_m\}\!\!\}$.

Note that $|V_{\mathsf{change}}|$ is also equal to m by assumption. Let us then denote $V_{\mathsf{change}} \stackrel{\text{def}}{=} \{\!\!\{v_1',\ldots,v_m'\}\!\!\}$.

Since, by assumption on the form of the processes, the $\mathsf{Voted}(a,v)$ event can only be emitted by the process $\mathsf{Voter}(a,v,c)$ for some credential c, we have $\alpha(a_i) = v_i$ for all $i \in [\![1,m]\!]$.

For $i \in [\![0, m]\!]$, let α_i denote the affectation of votes obtained from α by turning the first i votes from V_{wanted} to V_{change}, *i.e.*

- $\alpha_i(a_j) = v'_j$ if $j \in [\![1, i]\!]$;
- $\alpha_i(a_j) = v_j$ if $j \in [\![i+1, m]\!]$;
- $\alpha_i(a) = \alpha(a)$ if $a \in \mathrm{dom}(\alpha)$ is not one of the a_j, $j \in [\![1, m]\!]$.

Let $\beta \stackrel{\text{def}}{=} \alpha_m$. Note that $\alpha_0 = \alpha$, and that all the α_i have the same domain.

Let us show that for all i, the same actions as t can be performed in P_{α_i} with the same agents emitting Voted events, *i.e.* that

$$\forall i \in [\![0, k_1]\!].\ \exists t_i.\ t_i =_\tau t\ \wedge\ \neg\mathsf{blocking}(t_i, P_{\alpha_i}).$$

By contradiction, assume this property does not hold, and let i be the smallest index that falsifies it. Hence,

$$\forall t_i.\ t_i =_\tau t\ \Rightarrow\ \mathsf{blocking}(t_i, P_{\alpha_i}). \tag{1}$$

In addition, note that since the property is clearly satisfied at the 0th step, $i \geq 1$. Hence, it holds for $i-1$, *i.e.* there exists t_{i-1} such that $t_{i-1} =_\tau t$, and $\neg\mathsf{blocking}(t_{i-1}, P_{\alpha_{i-1}})$. Since $t = t'.\mathtt{out}(c_r, x)$, we also have $t_{i-1} = t'_{i-1}.\mathtt{out}(c_r, x)$ for some t'_{i-1} such that $t'_{i-1} =_\tau t'$.

Then, for all $t'_i =_\tau t'_{i-1}$ ($=_\tau t'$), by (1), $\mathsf{blocking}(t'_i.\mathtt{out}(c_r, x), P_{\alpha_i})$ holds.

The same sequence of actions t_{i-1} is blocking at step i and not at step $i-1$, which differs only by the vote of of a_i. This lets us construct an attack on privacy, which constitutes a contradiction. Indeed, by assumption 5, we may add a voter $b \notin \mathrm{dom}(\alpha)$, who votes for v'_i at step $i-1$, and for v_i at step i, and there exists tr such that

- there exists $tr' =_\tau tr$ and ψ such that $(t'_{i-1}.tr'.\mathtt{out}(c_r, x), \psi) \in \mathsf{trace}(P_{\alpha_{i-1} \cup \{b \mapsto v'_i\}})$.
- for all $t'_i =_\tau t'_{i-1}$ ($=_\tau t'$), we have shown that $\mathsf{blocking}(t'_i.\mathtt{out}(c_r, x), P_{\alpha_i})$, and thus for all $tr'' =_\tau tr$ and all ψ', $(t'_i.tr''.\mathtt{out}(c_r, x), \psi') \notin \mathsf{trace}(P_{\alpha_i \cup \{b \mapsto v_i\}})$.

Therefore, the processes $P_{\alpha_i \cup \{b \mapsto v_i\}}$ and $P_{\alpha_{i-1} \cup \{b \mapsto v'_i\}}$ are not trace equivalent. Since they only differ by the votes of a_i and b, who respectively vote for v_i, v'_i on the left and v'_i, v_i on the right, this breaks privacy, which contradicts the hypotheses.

Thus, for all i, there exists t_i such that $t_i =_\tau t$, of the form $t'_i.\mathtt{out}(c_r, x)$, such that $\neg\mathsf{blocking}(t_i, P_{\alpha_i})$. In other words, there exists ϕ_i such that $(t_i, \phi_i) \in \mathsf{trace}(P_{\alpha_i})$. By assumption 1, there exists V_i such that $\phi_i(x) \in R(\rho(V_i))$, *i.e.* $\mathsf{result}(t_i, \phi_i, V_i)$. Let $V' \stackrel{\text{def}}{=} V_m$. Note that $V_0 = V$.

For all $i \in [\![0, m-1]\!]$, α_i and α_{i+1} only differ by the vote of a_{i+1}, which is v'_{i+1} in α_{i+1} and v_{i+1} in α_i. Hence, by F, we have $\rho(V_i \uplus \{\!|v'_{i+1}|\!\}) = \rho(V_{i+1} \uplus \{\!|v_{i+1}|\!\})$.

That is to say, by assumption 3, that $\rho(V_i) * \rho(\{\!|v'_{i+1}|\!\}) = \rho(V_{i+1}) * \rho(\{\!|v_{i+1}|\!\})$.

Therefore, by rewriting these m equalities successively, we have

$$\rho(V_0) * \rho(\{\!|v'_1|\!\}) * \ldots * \rho(\{\!|v'_m|\!\}) = \rho(V_m) * \rho(\{\!|v_1|\!\}) * \ldots * \rho(\{\!|v_m|\!\}),$$

i.e., by assumption 3,

$$\rho(V) * \rho(\{\!|v'_i \mid i \in [\![1, m]\!]|\!\}) = \rho(\{\!|v_i \mid i \in [\![1, m]\!]|\!\}) * \rho(V').$$

By definition, this means

$$\rho(V) * \rho(V_{\text{change}}) = \rho(V_{\text{wanted}}) * \rho(V')$$

which concludes the proof. □

Lemma A.3 (*FF* implies *V* with a neutral vote). *Under assumptions 1, 2, 3, 4, 5, and assuming the existence of a neutral vote (assumption 6),*

$$FF \Rightarrow V$$

Proof. Under assumption 6, there exists a neutral vote v_{neutral}. Assume FF holds.

Let α be an affectation of votes, and let $(t'.\mathtt{out}(c_r, x), \phi) \in \mathsf{trace}(P_\alpha)$. Let $t \stackrel{\text{def}}{=} t'.\mathtt{out}(c_r, x)$. To prove individual verifiability, we need to show that the result in this trace contains at least the (intended) votes of the honest voters. That is to say, we must show that there exists V_c such that $\phi(x) \in R(\rho(\{\!|v \mid \exists a.\ \mathsf{Voted}(a, v) \in t|\!\} \uplus V_c))$.

By assumption 1, there exists V such that $\phi(x) \in R(\rho(V))$. We have $\mathsf{result}(t, \phi, V)$.

Let $V_{\text{wanted}} \stackrel{\text{def}}{=} \{\!|v \mid \exists a.\ \mathsf{Voted}(a, v) \in t|\!\}$ be the multiset of all intended honest votes in t. Let $k \stackrel{\text{def}}{=} |V_{\text{wanted}}|$, and $V_{\text{change}} \stackrel{\text{def}}{=} k \cdot v_{\text{neutral}}$.

By FF, there exists a multiset V_c such that $\rho(V) * \rho(V_{\text{change}}) = \rho(V_{\text{wanted}}) * \rho(V_c)$.

By assumption 3, $\rho(V_{\text{change}}) = \rho(\{\!|v_{\text{neutral}}|\!\})^k$. As, by assumption 6, $\rho(\{\!|v_{\text{neutral}}|\!\})$ is neutral for $*$, so is $\rho(V_{\text{change}})$.

Therefore, $\rho(V) = \rho(V_{\text{wanted}}) * \rho(V_c) = \rho(V_{\text{wanted}} \uplus V_c)$, which proves the claim. □

Lemma A.4 (*FF* implies *V* with a special vote). *Under assumptions 1, 2, 3, 4, 5, and assuming the existence of a vote that is counted separately (assumption 7),*

$$FF \Rightarrow V$$

Proof. Under assumption 7, there exists a special vote v_{special} which is counted separately in the result.

Assume FF holds.

Let α be an affectation of votes, and let $(t'.\mathtt{out}(c_r, x), \phi) \in \text{trace}(P_\alpha)$. Let $t \overset{\text{def}}{=} t'.\mathtt{out}(c_r, x)$. To prove individual verifiability, we need to show that the result in this trace contains at least the (intended) votes of the honest voters. That is to say, we must show that there exists V_c such that $\phi(x) \in R(\rho(\{\!|v \mid \exists a.\ \mathsf{Voted}(a, v) \in t|\!\} \uplus V_c))$.

By assumption 1, there exists V such that $\phi(x) \in R(\rho(V))$. We have $\mathsf{result}(t, \phi, V)$.

Let $V_{\text{wanted}} \overset{\text{def}}{=} \{\!|v \mid \exists a.\ \mathsf{Voted}(a, v) \in t \ \wedge \ v \neq v_{\text{special}}|\!\}$ be the multiset of all intended honest votes in t that are not v_{special}.

Let $k \overset{\text{def}}{=} |V_{\text{wanted}}|$, and $V_{\text{change}} \overset{\text{def}}{=} k \cdot v_{\text{special}}$.

By FF, there exists a multiset V_c such that $\rho(V) * \rho(V_{\text{change}}) = \rho(V_{\text{wanted}}) * \rho(V_c)$.

We may rewrite this equality as $\rho(V \uplus k \cdot v_{\text{special}}) = \rho(V_{\text{wanted}} \uplus V_c)$ by assumption 3.

Assumption 7 then lets us deduce that

$$(V \uplus k \cdot v_{\text{special}})(v_{\text{special}}) = (V_{\text{wanted}} \uplus V_c)(v_{\text{special}}).$$

Yet, $V_{\text{wanted}}(v_{\text{special}}) = 0$ by definition. Therefore, $V_c(v_{\text{special}}) \geq k$, and we may write $V_c = V_c' \uplus k \cdot v_{\text{special}}$ for some V_c'. We then have

$$\rho(V \uplus k \cdot v_{\text{special}}) = \rho(V_{\text{wanted}} \uplus V_c' \uplus k \cdot v_{\text{special}}),$$

which implies by assumption 7 that

$$\rho(V) = \rho(V_{\text{wanted}} \uplus V_c').$$

Since V_{wanted} contains all the intended honest votes different from v_{special}, it remains to be proved that V_c' contains sufficiently many instances of v_{special}.

Let $k' \overset{\text{def}}{=} |\{\!|\mathsf{Voted}(a, v_{\text{special}}) \in t \ \mid \ a \in \mathcal{A}|\!\}|$ the number of intended votes for v_{special} in t; and $V'_{\text{wanted}} \overset{\text{def}}{=} k' \cdot v_{\text{special}}$.

Let $v \in \mathcal{V}$ such that $v \neq v_{\text{special}}$. Let $V'_{\text{change}} \overset{\text{def}}{=} k' \cdot v$.

By FF, there exists V_c'' such that

$$\rho(V) * \rho(V'_{\text{change}}) = \rho(V'_{\text{wanted}}) * \rho(V_c''),$$

i.e., by assumption 3,

$$\rho(V \uplus k' \cdot v) = \rho(k' \cdot v_{\text{special}} \uplus V_c'').$$

By assumption 7, we then have

$$(V \uplus k' \cdot v)(v_{\text{special}}) = (k' \cdot v_{\text{special}} \uplus V_c'')(v_{\text{special}}).$$

As before, since $v \neq v_{\text{special}}$, this means that $V(v_{\text{special}}) \geq k'$.

We already know that $\rho(V) = \rho(V_{\text{wanted}} \uplus V_c')$. By applying assumption 7 again, $(V_{\text{wanted}} \uplus V_c')(v_{\text{special}}) = V(v_{\text{special}}) \geq k'$. Since $v_{\text{special}} \notin V_{\text{wanted}}$, $(V_c')(v_{\text{special}}) \geq k'$, *i.e.* $V_c' = V_c''' \uplus k' \cdot v_{\text{special}}$ for some V_c'''.

Therefore we have

$$\rho(V) = \rho(V_{\text{wanted}} \uplus k' \cdot v_{\text{special}} \uplus V_c''') = \rho(\{\!|v \ \mid \ \exists a.\ \mathsf{Voted}(a, v) \in t|\!\} \uplus V_c'''),$$

which concludes the proof. □

The next theorem corresponds to Theorem 3.11.

Theorem A.5 (Privacy implies individual verifiability when there is a neutral or a special vote). *Under assumptions 1, 2, 3, 4, 5, and assuming the existence of either a neutral vote or a special vote counted separately (assumptions 6 or 7),*

$$P \Rightarrow V$$

Proof. This follows directly from Lemmas A.1, A.2, A.3, and A.4. □

Appendix B COMPUTATIONAL PROOF

B.1 Assumptions summary

We first recall some of the hypotheses used in the proofs, that were presented in Sections 2, 4.1 and 4.3. Some of these assumptions differ depending on whether the scheme is *id*-based or *cred*-based, or apply only to one of these two classes of schemes. In such cases, the differences will be clearly stated.

(1) The voting scheme has the *piecewise tally* property. In the case of *id*-based schemes, the assumption is that for all boards BB_1, BB_2, if sk is the election key and U is a list of registered users and credentials, and if

$$\begin{aligned}&\mathsf{ValidTally}(\mathsf{BB}_1 \uplus \mathsf{BB}_2, \mathsf{sk}, \mathsf{U}) \wedge \\ &\quad \forall b \in \mathsf{BB}_1.\ \forall b' \in \mathsf{BB}_2.\ \mathsf{open}_{id}(b) \neq \mathsf{open}_{id}(b')\end{aligned}$$

then the tally can be computed separately:

$$Tally(\mathsf{BB}_1 \uplus \mathsf{BB}_2, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}_1, \mathsf{sk}, \mathsf{U}) * \mathsf{Tally}(\mathsf{BB}_2, \mathsf{sk}, \mathsf{U}).$$

In the case of credential-based schemes, the assumption is that for all boards BB_1, BB_2, if sk is the election key and U is a list of registered users and credentials, and if

$$\forall b \in \mathsf{BB}_1.\ \forall b' \in \mathsf{BB}_2.\ \mathsf{open}_{cred}(b, \mathsf{sk}, \mathsf{U}) \neq \mathsf{open}_{cred}(b', \mathsf{sk}, \mathsf{U})$$

then the tally can be computed separately:

$$Tally(\mathsf{BB}_1 \uplus \mathsf{BB}_2, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}_1, \mathsf{sk}, \mathsf{U}) * \mathsf{Tally}(\mathsf{BB}_2, \mathsf{sk}, \mathsf{U}).$$

(2) In the case of *id*-based schemes only, the tally only counts ballots cast with registered *id*s, *i.e.* $\forall \mathsf{BB}, \mathsf{sk}, \mathsf{U}.\ \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}', \mathsf{sk}, \mathsf{U})$ where $\mathsf{BB}' = [b \in \mathsf{BB} \mid (\mathsf{open}_{id}(b), *) \in \mathsf{U}]$.
(3) Registering more voters does not change the tally. In the case of *id*-based schemes, the assumption is that for all board BB, election key sk and list of voters U, if U, U′ have no *id* in common and $\forall b \in \mathsf{BB}.\ (\mathsf{open}_{id}(b), *) \notin \mathsf{U}'$, then $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U} \cap \mathsf{U}')$. In the case of credential-based schemes, the assumption is that if U, U′ share no credentials and $\forall b \in \mathsf{BB}.\ (*, \mathsf{open}_{cred}(b, \mathsf{sk}, \mathsf{U} \cup \mathsf{U}')) \notin \mathsf{U}'$, then $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U} \cup \mathsf{U}')$.
(4) The voting scheme is *correct,i.e.* for all distinct identities $\mathsf{U} = id_1, \ldots, id_n$, and credentials $cred_1, \ldots, cred_n$, for all votes $v_1, \ldots, v_n$, for all election keys (pk, sk), if $\mathsf{BB} = [\mathsf{Vote}(id_i, cred_i, \mathsf{pk}, v_i) | i \in [\![1, n]\!]]$, then

$$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \rho(v_1, \ldots, v_n).$$

(5) There exists a neutral vote $v_{\mathsf{neutral}} \in \mathcal{V}$, such that $\rho(\{v_{\mathsf{neutral}}\})$ is neutral for $*$.
(6) Given a multiset of valid votes V and a result r, it is possible to efficiently decide whether r can be decomposed into $\rho(V) * \rho(V')$ for some multiset V' of valid votes.
That is to say there exists a PPTM D such that

$$\forall r, V. \quad \mathsf{D}(r, V) = 1 \iff \exists V'.\ r = \rho(V) * \rho(V').$$

In the proof that privacy implies individual verifiability against a dishonest board provided the voters are careful, we also use the following two hypotheses:

(7) If a voter's intended ballot is indeed the one which will be selected from the board by the revote policy, then this voter must be satisfied with the board. Formally, for all registered voter *id* with credential *cred*, for all ballot b, voter knowledge L, and board BB,
   - For *id*-based schemes: if the revote policy is to count the last (resp. first) ballot cast for each *id*, the assumption is that if the last (resp. first) element of L is $(b, *)$, and the last (resp. first) ballot $b' \in \mathsf{BB}$ such that $\mathsf{open}_{id}(b') = id$ is b, then $\mathsf{VerifVoter}(id, cred, \mathsf{L}, \mathsf{BB})$ holds.
   - For credential-based schemes: if the revote policy is to count the last (resp. first) ballot cast for each credential, the assumption is that if the last (resp. first) element of L is $(b, *)$, and the last (resp. first) ballot $b' \in \mathsf{BB}$ such that $\mathsf{open}_{cred}(b', \mathsf{sk}, \mathsf{U}) = cred$ is b, then $\mathsf{VerifVoter}(id, cred, \mathsf{L}, \mathsf{BB})$ holds.

(8) If a voter *id* is satisfied with a board BB, then *id* remains satisfied with any board obtained from BB by adding new ballots that do not interfere with *id*'s given the revote policy.
Formally, for all board BB, election key pk, registered voter *id* with credential *cred* and knowledge L, for all BB′,
   - For *id*-based schemes: the assumption is that if $\forall b \in \mathsf{BB}'.\ \mathsf{open}_{id}(b) \neq id$ then

$$\mathsf{VerifVoter}(id, cred, \mathsf{L}, \mathsf{BB}) \iff \mathsf{VerifVoter}(id, cred, \mathsf{L}, \mathsf{BB} \uplus \mathsf{BB}').$$

   - For credential-based schemes: the assumption is that if $\forall b \in \mathsf{BB}'.\ \mathsf{open}_{cred}(b, \mathsf{sk}, \mathsf{U}) \neq cred$ then

$$\mathsf{VerifVoter}(id, cred, \mathsf{L}, \mathsf{BB}) \iff \mathsf{VerifVoter}(id, cred, \mathsf{L}, \mathsf{BB} \uplus \mathsf{BB}').$$

We will also assume, depending on whether the voting scheme is *id* or credential based, that no polynomial adversary wins $\mathsf{Exp}^{\mathsf{ValidTally}}$ with non-negligible probability, *i.e.*

$$\forall \mathcal{A}.\ \mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{\mathcal{A}}(\lambda) = 1\right] \text{ is negligible,}$$

or that no polynomial adversary wins $\mathsf{Exp}^{\mathsf{NM}}$ with non-negligible probability,*i.e.*

$$\forall \mathcal{A}.\ \mathrm{P}\left[\mathsf{Exp}^{\mathsf{NM}}_{\mathcal{A}}(\lambda) = 1\right] \text{ is negligible,}$$

where $\mathsf{Exp}^{\mathsf{ValidTally}}$ is defined on Figure 6 and $\mathsf{Exp}^{\mathsf{NM}}$ is defined on Figure 7.

$\underline{O^{v,\mathsf{f}}_{\mathsf{vote}}(id, v)}$

if $(id, *) \in \mathsf{U} \backslash \mathsf{CU} \ \wedge\ (id, *) \notin \mathsf{Voted}$ **then**

$\quad b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v)$

$\quad \mathsf{BB} \leftarrow \mathsf{BB} \| b$

$\quad \mathsf{Voted} \leftarrow \mathsf{Voted} \| (id, v)$

$\quad$ **return** b

where $(id, cred_{id}) \in \mathsf{U}$

$\underline{O^{p,\mathsf{f}}_{\mathsf{vote}}(id, v_0, v_1)}$

if $(id, *) \in \mathsf{U} \backslash \mathsf{CU} \ \wedge\ (id, *) \notin \mathsf{V}_0 \cup \mathsf{V}_1$ **then**

$\quad b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v_\beta)$

$\quad \mathsf{BB} \leftarrow \mathsf{BB} \| b$

$\quad \mathsf{V}_0 \leftarrow \mathsf{V}_0 \| (id, v_0)$

$\quad \mathsf{V}_1 \leftarrow \mathsf{V}_1 \| (id, v_1)$

$\quad$ **return** b

where $(id, cred_{id}) \in \mathsf{U}$

Figure 12: Oracles for the individual verifiability and privacy games (revote policy = first vote)

B.2 Privacy implies individual verifiability with a honest board (proof of Theorem 4.4)

We consider the case of protocols where the revote policy is to count only the last ballot (for each *id* or credential) or the first ballot. In the case of the last ballot, the definitions of the individual verifiability and privacy games can be found on Figures 4 and 5. In the case of the first ballot, we adapt these definitions by replacing the oracles O^{v}_{vote} and O^{p}_{vote} with $O^{v,\mathsf{f}}_{\mathsf{vote}}$ and $O^{p,\mathsf{f}}_{\mathsf{vote}}$, described on Figure 12. These two oracles are analogous to O^{v}_{vote} and O^{p}_{vote}, but keep only the first votes from each voter in the lists V_0, V_1, Voted, instead of the last.

The following theorem corresponds to Theorem 4.4.

Theorem B.1. *Under assumptions 1, 2, 3, 4, 5, 6:*

- *for id-based schemes, assuming that no polynomial adversary wins* $\mathsf{Exp}^{\mathsf{ValidTally}}$ *with non-negligible probability,*
- *and for credential-based schemes, assuming that no polynomial adversary wins* $\mathsf{Exp}^{\mathsf{NM}}$ *with non-negligible probability,*

if

$$\forall \mathcal{A}.\ \left| \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{A}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{A}}(\lambda) = 1\right] \right| \text{ is negligible,}$$

then

$$\forall \mathcal{A}.\ \mathrm{P}\left[\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda) = 1\right] \text{ is negligible.}$$

Proof. We first consider the case of *id*-based protocols.

Let $\mathcal{A} = \mathcal{A}_1, \mathcal{A}_2$ be an adversary that breaks individual verifiability, *i.e.* wins $\mathsf{Exp}^{\mathsf{verif}}$. We consider an adversary $\mathcal{B} = \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ that attacks privacy, *i.e.* plays $\mathsf{Exp}^{\mathsf{priv},\beta}$:

- $\mathcal{B}_1^{O_{\mathsf{reg}}, O_{\mathsf{corr}}}(\mathsf{pk})$ first simulates $\mathcal{A}_1^{O_{\mathsf{reg}}, O_{\mathsf{corr}}}(\mathsf{pk})$, *i.e.* $\mathcal{B}$ registers and corrupts the same identities as $\mathcal{A}$, while keeping a list U_1 of the identities it registers by calling O_{reg}. $\mathcal{A}$ returns some *state*. $\mathcal{B}_1$ then calls O_{reg} another $|\mathsf{U}_1|$ times, on fresh identities that do not appear in U_1. It keeps a list U_2 of these fresh identities.

- $\mathcal{B}_2^{O^{p}_{\mathsf{vote}}, O_{\mathsf{cast}}}(state_1, \mathsf{pk})$ maintains a list L, initially empty, which will be used to record the calls to O^{p}_{vote}, and a representation of the current board BB'. $\mathcal{B}_2$ first simulates $\mathcal{A}_2^{O^{v}_{\mathsf{vote}}, O_{\mathsf{cast}}}(state, \mathsf{pk})$:
  - for each call to $O^{v}_{\mathsf{vote}}(id, v)$, provided $id \in \mathsf{U}_1$, $\mathcal{B}$ calls $O^{p}_{\mathsf{vote}}(id, v, v_{\mathsf{neutral}})$, and, depending on the revote policy
    - either checks if *id* is already present in L, to add (id, v) to L only if it is not (note that, in case it is, $O^{p}_{\mathsf{vote}}(id, v, v_{\mathsf{neutral}})$ returns nothing);
    - or adds (id, v) to L and removes any previous couple (id, v') (for any v') from L.

    If $O^{p}_{\mathsf{vote}}(id, v, v_{\mathsf{neutral}})$ returns a ballot b, $\mathcal{B}$ then adds it to BB', and passes it to $\mathcal{A}_2$.
  - for each call to $O_{\mathsf{cast}}(id, b)$, provided $id \in \mathsf{U}_1$, $\mathcal{B}$ calls $O_{\mathsf{cast}}(id, b)$, and, if $\mathsf{Valid}(id, b, \mathsf{BB}', \mathsf{pk})$, adds it to BB'.

  Let then Voted be the list of the votes in L: Voted = $[v \text{ for } (id, v) \in L]$. Let also BB_a be the value of the board BB' at that point. Note that BB_a only contains identities from U_1.

  If at any point during the simulation $\mathcal{A}_2$ blocks or fails, $\mathcal{B}$ stops the simulation. In any case, $\mathcal{B}_2$ calls $O^{p}_{\mathsf{vote}}(id_1, v_{\mathsf{neutral}}, v_1)$, …, $O^{p}_{\mathsf{vote}}(id_l, v_{\mathsf{neutral}}, v_l)$, where $v_1, \ldots, v_l$ are the elements of Voted, and $id_1, \ldots, id_l$ are pairwise distinct identities from U_2. Note that since all identities in L are distinct and in U_1, l is indeed smaller than $|\mathsf{U}_1| = |\mathsf{U}_2|$. Let $\mathsf{BB}_b = [b_1, \ldots, b_l]$ be the set of new ballots added to the board by these l calls, *i.e.* ballots for v_{neutral} if $\beta = 0$ and $v_1, \ldots, v_l$ if $\beta = 1$.
  At this point, we have $\mathsf{BB} = \mathsf{BB}_a \uplus \mathsf{BB}_b$.

$\mathcal{B}_2$ then returns a $state_2$ indicating whether $\mathcal{A}_2$ has failed. Intuitively, $\mathcal{A}$ is accurately simulated only if $\beta = 0$, *i.e.* is shown a board where the votes it wanted to cast have indeed been cast. Hence whenever $\mathcal{A}$ wins $\mathsf{Exp}^{\mathsf{verif}}$, the simulated $\mathcal{A}_2$ failing can only mean that $\beta = 1$.

- $\mathsf{Exp}^{\mathsf{priv}}$ will then check that $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$, where V_0 and V_1 are the lists it keeps, which contain the first (or last, depending on the revote policy) votes $\mathcal{O}^{p}_{\mathsf{vote}}$ has been called on for each id. Considering the definition of $\mathcal{B}_2$, at this point we always have
$$\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1) = \rho(v_1, \dots, v_l, \underbrace{v_{\mathsf{neutral}}, \dots, v_{\mathsf{neutral}}}_{l \text{ times}})$$
Hence this check necessarily succeeds, and $\mathsf{Exp}^{\mathsf{priv}}$ computes $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U})$.
- $\mathcal{B}_3$ obtains a result r. If $r = \bot$, $\mathcal{B}_3$ returns 1. If $\mathcal{A}_2$ blocked previously, $\mathcal{B}$ returns $\beta' = 1$. Otherwise, $\mathcal{B}$ computes $\mathsf{D}(r, \mathsf{Voted})$ and:
  - if $\exists V_c.\ r = \rho(\mathsf{Voted}) * \rho(V_c)$ then $\mathcal{B}$ returns $\beta' = 1$
  - otherwise, $\mathcal{B}$ returns $\beta' = 0$.

We also construct an adversary C, who plays the game $\mathsf{Exp}^{\mathsf{ValidTally}}$.

- C_1 is identical to $\mathcal{B}_1$.
- C_2 first draws at random a bit β'', and then simulates $\mathcal{B}_2$ up to the point where $\mathcal{B}_2$ has finished simulating $\mathcal{A}_2$. C_2 keeps a list BB, initially empty. It simulates each call to $\mathcal{O}^{p}_{\mathsf{vote}}(id, v_0, v_1)$ $\mathcal{B}$ does by calling $\mathcal{O}^{vt}_{\mathsf{vote}}(id, v_{\beta''})$, and appending the obtained ballot to BB. It simulates each call to $\mathcal{O}_{\mathsf{cast}}(id, b)$ by appending b to BB, provided id is dishonest and $\mathsf{Valid}(id, b, \mathsf{BB}, \mathsf{pk})$.
- Once C_2 has finished simulating $\mathcal{B}_2$, it draws at random a number $k \in [\![1, l]\!]$ Recall that l is the number of different ids $\mathcal{O}^{p}_{\mathsf{vote}}$ (and thus $\mathcal{O}^{vt}_{\mathsf{vote}}$) has been called on, and is also the number of additional calls to $\mathcal{O}^{p}_{\mathsf{vote}}$ $\mathcal{B}$ will make. Note that, at this point, BB in $\mathsf{Exp}^{\mathsf{ValidTally}}_{C}$ is the same as BB_a in $\mathsf{Exp}^{\mathsf{priv},\beta''}_{\mathcal{B}}$. C then simulates the first $k-1$ calls to $\mathcal{O}^{p}_{\mathsf{vote}}(id, v_0, v_1)$, again by calling $\mathcal{O}^{vt}_{\mathsf{vote}}(id, v_{\beta''})$. If the kth call is $\mathcal{O}^{p}_{\mathsf{vote}}(id, v_0, v_1)$, C returns $(\mathsf{BB}, id, v_{\beta''})$.

We will now prove that if $\mathcal{A}$ breaks individual verifiability, then $\mathcal{B}$ breaks privacy provided C does not break $\mathsf{Exp}^{\mathsf{ValidTally}}$.

The adversary C is polynomial, *i.e.* there exists a polynomial $q(\lambda)$ bounding its number of operations.

For any β, assume $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1)$ holds and $\mathsf{ValidTally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}_1 \cup \mathsf{U}_2)$ does not. Thus, by assumption 3, $\mathsf{ValidTally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}_1 \cup [(id_1, cred_{id_1}), \dots, (id_l, cred_{id_l})])$ does not hold either. Hence, there exists a smallest $k \in [\![1, l]\!]$ such that $\mathsf{ValidTally}(\mathsf{BB}_a \uplus [b_1, \dots, b_{k-1}], \mathsf{sk}, \mathsf{U}_1 \cup \mathsf{U}'_2)$ holds and $\mathsf{ValidTally}(\mathsf{BB}_a \uplus [b_1, \dots, b_k], \mathsf{sk}, \mathsf{U}_1 \cup \mathsf{U}'_2 \cup [(id_k, cred_{id_k})])$ does not, where $\mathsf{U}'_2 = [(id_1, cred_{id_1}), \dots, (id_{k-1}, cred_{id_{k-1}})]$. b_k has been added to BB_b by the kth call to $\mathcal{O}^{p}_{\mathsf{vote}}$ by $\mathcal{B}$: $b_k = \mathsf{Vote}(id_k, cred_{id_k}, \mathsf{pk}, v_\beta)$ for some v_β. Thus, provided C correctly guesses $\beta'' = \beta$ and k, C returns $(\mathsf{BB}_a \uplus [b_1, \dots, b_{k-1}], id_k, v_\beta)$ the conditions on BB in $\mathsf{Exp}^{\mathsf{ValidTally}}_{C}$ holds, and thus $\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1$. Therefore, $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1)$ holds and $\mathsf{ValidTally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}_1 \cup \mathsf{U}_2)$ does not with probability at most $2l\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]$, which is smaller than $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]$ since $l \le q(\lambda)$.

Since BB_a only contains ballots cast for identities in U_1 and BB_b for identities in U_2, and $\mathsf{U}_1 \cap \mathsf{U}_2 = \emptyset$, if $\mathsf{ValidTally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}_1 \cup \mathsf{U}_2)$, by assumption 1 we have (regardless of β)
$$r = \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$$
In addition, by assumption 3, $\mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1)$. Hence
$$r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U}).$$

- If $\beta = 0$: then $\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{B}}(\lambda) = 1$ if and only if $\mathcal{B}_3$ returns 1 in this game, which happens either when $\mathcal{A}$ (simulated by $\mathcal{B}$) blocks, or when it does not and $\exists V_c.\ r = \rho(\mathsf{Voted}) * \rho(V_c)$ or $r = \bot$.
  Assume $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) \Rightarrow \mathsf{ValidTally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$, which, as we have established, holds except with probability at most $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]$.
  Let us first examine the case where $\mathcal{A}$ does not block and $r \neq \bot$. Since $r \neq \bot$, $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1)$ holds. Hence, $\mathsf{ValidTally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$ also holds, and as explained previously we thus know that $r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$. Since $\beta = 0$, BB_b only contains ballots for v_{neutral}. Hence, by assumption 4, $\mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U}) = \rho(v_{\mathsf{neutral}})^l = \rho(v_{\mathsf{neutral}})$. Thus $r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1)$.
  The condition $\exists V_c.\ r = \rho(\mathsf{Voted}) * \rho(V_c)$ is therefore equivalent to $\exists V_c.\ \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) = \rho(\mathsf{Voted}) * \rho(V_c)$. Since, in this case, $\mathcal{A}$ has been accurately simulated without blocking, does not return $\bot$, and BB_a is the board after its execution, this is exactly the condition under which $\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda)$ does not return 1.

Hence $\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{B}}(\lambda) = 1$ if and only if either $\mathcal{A}$ (simulated by $\mathcal{B}$) blocks, or constructs a board whose tally is $\perp$, or it does not and $\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda) \neq 1$.
Since $\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda)$ also does not return 1 when $\mathcal{A}$ blocks or when the tally is $\perp$, this implies that, unless the implication $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) \Rightarrow \mathsf{ValidTally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$ is false, $\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{B}}(\lambda) = 1$ if and only if $\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda) \neq 1$. Thus

$$\left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{B}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda) \neq 1\right]\right| \leq 2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]. \tag{2}$$

- If $\beta = 1$: then $\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{B}}(\lambda) = 1$ if and only if $\mathcal{B}_3$ returns 1 in this game, which happens either when $\mathcal{A}$ (simulated by $\mathcal{B}$) blocks, or when it does not and $\exists V_c.\ r = \rho(\mathsf{Voted}) * \rho(V_c)$ or $r = \perp$.
Assume $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) \Rightarrow \mathsf{ValidTally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$, which, as we have established, holds except with probability at most $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]$.
Let us first examine the case where $\mathcal{A}$ does not block and $r \neq \perp$. As in the $\beta = 0$ case, we thus have $r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$. Since $\beta = 1$, BB_b contains ballots for $v_1, \ldots, v_l$, *i.e.* for Voted. Hence, by assumption 4, $\mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U}) = \rho(\mathsf{Voted})$, and therefore $r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) * \rho(\mathsf{Voted})$. By definition of Tally, there exists V such that $\mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) = \rho(V)$. Hence the condition $\exists V_c.\ r = \rho(\mathsf{Voted}) * \rho(V_c)$ necessarily holds.

  Therefore, unless the implication $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) \Rightarrow \mathsf{ValidTally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$ is false, $\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{B}}(\lambda) = 1$ if and only if either $\mathcal{A}$ (simulated by $\mathcal{B}$) blocks, or it does not and returns a board whose tally is $\perp$, or it does not and returns a board whose tally is not $\perp$.
Hence

$$1 - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{B}}(\lambda) = 1\right] \leq 2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]. \tag{3}$$

We thus have, using 4 and 5:

$$\begin{array}{rcl}
\mathrm{P}\left[\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda) = 1\right] & = & \left(1 - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda) \neq 1\right]\right) + \left(\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{B}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{B}}(\lambda) = 1\right]\right) \\
& & + \left(\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{B}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{B}}(\lambda) = 1\right]\right) \\
& = & \left(\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{B}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda) \neq 1\right]\right) + \left(\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{B}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{B}}(\lambda) = 1\right]\right) \\
& & + \left(1 - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{B}}(\lambda) = 1\right]\right) \\
& \leq & \left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{B}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{B}}(\lambda) = 1\right]\right| + 4q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]
\end{array}$$

Therefore, if $\mathcal{A}$ breaks individual verifiability, *i.e.* if $\mathrm{P}\left[\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda) = 1\right]$ is not negligible, then $\mathcal{B}$ breaks privacy, or C breaks $\mathsf{Exp}^{\mathsf{ValidTally}}$.

The proof for the case of credential-based protocols is very similar. In that case, instead of the $\mathsf{Exp}^{\mathsf{ValidTally}}$ assumption, we assume that

$$\forall \mathcal{A}.\ \mathrm{P}\left[\mathsf{Exp}^{\mathsf{NM}}_{\mathcal{A}}(\lambda) = 1\right] \text{ is negligible}$$

where $\mathsf{Exp}^{\mathsf{NM}}$ is defined on Figure 7.

Let $\mathcal{A} = \mathcal{A}_1, \mathcal{A}_2$ be an adversary that breaks individual verifiability, *i.e.* wins $\mathsf{Exp}^{\mathsf{verif}}$. We construct the adversary $\mathcal{B}$, that plays $\mathsf{Exp}^{\mathsf{priv}}$, as in the *id*-based case. We also construct $\mathcal{D}$, that plays $\mathsf{Exp}^{\mathsf{NM}}$, and is similar to C in the *id*-based case, except that $\mathcal{D}$ simulates calls to O^{p}_{vote} by calling O_{c} instead of O^{vt}_{vote}:

- $\mathcal{D}_1$ is identical to $\mathcal{B}_1$.
- $\mathcal{D}_2$ first draws at random a bit β'', and then simulates $\mathcal{B}_2$ up to the point where $\mathcal{B}_2$ has finished simulating $\mathcal{A}_2$. It simulates each call to $O^{p}_{\mathsf{vote}}(id, v_0, v_1)$ $\mathcal{B}$ makes by calling $O_{\mathsf{c}}(id, v_{\beta''})$.
- Once $\mathcal{D}_2$ has finished simulating $\mathcal{B}_2$ up to the point $\mathcal{B}_2$ has simulated $\mathcal{A}_2$, $\mathcal{D}$ obtains a board BB_a (which is the same as BB_a in $\mathsf{Exp}^{\mathsf{priv},\beta''}_{\mathcal{B}}$). Since BB_a is obtained by simulating $\mathcal{A}$, which is polynomial, in all executions of $\mathcal{D}$ the length of BB_a is bounded by some polynomial $p(\lambda)$. $\mathcal{D}$ then draws at random a ballot in BB_a, and returns it.

Similarly to the proof for the *id*-based case, and keeping the same notations, it then follows that BB_a contains a ballot with a credential in U_2 the same credential with probability at most $2p(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{NM}}_{\mathcal{D}} = 1\right]$. Indeed, if such a ballot exists, it cannot have been produced by a call to O_{c}. Otherwise, it would necessarily have been produced by $O_{\mathsf{c}}(id, v)$ for some id, v such that $(id, cred) \in \mathsf{U}_2$, and the only calls to this oracle simulate calls made by $\mathcal{B}$ to O^{p}_{vote} when simulating $\mathcal{A}$. Since $\mathcal{B}$ only calls O^{p}_{vote} on *ids* in U_1 when simulating $\mathcal{A}$, this is contradictory. Thus, a ballot in BB_a with a credential in U_2 cannot have been produced by O_{c}. Hence, provided $\mathcal{D}$ picked $\beta'' = \beta$, and picks the right ballot in BB_a, which happens with probability at least $\frac{1}{p(\lambda)}$, $\mathcal{D}$ wins $\mathsf{Exp}^{\mathsf{NM}}$.

Therefore, BB_a contains no ballot with a credential in U_2, except with probability at most $2p(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{NM}}_{\mathcal{D}}(\lambda) = 1\right]$.

$\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}(\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{U}, \mathsf{CU} \leftarrow []$
$state \leftarrow \mathcal{A}_1^{O_{\mathsf{reg}}, O_{\mathsf{corr}}}(\mathsf{pk})$
$\mathsf{Voted}, \mathsf{L}_{id}$ (for all id in U) $\leftarrow []$
$\mathsf{BB}, state_2 \leftarrow \mathcal{A}_2^{O^{v,c}_{\mathsf{vote}}}(state, \mathsf{pk})$
$state_3 \leftarrow \mathcal{A}_3^{O_{\mathsf{happy\,BB}}}(state_2)$
if $\forall id.\ (id, *) \in \mathsf{Voted} \Rightarrow id \in \mathsf{H}$ **then**
  $r \leftarrow \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U})$
  if $r \neq \bot \wedge \forall V_c$ (finite) . $r \neq \rho(\{v_i\}_{1 \leq i \leq k} \uplus V_c)$ **then**
  return 1

where $\mathsf{Voted} = \{(id_1, v_1), \ldots, (id_k, v_k)\}$

$O^{v,c}_{\mathsf{vote}}(id, v)$

---

if $(id, *) \in \mathsf{U} \backslash \mathsf{CU}$ **then**
  $b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v)$
  $\mathsf{Voted} \leftarrow \mathsf{Voted}' \| (id, v)$
  $\mathsf{L}_{id} \leftarrow \mathsf{L}_{id} \| (b, v)$
  return b
    where $(id, cred_{id}) \in \mathsf{U}$
    and Voted' is obtained from
    Voted by removing all instances of $(id, *)$

Figure 13: Individual verifiability against a dishonest board with careful voters.

Note that by construction, all ballots in BB_b have credentials in U_2. Consequently, unless BB_a contains a ballot with a credential in U_2, by assumption 1, we have (regardless of β)

$$r = \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$$

which means, by assumption 3, that

$$r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U}).$$

The remainder of the proof is the same as before, and establishes that

$$\mathrm{P}\left[\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda) = 1\right] \leq \left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{B}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{B}}(\lambda) = 1\right]\right| + 4p(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{NM}}_{\mathcal{D}} = 1\right].$$

Thus if $\mathcal{A}$ breaks verifiability, *i.e.* if $\mathrm{P}\left[\mathsf{Exp}^{\mathsf{verif}}_{\mathcal{A}}(\lambda) = 1\right]$ is not negligible, then $\mathcal{B}$ breaks privacy, or $\mathcal{D}$ breaks $\mathsf{Exp}^{\mathsf{NM}}$, which proves the claim.

Note that, in these reductions, for each id in U_1, $\mathcal{B}$ makes at most as many calls to O^p_{vote} as $\mathcal{A}$ makes to O^v_{vote}, and at most as many calls to O_{cast} as $\mathcal{A}$ makes to O_{cast}. In addition, for each id in U_2, $\mathcal{B}$ makes at most one call to O^p_{vote}, and no call to O_{cast}. Thus, the exact same proof proves that the result also holds if both the games $\mathsf{Exp}^{\mathsf{priv}}$ and $\mathsf{Exp}^{\mathsf{verif}}$ are modified to prevent revote, by allowing only one call to $O^p_{\mathsf{vote}}/O^v_{\mathsf{vote}}$ and/or to $O_{\mathsf{cast}}/O_{\mathsf{cast}}$ for each id. □

B.3 Privacy implies individual verifiability with a dishonest board and careful voters (proof of Theorem 5.2)

We consider the case of protocols where the revote policy is to count only the last ballot (for each id or credential) or the first ballot. In the case of the last ballot, the definitions of the privacy game with a dishonest board and careful voters can be found on Figure 9. We adapt the verifiability game to the case of a dishonest board and careful voters as follows.

Definition B.2 (Individual verifiability against a dishonest board with careful voters). For an adversary $\mathcal{A} = \mathcal{A}_1, \mathcal{A}_2$ and a parameter λ, we consider the game $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}(\lambda)$ defined on Figure 13. The voting system is verifiable against a dishonest board with careful voters if

$$\forall \mathcal{A}.\ \mathrm{P}\left[\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}(\lambda) = 1\right] \text{ is negligible.}$$

In the case of the first ballot, we adapt these definitions by replacing the oracles $O^{v,c}_{\mathsf{vote}}$ and $O^{p,c}_{\mathsf{vote}}$ with $O^{v,c,\mathsf{f}}_{\mathsf{vote}}$ and $O^{p,c,\mathsf{f}}_{\mathsf{vote}}$, described on Figure 14. These two oracles are analogous to $O^{v,c}_{\mathsf{vote}}$ and $O^{p,c}_{\mathsf{vote}}$, but keep only the first votes from each voter in the lists V_0, V_1, Voted, instead of the last.

The following theorem corresponds to Theorem 5.2.

Theorem B.3. *Under assumptions 1, 2, 3, 4, 5, 6, 7, 8:*

- *for id-based schemes, assuming that no polynomial adversary wins* $\mathsf{Exp}^{\mathsf{ValidTally}}$ *with non-negligible probability,*
- *and for credential-based schemes, assuming that no polynomial adversary wins* $\mathsf{Exp}^{\mathsf{NM}}$ *with non-negligible probability,*

$$\begin{array}{l}
\underline{O_{\mathsf{vote}}^{v,c,\mathsf{f}}(id, v)} \\
\textbf{if } (id, *) \in \mathsf{U} \backslash \mathsf{CU} \ \wedge \ (id, *) \notin \mathsf{Voted} \textbf{ then} \\
\quad b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v) \\
\quad \mathsf{Voted} \leftarrow \mathsf{Voted} \| (id, v) \\
\quad \mathsf{L}_{id} \leftarrow \mathsf{L}_{id} \| (b, v) \\
\quad \textbf{return } b \\
\qquad \text{where } (id, cred_{id}) \in \mathsf{U}
\end{array}
\qquad
\begin{array}{l}
\underline{O_{\mathsf{vote}}^{p,c,\mathsf{f}}(id, v_0, v_1)} \\
\textbf{if } (id, *) \in \mathsf{U} \backslash \mathsf{CU} \ \wedge \ (id, *) \notin \mathsf{V}_0, \mathsf{V}_1 \textbf{ then} \\
\quad b \leftarrow \mathsf{Vote}(id, cred_{id}, \mathsf{pk}, v_\beta) \\
\quad \mathsf{V}_0 \leftarrow \mathsf{V}_0 \| (id, v_0) \\
\quad \mathsf{V}_1 \leftarrow \mathsf{V}_1 \| (id, v_1) \\
\quad \mathsf{L}_{id} \leftarrow \mathsf{L}_{id} \| (b, v_\beta) \\
\quad \textbf{return } b \\
\qquad \text{where } (id, cred_{id}) \in \mathsf{U}
\end{array}$$

Figure 14: Oracles for the verifiability and privacy games with dishonest boards and careful voters (revote policy = first vote)

if

$$\forall \mathcal{A}. \ \left| \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},1}(\lambda) = 1\right]\right| \text{ is negligible,}$$

then

$$\forall \mathcal{A}. \ \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{verif-careful}}(\lambda) = 1\right] \text{ is negligible.}$$

Proof. We first consider the case of *id*-based protocols.

Let $\mathcal{A} = \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ be an adversary that breaks individual verifiability against a dishonest board with careful voters, *i.e.* wins $\mathsf{Exp}^{\mathsf{verif-careful}}$. We consider an adversary $\mathcal{B} = \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$ that attacks privacy against a dishonest board with careful voters, *i.e.* plays $\mathsf{Exp}^{\mathsf{priv-careful},\beta}$:

- $\mathcal{B}_1^{O_{\mathsf{reg}}, O_{\mathsf{corr}}}(\mathsf{pk})$ first simulates $\mathcal{A}_1^{O_{\mathsf{reg}}, O_{\mathsf{corr}}}(\mathsf{pk})$, *i.e.* $\mathcal{B}$ registers and corrupts the same identities as $\mathcal{A}$, while keeping a list U_1 of the identities it registers by calling O_{reg}. $\mathcal{A}$ returns some *state*. $\mathcal{B}_1$ then calls O_{reg} another $|\mathsf{U}_1|$ times, on fresh identities that do not appear in U_1. It keeps a list U_2 of these fresh identities.

- $\mathcal{B}_2^{O_{\mathsf{vote}}^{p,c}}(state_1, \mathsf{pk})$ maintains a list L, initially empty, which will be used to record the calls to $O_{\mathsf{vote}}^{p,c}$. $\mathcal{B}_2$ first simulates $\mathcal{A}_2^{O_{\mathsf{vote}}^{v,c}}(state, \mathsf{pk})$. For each call to $O_{\mathsf{vote}}^{v,c}(id, v)$, provided $id \in \mathsf{U}_1$, $\mathcal{B}$ calls $O_{\mathsf{vote}}^{p,c}(id, v, v_{\mathsf{neutral}})$ and (potentially) obtains a ballot b. Depending on the revote policy, $\mathcal{B}_2$
  - either checks if id is already present in L, to add (id, v, b) to L only if it is not, if the revote policy is "first" (note that in this case, $O_{\mathsf{vote}}^{p,c}$ indeed returns a ballot);
  - or adds (id, v, b) to L and removes any previous entry (id, v', b') (for any v', b') from L, if the revote policy is "last" (note that in this case, $O_{\mathsf{vote}}^{p,c}$ indeed returns a ballot).

  The simulated $\mathcal{A}_2$ returns a board BB_a and a state $state_2'$. Let BB_a' be the list of ballots in BB_a that $\mathcal{A}_2$ was allowed to cast, *i.e.* $\mathsf{BB}_a' = [b \in \mathsf{BB}_a | \mathsf{open}_{id}(b) \in \mathsf{U}_1]$.

  Let then Voted be the list of the votes in L: $\mathsf{Voted} = [v \text{ for } (id, v, b) \in L]$.

  If at any point during the simulation $\mathcal{A}_2$ blocks or fails, $\mathcal{B}$ stops the simulation, and lets BB_a be the list of ballots in L, *i.e.* $[b \text{ for } (id, v, b) \in L]$.

  In any case, $\mathcal{B}_2$ calls $O_{\mathsf{vote}}^{p,c}(id_1, v_{\mathsf{neutral}}, v_1), \ldots, O_{\mathsf{vote}}^{p,c}(id_l, v_{\mathsf{neutral}}, v_l)$, where $v_1, \ldots, v_l$ are the elements of Voted, and $id_1, \ldots, id_l$ are pairwise distinct identities from U_2. Note that since all identities in L are distinct and in U_1, l is indeed smaller than $|\mathsf{U}_1| = |\mathsf{U}_2|$. $\mathcal{B}_2$ thus obtains l ballots $b_1, \ldots, b_l$, for v_{neutral} if $\beta = 0$ and $v_1, \ldots, v_l$ if $\beta = 1$. Let BB_b be the list of these ballots. Let then $\mathsf{BB} = \mathsf{BB}_a' \uplus \mathsf{BB}_b$.

  $\mathcal{B}_2$ then returns a $state_2$ indicating whether $\mathcal{A}_2$ has failed, and the board BB. Intuitively, $\mathcal{A}$ is accurately simulated only if $\beta = 0$, *i.e.* is actually provided with ballots for the votes it wanted to cast. Hence whenever $\mathcal{A}$ wins $\mathsf{Exp}^{\mathsf{verif-careful}}$, the simulated $\mathcal{A}_2$ failing can only mean that $\beta = 1$.

- $\mathcal{B}_3^{O_{\mathsf{happy}\,\mathsf{BB}}}(state_2)$ first simulates $\mathcal{A}_3^{O_{\mathsf{happy}\,\mathsf{BB}}}(state_2')$, unless $\mathcal{A}_2$ has failed, in which case it simply calls $O_{\mathsf{happy}}(\mathsf{open}_{id}(b))$ for each b occurring in BB_a'.

$\mathcal{B}_3$ then calls $O_{\mathsf{happy}_{\mathsf{BB}}}(id_i)$ for each $i \in [\![1, l]\!]$. Since BB'_a only contains ballots b such that $\mathsf{open}_{id}(b) \in \mathsf{U}_1$ (by definition), only BB_b contains ballots for the $id_i \in \mathsf{U}_2$. BB_b, by construction, contain exactly one ballot for each id_i, which is the ballot b_i produced by the (only) call to $O^{p}_{\mathsf{vote}}(id_i, *, *)$. Since the revote policy is to count the first (or last) ballot for each id, by assumption 7, $\mathsf{VerifVoter}(id_i, cred_{id_i}, \mathsf{L}_{id_i}, \mathsf{BB})$ holds, and thus the call to $O_{\mathsf{happy}_{\mathsf{BB}}}(id_i)$ adds id_i to H.
After that step, H thus necessarily contains at least $id_1, \ldots, id_l$.

- $\mathsf{Exp}^{\mathsf{priv-careful}}$ will then check that all ids occurring in V_0 or V_1 are also in H, where V_0 and V_1 are the lists it keeps, which contain the first (or last, depending on the revote policy) votes $O^{p,c}_{\mathsf{vote}}$ has been called on for each id, and H is the list of identities $O_{\mathsf{happy}_{\mathsf{BB}}}$ has successfully been called on. If H_a and H_b denote the value of the list H at this point respectively in $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}$ and $\mathsf{Exp}^{\mathsf{priv-careful},\beta}_{\mathcal{B}}$, we have $\mathsf{H}_b = \mathsf{H}_a \uplus [id_1, \ldots, id_l]$. Indeed:
  - we have established that H_b contains $id_1, \ldots, id_l$,
  - and for all $(id, cred) \in \mathsf{U}_1$, $id \in \mathsf{H}_b$ if and only if $O_{\mathsf{happy}_{\mathsf{BB}}}(id)$ has been called by $\mathcal{B}$ and $\mathsf{VerifVoter}(id, cred, \mathsf{L}_{id}, \mathsf{BB})$ succeeds, *i.e.* if and only if $O_{\mathsf{happy}_{\mathsf{BB}_a}}(id)$ was called by the simulated $\mathcal{A}$ and $\mathsf{VerifVoter}(id, cred, \mathsf{L}_{id}, \mathsf{BB})$ succeeds. Since $\mathsf{BB} = \mathsf{BB}'_a \uplus \mathsf{BB}_b$, and given the definition of BB_b, by assumption 8, $\mathsf{VerifVoter}(id, cred, \mathsf{L}_{id}, \mathsf{BB})$ is equivalent to $\mathsf{VerifVoter}(id, cred, \mathsf{L}_{id}, \mathsf{BB}'_a)$, which is itself equivalent to $\mathsf{VerifVoter}(id, cred, \mathsf{L}_{id}, \mathsf{BB}_a)$. Thus $id \in \mathsf{H}_b$ if and only if $id \in \mathsf{H}_a$.

  Therefore, the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ succeeds in $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}$ if and only if it succeeds in $\mathsf{Exp}^{\mathsf{priv-careful},\beta}_{\mathcal{B}}$.
- $\mathsf{Exp}^{\mathsf{priv-careful}}$ will then check that $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$. Considering the definition of $\mathcal{B}_2$, if this point is reached, we always have

$$\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1) = \rho(v_1, \ldots, v_l, \underbrace{v_{\mathsf{neutral}}, \ldots, v_{\mathsf{neutral}}}_{l \text{ times}})$$

  Hence this check necessarily succeeds, and $\mathsf{Exp}^{\mathsf{priv-careful}}$ computes $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U})$.
- $\mathcal{B}_4$ obtains a result r (or $\perp$, if the previous tests by $\mathsf{Exp}^{\mathsf{priv}}$ failed or the tally returned $\perp$). If
  - $\mathcal{A}_2$ blocked previously;
  - or $r = \perp$, which means, as we have established that $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$ always holds, that the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ fails, or the tally returns $\perp$;

  $\mathcal{B}$ returns $\beta' = 1$.
  Otherwise, $\mathcal{B}$ computes $\mathsf{D}(r, \mathsf{Voted})$ and:
  - if $\exists V_c.\ r = \rho(\mathsf{Voted}) * \rho(V_c)$ then $\mathcal{B}$ returns $\beta' = 1$
  - otherwise, $\mathcal{B}$ returns $\beta' = 0$.

We also construct an adversary C, who plays the game $\mathsf{Exp}^{\mathsf{ValidTally}}$.

- C_1 is identical to $\mathcal{B}_1$.
- C_2 first draws at random a bit β'', and then simulates $\mathcal{B}_2$ up to the point where $\mathcal{B}_2$ has finished simulating $\mathcal{A}_2$. It simulates each call to $O^{p,c}_{\mathsf{vote}}(id, v_0, v_1)$ $\mathcal{B}$ does by calling $O^{vt}_{\mathsf{vote}}(id, v_{\beta''})$.
- Once C_2 has finished simulating $\mathcal{B}_2$ up to the point $\mathcal{B}_2$ has simulated $\mathcal{A}_2$ and obtained a board BB'_a, it draws at random a number $k \in [\![1, l]\!]$ Recall that l is the number of different ids $O^{p,c}_{\mathsf{vote}}$ has been called on, and is also the number of additional calls to $O^{p,c}_{\mathsf{vote}}$ $\mathcal{B}$ will make. C then simulates the first $k-1$ calls to $O^{p,c}_{\mathsf{vote}}(id, v_0, v_1)$, again by calling $O^{vt}_{\mathsf{vote}}(id, v_{\beta''})$. It obtains $k-1$ ballots, and stores them in a board $\mathsf{BB}_b = [(id_1, b_1), \ldots, (id_{k-1}, b_{k-1})]$. If the kth call is $O^{p,c}_{\mathsf{vote}}(id, v_0, v_1)$, C returns $(\mathsf{BB}'_a \uplus \mathsf{BB}_b, id, v_{\beta''})$.

We will now prove that if $\mathcal{A}$ breaks $\mathsf{Exp}^{\mathsf{verif-careful}}$, then $\mathcal{B}$ breaks $\mathsf{Exp}^{\mathsf{priv-careful}}$ provided C does not break $\mathsf{Exp}^{\mathsf{ValidTally}}$.

The adversary C is polynomial, *i.e.* there exists a polynomial $q(\lambda)$ bounding its number of operations.
For any β, assume $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1)$ holds and $\mathsf{ValidTally}(\mathsf{BB}'_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}_1 \cup \mathsf{U}_2)$ does not. Thus, by assumptions 3 and 2, $\mathsf{ValidTally}(\mathsf{BB}'_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}_1 \cup [(id_1, cred_{id_1}), \ldots, (id_l, cred_{id_l})])$ does not hold either, but $\mathsf{ValidTally}(\mathsf{BB}'_a, \mathsf{sk}, \mathsf{U}_1)$ does. Hence, there exists a smallest $k \in [\![1, l]\!]$ such that $\mathsf{ValidTally}(\mathsf{BB}'_a \uplus [b_1, \ldots, b_{k-1}], \mathsf{sk}, \mathsf{U}_1 \cup \mathsf{U}'_2)$ holds and $\mathsf{ValidTally}(\mathsf{BB}'_a \uplus [b_1, \ldots, b_k], \mathsf{sk}, \mathsf{U}_1 \cup \mathsf{U}'_2 \cup [(id_k, cred_{id_k})])$ does not, where $\mathsf{U}'_2 = [(id_1, cred_{id_1}), \ldots, (id_{k-1}, cred_{id_{k-1}})]$. b_k was produced by the kth call to $O^{p,c}_{\mathsf{vote}}$ by $\mathcal{B}$: $b_k = \mathsf{Vote}(id_k, cred_{id_k}, \mathsf{pk}, v_\beta)$ for some v_β. Thus, provided C correctly guesses $\beta'' = \beta$ and k, C returns $(\mathsf{BB}'_a \uplus [b_1, \ldots, b_{k-1}], id_k, v_\beta)$ the conditions on BB in $\mathsf{Exp}^{\mathsf{ValidTally}}_{C}$ holds, and thus $\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1$. Therefore, $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1)$ holds and $\mathsf{ValidTally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}_1 \cup \mathsf{U}_2)$ does not with probability at most $2l\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]$, which is smaller than $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]$ since $l \leq q(\lambda)$.

Since BB'_a only contains ballots cast for identities in U_1 (by definition) and BB_b for identities in U_2, and $\mathsf{U}_1 \cap \mathsf{U}_2 = \emptyset$, if $\mathsf{ValidTally}(\mathsf{BB}'_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}_1 \cup \mathsf{U}_2)$, by assumption 1 we have (regardless of β)

$$r = \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}'_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}'_a, \mathsf{sk}, \mathsf{U}) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$$

In addition, by assumption 3, $\mathsf{Tally}(\mathsf{BB}'_a, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}'_a, \mathsf{sk}, \mathsf{U}_1)$. Hence

$$r = \mathsf{Tally}(\mathsf{BB}'_a, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U}).$$

Moreover, since $\mathsf{BB}_a \backslash \mathsf{BB}'_a$ contains only ballots for identities not in U_1, by assumption 2, $\mathsf{Tally}(\mathsf{BB}'_a, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1)$, and thus

$$r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U}).$$

- If $\beta = 0$: then $\mathsf{Exp}^{\mathsf{priv-careful},0}_{\mathcal{B}}(\lambda) = 1$ if and only if $\mathcal{B}_4$ (called on the result r) returns 1 in this game, which happens
  - either when $\mathcal{A}$ (simulated by $\mathcal{B}$) blocks;
  - or when $r = \perp$, *i.e.*, as already mentioned, if the tally returns $\perp$ or the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ fails in $\mathsf{Exp}^{\mathsf{priv-careful},\beta}_{\mathcal{B}}$;
  - or when $\mathcal{A}$ does not block, the previous test succeeds, and $\exists V_c.\ r = \rho(\mathsf{Voted}) * \rho(V_c)$.

  Assume $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) \Rightarrow \mathsf{ValidTally}(\mathsf{BB}'_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$, which, as we have established, holds except with probability at most $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]$.

  We have already established that the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ succeeds in $\mathsf{Exp}^{\mathsf{priv-careful},\beta}_{\mathcal{B}}$ if and only if it succeeds in $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}$.

  Let us first examine the case where $\mathcal{A}$ does not block, the tally does not return $\perp$, and the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ succeeds. Since $r \neq \perp$, $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1)$ holds. Hence, $\mathsf{ValidTally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$ also holds, and as explained previously we thus know that $r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$. Since $\beta = 0$, BB_b only contains ballots for v_{neutral}, and then by assumption 4, $\mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U}) = \rho(v_{\mathsf{neutral}})^l = \rho(v_{\mathsf{neutral}})$. Thus $r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1)$.

  The condition $\exists V_c.\ r = \rho(\mathsf{Voted}) * \rho(V_c)$ is therefore equivalent to $\exists V_c.\ \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) = \rho(\mathsf{Voted}) * \rho(V_c)$. Since, in this case,
  - $\mathcal{A}$ has been accurately simulated without blocking,
  - BB_a is the board it returns,
  - the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ succeeds in $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}$,

  this is exactly the condition under which $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}(\lambda)$ does not return 1.

  Hence, in that case, $\mathsf{Exp}^{\mathsf{priv-careful},0}_{\mathcal{B}}(\lambda) = 1$ if and only if
  - either $\mathcal{A}$ (simulated by $\mathcal{B}$) blocks;
  - or the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ fails in $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}$;
  - or the tally returns $\perp$;
  - or $\mathcal{A}$ does not block, the previous test succeeds, the tally does not return $\perp$, and $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}(\lambda)$ does not return 1.

  Since $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}(\lambda)$ also does not return 1 when $\mathcal{A}$ blocks, or when the test fails, or when the tally returns $\perp$, this implies that, unless the implication $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) \implies \mathsf{ValidTally}(\mathsf{BB}'_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$ is false, $\mathsf{Exp}^{\mathsf{priv-careful},0}_{\mathcal{B}}(\lambda) = 1$ if and only if $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}(\lambda) \neq 1$. Thus

  $$\left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv-careful},0}_{\mathcal{B}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}(\lambda) \neq 1\right]\right| \leq 2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]. \tag{4}$$

- If $\beta = 1$: then $\mathsf{Exp}^{\mathsf{priv-careful},1}_{\mathcal{B}}(\lambda) = 1$ if and only if $\mathcal{B}_4$ (called on the result r) returns 1 in this game, which happens
  - either when $\mathcal{A}$ (simulated by $\mathcal{B}$) blocks;
  - or when $r = \perp$, *i.e.*, as already mentioned, if the tally returns $\perp$ or the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ fails in $\mathsf{Exp}^{\mathsf{priv-careful},\beta}_{\mathcal{B}}$;
  - or when $\mathcal{A}$ does not block, the previous test succeeds, the tally returns $r \neq \perp$, and $\exists V_c.\ r = \rho(\mathsf{Voted}) * \rho(V_c)$.

  Assume $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) \Rightarrow \mathsf{ValidTally}(\mathsf{BB}'_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$, which, as we have established, holds except with probability at most $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ValidTally}}_{C} = 1\right]$.

  We have already established that the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ succeeds in $\mathsf{Exp}^{\mathsf{priv-careful},\beta}_{\mathcal{B}}$ if and only if it succeeds in $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}$.

  Let us first examine the case where $\mathcal{A}$ does not block, the tally does not return $\perp$, and the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ succeeds.

  As in the $\beta = 0$ case, we thus have $r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$. Since $\beta = 1$, BB_b contains ballots for $v_1, \ldots, v_l$, *i.e.* for Voted. Hence, by assumption 4, $\mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U}) = \rho(\mathsf{Voted})$, and therefore $r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) * \rho(\mathsf{Voted})$. By definition of Tally, there exists V such that $\mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) = \rho(V)$. Hence the condition $\exists V_c.\ r = \rho(\mathsf{Voted}) * \rho(V_c)$ necessarily holds.

  In addition we know that the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ succeeds in $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}$. Therefore, unless the implication $\mathsf{ValidTally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) \Rightarrow \mathsf{ValidTally}(\mathsf{BB}'_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$ is false, $\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{B}}(\lambda) = 1$ if and only if
  - either $\mathcal{A}$ (simulated by $\mathcal{B}$) blocks;
  - or the test $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ fails in $\mathsf{Exp}^{\mathsf{verif-careful}}_{\mathcal{A}}$;
  - or the tally returns $\perp$;

– or $\mathcal{A}$ does not block, the tally does not return $\perp$ and the previous test succeeds;
which means that

$$1 - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},1}(\lambda) = 1\right] \leq 2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_{C}^{\mathsf{ValidTally}} = 1\right]. \tag{5}$$

We thus have, using 4 and 5:

$$\begin{array}{rcl}
\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{verif-careful}}(\lambda) = 1\right] & = & \left(1 - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{verif-careful}}(\lambda) \neq 1\right]\right) + \left(\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},0}(\lambda) = 1\right]\right) \\
& & + \left(\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},1}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},1}(\lambda) = 1\right]\right) \\
& = & \left(\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{verif-careful}}(\lambda) \neq 1\right]\right) \\
& & + \left(\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},1}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},0}(\lambda) = 1\right]\right) + \left(1 - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},1}(\lambda) = 1\right]\right) \\
& \leq & \left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},1}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},0}(\lambda) = 1\right]\right| + 4q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_{C}^{\mathsf{ValidTally}} = 1\right]
\end{array}$$

Therefore, if $\mathcal{A}$ breaks verifiability with careful voters, *i.e.* if $\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{verif-careful}}(\lambda) = 1\right]$ is not negligible, then $\mathcal{B}$ breaks privacy with careful voters, or C breaks $\mathsf{Exp}^{\mathsf{ValidTally}}$.

The proof for the case of credential-based protocols is very similar. In that case, we assume that

$$\forall \mathcal{A}.\ \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{NM}}(\lambda) = 1\right] \text{ is negligible.}$$

Let $\mathcal{A} = \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ be an adversary that breaks individual verifiability, *i.e.* wins $\mathsf{Exp}^{\mathsf{verif-careful}}$. We construct the adversary $\mathcal{B}$, that plays $\mathsf{Exp}^{\mathsf{priv-careful}}$, as in the *id*-based case. However, contrary to the *id*-based case, $\mathcal{B}$ does not remove any ballots from BB_a, and simply uses $\mathsf{BB}'_a = \mathsf{BB}_a$. We also construct $\mathcal{D}$, that plays $\mathsf{Exp}^{\mathsf{NM}}$, and is similar to C in the *id*-based case, except that $\mathcal{D}$ simulates calls to $O_{\mathsf{vote}}^{p,c}$ by calling O_{c} instead of O_{vote}^{vt}:

- $\mathcal{D}_1$ is identical to $\mathcal{B}_1$.
- $\mathcal{D}_2$ first draws at random a bit β'', and then simulates $\mathcal{B}_2$ up to the point where $\mathcal{B}_2$ has finished simulating $\mathcal{A}_2$. It simulates each call to $O_{\mathsf{vote}}^{p,c}(id, v_0, v_1)$ $\mathcal{B}$ makes by calling $O_{\mathsf{c}}(id, v_{\beta''})$.
- Once $\mathcal{D}_2$ has finished simulating $\mathcal{B}_2$ up to the point $\mathcal{B}_2$ has simulated $\mathcal{A}_2$, $\mathcal{D}$ obtains a board BB_a (which is the same as BB_a in $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{priv-careful},\beta''}$). Since BB_a is obtained by simulating $\mathcal{A}$, which is polynomial, in all executions of $\mathcal{D}$ the length of BB_a is bounded by some polynomial $p(\lambda)$. $\mathcal{D}$ then draws at random a ballot in BB_a, and returns it.

Similarly to the proof for the *id*-based case, and keeping the same notations, it then follows that BB_a contains a ballot with a credential in U_2 the same credential with probability at most $2p(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_{\mathcal{D}}^{\mathsf{NM}} = 1\right]$. Indeed, if such a ballot exists, it cannot have been produced by a call to O_{c}. Otherwise, it would necessarily have been produced by $O_{\mathsf{c}}(id, v)$ for some id, v such that $(id, cred) \in \mathsf{U}_2$, and the only calls to this oracle simulate calls made by $\mathcal{B}$ to $O_{\mathsf{vote}}^{p,c}$ when simulating $\mathcal{A}$. Since $\mathcal{B}$ only calls $O_{\mathsf{vote}}^{p,c}$ on *id*s in U_1 when simulating $\mathcal{A}$, this is contradictory. Thus, a ballot in BB_a with a credential in U_2 cannot have been produced by O_{c}. Hence, provided $\mathcal{D}$ picked $\beta'' = \beta$, and picks the right ballot in BB_a, which happens with probability at least $\frac{1}{p(\lambda)}$, $\mathcal{D}$ wins $\mathsf{Exp}^{\mathsf{NM}}$.

Therefore, BB_a contains no ballot with a credential in U_2, except with probability at most $2p(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_{\mathcal{D}}^{\mathsf{NM}}(\lambda) = 1\right]$.

Note that by construction, all ballots in BB_b have credentials in U_2. Consequently, unless BB_a contains a ballot with a credential in U_2, by assumption 1, we have (regardless of β)

$$r = \mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}_a \uplus \mathsf{BB}_b, \mathsf{sk}, \mathsf{U}) = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U})$$

which means, by assumption 3, that

$$r = \mathsf{Tally}(\mathsf{BB}_a, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}_b, \mathsf{sk}, \mathsf{U}).$$

The remainder of the proof is the same as before, and establishes that if $\mathcal{A}$ breaks verifiability with careful voters, *i.e.* if $\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{verif-careful}}(\lambda) = 1\right]$ is not negligible, then $\mathcal{B}$ breaks privacy with careful voters, or $\mathcal{D}$ breaks $\mathsf{Exp}^{\mathsf{NM}}$, which proves the claim.

Note that, in these reduction, for each *id* in U_1, $\mathcal{B}$ makes at most as many calls to $O_{\mathsf{vote}}^{p,c}$ as $\mathcal{A}$ makes to $O_{\mathsf{vote}}^{v,c}$. In addition, for each *id* in U_2, $\mathcal{B}$ makes at most one call to $O_{\mathsf{vote}}^{p,c}$.

Thus, the exact same proof proves that the result also holds if both the games $\mathsf{Exp}^{\mathsf{priv-careful}}$ and $\mathsf{Exp}^{\mathsf{verif-careful}}$ are modified to prevent revote, by allowing only one call to $O_{\mathsf{vote}}^{p,c}/O_{\mathsf{vote}}^{v,c}$ for each *id*. $\square$

$$\begin{array}{l}
\underline{\mathsf{Exp}^{\mathsf{ind},\beta}_{\mathcal{A}}(\lambda)} \\
(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda) \\
\mathsf{U},\mathsf{CU} \leftarrow [] \\
\mathit{state} \leftarrow \mathcal{A}_1^{O_{\mathrm{reg}},O_{\mathrm{corr}}}(\mathsf{pk}) \\
\mathsf{L} \leftarrow [] \\
\beta' \leftarrow \mathcal{A}_2^{O^i_{\mathrm{c}},O^i_{\mathrm{d}}}(\mathit{state},\mathsf{pk}) \\
\textbf{return } \beta' \\
\quad \mathcal{A}_2 \text{ is only allowed one call to } O^i_{\mathrm{d}}
\end{array}
\qquad
\begin{array}{l}
\underline{O^i_{\mathrm{c}}(\mathit{id},\mathit{cred},v_0,v_1)} \\
b \leftarrow \mathsf{Vote}(\mathit{id},\mathit{cred},\mathsf{pk},v_\beta) \\
\mathsf{L} \leftarrow \mathsf{L}\|\mathsf{extract}(b) \\
\textbf{return } b
\end{array}
\qquad
\begin{array}{l}
\underline{O^i_{\mathrm{d}}(\mathsf{bL})} \\
\mathsf{dL} \leftarrow [] \\
\textbf{for } b \in \mathsf{bL} \textbf{ do} \\
\quad \textbf{if } \mathsf{extract}(b) \notin \mathsf{L} \textbf{ then} \\
\quad\quad \mathsf{dL} \leftarrow \mathsf{dL}\|(\mathsf{open}_{\mathit{cred}}(b,\mathsf{sk},\mathsf{U}),\mathsf{open}(b,\mathsf{sk},\mathsf{U})) \\
\quad \textbf{else} \\
\quad\quad \mathsf{dL} \leftarrow \mathsf{dL}\|\perp \\
\textbf{return } \mathsf{dL}
\end{array}$$

Figure 15: IND − CCA-**like property on the ballot creation function**

Appendix C CASE STUDY

C.1 Assumptions

To prove that the protocols we study satisfy the different privacy properties, we will in some cases assume that no adversary wins $\mathsf{Exp}^{\mathsf{NM}}$ (presented on Figure 7) with non-negligible probability. We will also in some cases use the assumption that the ballot creation function has a property similar to the IND − CCA property *i.e.* that

$$\forall \mathcal{A}.\ \left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},0}_{\mathcal{A}}(\lambda)=1\right]-\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},1}_{\mathcal{A}}(\lambda)=1\right]\right| \text{ is negligible}$$

where $\mathsf{Exp}^{\mathsf{ind}}$ is defined on Figure 15. This definition assumes a function $\mathsf{open}(b,\mathsf{sk},\mathsf{U})$ that returns the vote contained in the ballot b, *i.e.* for all election keys $(\mathsf{pk},\mathsf{sk})$ and list of users and credentials U,

$$\forall \mathit{id},\mathit{cred},v.\ \mathsf{open}(\mathsf{Vote}(\mathit{id},\mathit{cred},\mathsf{pk},v),\mathsf{sk},\mathsf{U})=v.$$

It also assumes a function $\mathsf{extract}(b)$, that represents the ciphertext part in b. Typically, if b has the form (id,c) where id is the identity of the voter and c the ciphertext containing the vote, we have $\mathsf{open}_{\mathit{id}}(b)=\mathit{id}$ and $\mathsf{extract}(b)=c$. This corresponds to the case of Helios and Belenios in our case study. For the other protocols we study, $\mathsf{extract}(b)=b$.

C.2 Proofs

C.2.1 Civitas is private for $\mathsf{Exp}^{\mathsf{priv}}$.

Theorem C.1. *Assuming no adversary wins* $\mathsf{Exp}^{\mathsf{ind}}$ *nor* $\mathsf{Exp}^{\mathsf{NM}}$ *with non-negligible probability, Civitas is private for* $\mathsf{Exp}^{\mathsf{priv}}$.

Proof. Let $\mathcal{A}=\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3$ be an adversary that wins $\mathsf{Exp}^{\mathsf{priv}}$. We consider an adversary $\mathcal{B}=\mathcal{B}_1,\mathcal{B}_2$ that plays $\mathsf{Exp}^{\mathsf{ind}}$:

- $\mathcal{B}_1^{O_{\mathrm{reg}},O_{\mathrm{corr}}}(\mathsf{pk})$ first simulates $\mathcal{A}_1^{O_{\mathrm{reg}},O_{\mathrm{corr}}}(\mathsf{pk})$, *i.e.* $\mathcal{B}$ registers and corrupts the same identities as $\mathcal{A}$, while keeping lists U_1, CU_1 of the identities it declares and corrupts by calling O_{reg} and O_{corr}. $\mathcal{A}$ returns some state'_1. $\mathcal{B}_1$ then corrupts each user $\mathcal{A}_1$ has registered, *i.e.*, $\mathcal{B}_1$ calls $O_{\mathrm{corr}}(\mathit{id})$ for each id $\mathcal{A}_1$ has declared, and stores each id's credential in a list CU_2.
- $\mathcal{B}_2^{O^i_{\mathrm{c}},O^i_{\mathrm{d}}}(\mathit{state}_1,\mathsf{pk})$ maintains lists $\mathsf{V}_0,\mathsf{V}_1,\mathsf{BB}$, initially empty, which will be used to simulate the lists with the same name in $\mathsf{Exp}^{\mathsf{priv}}$. $\mathcal{B}_2$ will also use lists hL,cL, initially empty.
  $\mathcal{B}_2$ first simulates $\mathcal{A}_2^{O^p_{\mathrm{vote}},O_{\mathrm{cast}}}(\mathit{state}'_1,\mathsf{pk})$:
  - for each call to $O^p_{\mathrm{vote}}(\mathit{id},v_0,v_1)$, provided $\mathit{id}\in\mathsf{U}_1\backslash\mathsf{CU}_1$, $\mathcal{B}$ checks whether id is already present in $\mathsf{V}_0,\mathsf{V}_1$. Provided it is not, $\mathcal{B}$ then retrieves id's credential $\mathit{cred}_{\mathit{id}}$ from CU_2, and calls $O^i_{\mathrm{c}}(\mathit{id},\mathit{cred}_{\mathit{id}},v_0,v_1)$. $\mathcal{B}$ obtains a ballot b. $\mathcal{B}$ then appends b to BB and hL, (id,v_0) to V_0, (id,v_1) to V_1. Finally $\mathcal{B}$ returns b to the simulated $\mathcal{A}$.
  - for each call to $O_{\mathrm{cast}}(\mathit{id},b)$, provided $\mathit{id}\in\mathsf{CU}_1$ and $\mathsf{Valid}(\mathit{id},b,\mathsf{BB},\mathsf{pk})$, $\mathcal{B}$ appends b to BB, and to cL.

  We write $\mathsf{cL}\backslash\mathsf{hL}$ (resp. $\mathsf{cL}\cap\mathsf{hL}$) the sublist of cL (in the same order) of ballots that do not occur (resp. do occur) in hL.
- $\mathcal{B}_2$ then computes $\rho(\mathsf{V}_0)$, $\rho(\mathsf{V}_1)$, and check they are equal.
- $\mathcal{B}_2$ calls $O^i_{\mathrm{d}}(\mathsf{cL}\backslash\mathsf{hL})$, and obtains a list L of pairs of credentials and votes. $\mathcal{B}_2$ then computes the list L' of the first vote for each credential in L. Note that, by construction, no ballot in $\mathsf{cL}\backslash\mathsf{hL}$ has been generated by a call to O^i_{c}, which means that O^i_{d} accepts to open all ballots in $\mathsf{cL}\backslash\mathsf{hL}$.
- $\mathcal{B}_2$ computes $r=\rho(\mathsf{V}_0)*\rho(L')$, calls $\mathcal{A}_3$ on r, and obtains a bit β'. $\mathcal{B}_2$ returns β'.

Note that the lists BB, V_0, V_1 in $\mathcal{B}$ are equal to the lists of the same name in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}$. Then, by construction, $\mathcal{A}_2$ is always accurately simulated by $\mathcal{B}_2$, *i.e.* it is called on the same inputs, shown the same board, and provided with the same oracles as what would happen in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}$.

We also construct an adversary C, that plays the game $\mathsf{Exp}^{\mathsf{NM}}$.

- C_1 simulates $\mathcal{A}_1$, similarly to $\mathcal{B}_1$. However, unlike $\mathcal{B}_1$, it stops there, and does not corrupt all identities.
- C_2 first draws at random a bit β'', Similarly to $\mathcal{B}_2$, C_2 then simulates $\mathcal{A}_2$ and uses lists BB, cL, hL, V:
  - for each call to $O^{p}_{\mathsf{vote}}(id, v_0, v_1)$, provided $id \in \mathsf{U}_1 \backslash \mathsf{CU}_1$, C calls $O_{\mathsf{c}}(id, v_{\beta''})$. C obtains a ballot b and appends b to BB and hL, and $(id, v_{\beta''})$ to V. Finally C returns b to the simulated $\mathcal{A}$.
  - for each call to $O_{\mathsf{cast}}(id, b)$, provided $id \in \mathsf{CU}_1$ and $\mathsf{Valid}(id, b, \mathsf{BB}, \mathsf{pk})$, C appends b to BB and cL.
- Once C_2 has finished simulating $\mathcal{A}_2$, it draws at random an element of cL and returns it.

Note that the lists BB, cL, hL are the same for C in $\mathsf{Exp}^{\mathsf{NM}}$ at the point C_2 returns, and for $\mathcal{B}$ in $\mathsf{Exp}^{\mathsf{ind}}$ at the point $\mathcal{B}_2$ returns. Similarly V in C is the same as $\mathsf{V}_{\beta''}$ in $\mathcal{B}$. Let us also notice that hL in C is equal to the list L in the game $\mathsf{Exp}^{\mathsf{NM}}$.

We will now prove that if $\mathcal{A}$ breaks privacy, then $\mathcal{B}$ wins $\mathsf{Exp}^{\mathsf{ind}}$ provided C does not win $\mathsf{Exp}^{\mathsf{NM}}$.

The adversary C is polynomial, *i.e.* there exists a polynomial $q(\lambda)$ bounding its number of operations. $q(\lambda)$ necessarily also bounds the length of the list cL that C uses.

For any β, assume $\mathsf{cL} \backslash \mathsf{hL}$ contains a ballot b such that there exists a honest $(*, cred) \in \mathsf{U} \backslash \mathsf{CU}$ such that $\mathsf{open}_{cred}(b, \mathsf{sk}, \mathsf{U}) = cred$ (note that U,CU in $\mathsf{Exp}_{C}^{\mathsf{NM}}$ are equal to U_1, CU_1 in $\mathcal{B}$). Thus, provided C correctly guesses $\beta'' = \beta$, and chooses this b from cL, the condition $b \notin \mathsf{L} \ \wedge \ \exists (*, cred) \in \mathsf{U} \backslash \mathsf{CU}.\ \mathsf{open}_{cred}(b, \mathsf{sk}, \mathsf{U}) = cred$ in $\mathsf{Exp}_{C}^{\mathsf{NM}}$ holds. Indeed, since $b \notin \mathsf{hL}$, $b \notin \mathsf{L}$. Thus $\mathsf{Exp}_{C}^{\mathsf{NM}} = 1$.

Therefore, such a ballot b exists with probability at most $2|\mathsf{cL}|\,\mathrm{P}\left[\mathsf{Exp}_{C}^{\mathsf{NM}} = 1\right]$, which is smaller than $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_{C}^{\mathsf{NM}} = 1\right]$ since $|\mathsf{cL}| \leq q(\lambda)$.

For any β, $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda) = 1$ if and only if $\mathcal{B}_2$ returns 1 in this game.

Assume $\mathsf{cL} \backslash \mathsf{hL}$ does not contain any ballot associated with a honest credential (*i.e.* a credential in $\mathsf{U}_1 \backslash \mathsf{CU}_1$), which, as we have established, holds except with probability at most $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_{C}^{\mathsf{NM}} = 1\right]$.

Let us then show that $\mathcal{A}_3$ is accurately simulated by $\mathcal{B}_2$, *i.e.* it is simulated by $\mathcal{B}_2$ only when it is called in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}$, that is, when $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$; and it is provided with the actually tally of the board BB (which $\mathcal{A}_2$ interacted with).

Indeed, by construction of $\mathcal{B}_2$, $\mathsf{BB} = \mathsf{hL} \uplus \mathsf{cL}$ is an interleaving of the ballots from hL and cL.

By assumption, $\mathsf{cL} \backslash \mathsf{hL}$ contains no ballot from honest credentials, while by construction hL (and thus $\mathsf{cL} \cap \mathsf{hL}$) only contains ballots from honest credentials. Hence, the list BB_d of ballots in BB associated with dishonest credentials is $\mathsf{cL} \backslash \mathsf{hL}$ (in that order). The list BB_h of ballots in BB associated with honest credentials is an interleaving of the ballots from the lists hL and $\mathsf{cL} \cap \mathsf{hL}$. However, by construction, hL contains at most one ballot for each credential. Thus, BB_h also contains at most one distinct ballot (of which there can be several copies) for each credential. The list of distinct ballots in BB_h (not necessarily in the same order) is thus hL.

The revote policy specified for Civitas is to count only the first ballot corresponding to each credential. Since BB can be separated into the lists BB_h and BB_d (whose ballots do not share any credential, by definition), and since the ballots of each of these two lists occur in the same order in BB, we have

$$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{BB}_h, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}_d, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{cL} \backslash \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1).$$

hL contains ballots for either the votes in V_0 or those in V_1, depending on β. Since at that point $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$, we have $\rho(\mathsf{V}_0) = \mathsf{Tally}(\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1)$. In addition, the oracle O_{d}^{i} returns the list L of the credentials and votes of each ballot in $\mathsf{cL} \backslash \mathsf{hL}$. Since L' is the list of the first vote for each credential in L, we thus have $\mathsf{Tally}(\mathsf{cL} \backslash \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) = \rho(L')$. Therefore, $\rho(V_0) * \rho(L')$, which is the result computed by $\mathcal{B}_2$, is indeed $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}_1)$, which concludes the proof that $\mathcal{A}_3$ is accurately simulated by $\mathcal{B}_2$.

Hence, unless $\mathsf{cL} \backslash \mathsf{hL}$ contains a ballot associated with a honest credential, $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda) = 1$ if and only if the accurately simulated $\mathcal{A}_3$ returns 1, *i.e.* if and only if $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}(\lambda) = 1$.

Thus

$$\left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}(\lambda) \neq 1\right]\right| \leq 2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_{C}^{\mathsf{NM}} = 1\right]. \tag{6}$$

We thus have:

$$
\begin{aligned}
\left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{A}}(\lambda)=1\right]-\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{A}}(\lambda)=1\right]\right| &= \left|\left(\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{A}}(\lambda)=1\right]-\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},0}_{\mathcal{B}}(\lambda)=1\right]\right)+\left(\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},0}_{\mathcal{B}}(\lambda)=1\right]-\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},1}_{\mathcal{B}}(\lambda)=1\right]\right)\right.\\
&\quad\left.+\left(\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},1}_{\mathcal{B}}(\lambda)=1\right]-\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{A}}(\lambda)=1\right]\right)\right|\\
&\le \left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{A}}(\lambda)=1\right]-\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},0}_{\mathcal{B}}(\lambda)=1\right]\right|+\left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},1}_{\mathcal{B}}(\lambda)=1\right]-\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},0}_{\mathcal{B}}(\lambda)=1\right]\right|\\
&\quad+\left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},1}_{\mathcal{B}}(\lambda)=1\right]-\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{A}}(\lambda)=1\right]\right|\\
&\le \left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},0}_{\mathcal{B}}(\lambda)=1\right]-\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},1}_{\mathcal{B}}(\lambda)=1\right]\right|+4q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}^{\mathsf{NM}}_{C}=1\right]
\end{aligned}
$$

Therefore, if $\mathcal{A}$ breaks privacy, *i.e.* if $\left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{A}}(\lambda)=1\right]-\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{A}}(\lambda)=1\right]\right|$ is not negligible, then $\mathcal{B}$ breaks $\mathsf{Exp}^{\mathsf{ind}}$, or C breaks $\mathsf{Exp}^{\mathsf{NM}}$.

□

C.2.2 *Civitas is private for* $\mathsf{Exp}^{\mathsf{priv-careful}}$.

Theorem C.2. *Assuming no adversary wins* $\mathsf{Exp}^{\mathsf{ind}}$ *nor* $\mathsf{Exp}^{\mathsf{NM}}$ *with non-negligible probability, Civitas is private for* $\mathsf{Exp}^{\mathsf{priv-careful}}$.

Proof. Let $\mathcal{A}=\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3,\mathcal{A}_4$ be an adversary that wins $\mathsf{Exp}^{\mathsf{priv-careful}}$. We consider an adversary $\mathcal{B}=\mathcal{B}_1,\mathcal{B}_2,\mathcal{B}_3$ that plays $\mathsf{Exp}^{\mathsf{ind}}$:

- $\mathcal{B}_1^{O_{\mathsf{reg}},O_{\mathsf{corr}}}(\mathsf{pk})$ first simulates $\mathcal{A}_1^{O_{\mathsf{reg}},O_{\mathsf{corr}}}(\mathsf{pk})$, *i.e.* $\mathcal{B}$ registers and corrupts the same identities as $\mathcal{A}$, while keeping lists U_1, CU_1 of the identities it declares and corrupts by calling O_{reg} and O_{corr}. $\mathcal{A}$ returns some $state'_1$. $\mathcal{B}_1$ then corrupts each user $\mathcal{A}_1$ has registered, *i.e.*, $\mathcal{B}_1$ calls $O_{\mathsf{corr}}(id)$ for each id $\mathcal{A}_1$ has declared, and stores each id's credential in a list CU_2.

- $\mathcal{B}_2^{O^i_{\mathsf{c}},O^i_{\mathsf{d}}}(state_1,\mathsf{pk})$ maintains lists $\mathsf{V}_0,\mathsf{V}_1,\mathsf{BB}$, initially empty, which will be used to simulate the lists with the same name in $\mathsf{Exp}^{\mathsf{priv-careful}}$. $\mathcal{B}_2$ will also use lists hL, H, and a list L_{id} for each $id\in\mathsf{U}_1$, all of them initially empty.
  $\mathcal{B}_2$ first simulates $\mathcal{A}_2^{O^{p,c}_{\mathsf{vote}}}(state'_1,\mathsf{pk})$. For each call to $O^{p,c}_{\mathsf{vote}}(id,v_0,v_1)$, provided $id\in\mathsf{U}_1\backslash\mathsf{CU}_1$, $\mathcal{B}$ checks whether id is already present in $\mathsf{V}_0,\mathsf{V}_1$. Provided it is not, $\mathcal{B}$ then retrieves id's credential $cred_{id}$ from CU_2, and calls $O^i_{\mathsf{c}}(id,cred_{id},v_0,v_1)$. $\mathcal{B}$ obtains a ballot b. $\mathcal{B}$ then appends (id,v_0) to V_0, (id,v_1) to V_1, b to hL, and b to L_{id}. Finally $\mathcal{B}$ returns b to $\mathcal{A}$.
  $\mathcal{A}_2$ eventually returns a board BB. We write $\mathsf{BB}\backslash\mathsf{hL}$ (resp. $\mathsf{BB}\cap\mathsf{hL}$) the sublist of BB (in the same order) of ballots that do not occur (resp. do occur) in hL.
- $\mathcal{B}_2$ calls $O^i_{\mathsf{d}}(\mathsf{BB}\backslash\mathsf{hL})$, and obtains a list L of pairs of credentials and votes. $\mathcal{B}_2$ then computes the list L' of the first vote for each credential in L. Note that, by construction, no ballot in $\mathsf{BB}\backslash\mathsf{hL}$ has been generated by a call to O^i_{c}, which means that O^i_{d} accepts to open all ballots in $\mathsf{BB}\backslash\mathsf{hL}$.
- $\mathcal{B}_2$ then simulates $\mathcal{A}_3^{O_{\mathsf{happy_{BB}}}}$. For each call to $O_{\mathsf{happy_{BB}}}(id)$, provided $id\in\mathsf{U}_1\backslash\mathsf{CU}_1$, following the specification of the voter verification for Civitas, $\mathcal{B}$ checks whether the first ballot b in L_{id} is in BB (note that by definition of the oracles, L_{id} actually only contains a single element). If so, $\mathcal{B}$ appends id to H. In any case, $\mathcal{B}$ then resumes the execution of $\mathcal{A}_3$.
- $\mathcal{B}_2$ then computes $\rho(\mathsf{V}_0)$, $\rho(\mathsf{V}_1)$, and checks that they are equal, and that every id occurring in $\mathsf{V}_0,\mathsf{V}_1$ is also an element of H. If so, $\mathcal{B}_2$ computes $r=\rho(\mathsf{V}_0)*\rho(L')$. Otherwise, $\mathcal{B}_2$ lets $r=\perp$.
- $\mathcal{B}_2$ then calls $\mathcal{A}_4$ on r, and obtains a bit β'. $\mathcal{B}_2$ returns β'.

Note that the lists BB, V_0, V_1, H in $\mathcal{B}$ are equal to the lists of the same name in $\mathsf{Exp}^{\mathsf{priv-careful},\beta}_{\mathcal{A}}$. Then, by construction, $\mathcal{A}_2$ is always accurately simulated by $\mathcal{B}_2$, *i.e.* it is called on the same inputs, and provided with the same oracles as what would happen in $\mathsf{Exp}^{\mathsf{priv},\beta}_{\mathcal{A}}$. Given the specification of VerifVoter for Civitas, $\mathcal{A}_3$ is also accurately simulated by $\mathcal{B}_2$.

We also construct an adversary C, who plays the game $\mathsf{Exp}^{\mathsf{NM}}$.

- C_1 simulates $\mathcal{A}_1$, similarly to $\mathcal{B}_1$. However, unlike $\mathcal{B}_1$, it stops there, and does not corrupt all identities.
- C_2 first draws at random a bit β'', Similarly to $\mathcal{B}_2$, C_2 then simulates $\mathcal{A}_2$ and uses lists hL, V. For each call to $O^{p,c}_{\mathsf{vote}}(id,v_0,v_1)$, provided $id\in\mathsf{U}_1\backslash\mathsf{CU}_1$ does not already occur in V, C calls $O_{\mathsf{c}}(id,v_{\beta''})$. C obtains a ballot b and appends $(id,v_{\beta''})$ to V, and b to hL. Finally C returns b.
- $\mathcal{A}_2$ eventually returns a board BB. C then draws at random a ballot of BB and returns it.

Note that the lists BB, hL, are the same for C in $\mathsf{Exp}^{\mathsf{NM}}$ at the point C_2 returns, and for $\mathcal{B}$ in $\mathsf{Exp}^{\mathsf{ind}}$ at the point the simulated $\mathcal{A}_2$ returns. Similarly V in C is the same as $\mathsf{V}_{\beta''}$ in $\mathcal{B}$. Let us also notice that hL in C is equal to the list L in the game $\mathsf{Exp}^{\mathsf{NM}}$.

We will now prove that if $\mathcal{A}$ wins $\mathsf{Exp}^{\mathsf{priv-careful}}$, then $\mathcal{B}$ wins $\mathsf{Exp}^{\mathsf{ind}}$ provided C does not win $\mathsf{Exp}^{\mathsf{NM}}$.

The adversary C is polynomial, *i.e.* there exists a polynomial $q(\lambda)$ bounding its number of operations. $q(\lambda)$ necessarily also bounds the length of the board BB that C computes.

For any β, assume $\mathsf{BB}\backslash\mathsf{hL}$ contains a ballot b such that there exists a honest $(*, cred) \in \mathsf{U}\backslash\mathsf{CU}$ such that $\mathsf{open}_{cred}(b, \mathsf{sk}, \mathsf{U}) = cred$ (note that U,CU in $\mathsf{Exp}^{\mathsf{NM}}$ are equal to U_1, CU_1 in $\mathcal{B}$). Thus, provided C correctly guesses $\beta'' = \beta$, and chooses this b from BB, the condition $b \notin \mathsf{L} \wedge \exists(id, cred) \in \mathsf{U}\backslash\mathsf{CU}.\ \mathsf{open}_{cred}(b, \mathsf{sk}, \mathsf{U}) = cred$ in $\mathsf{Exp}_C^{\mathsf{NM}}$ holds. Indeed, since $b \notin \mathsf{hL}$, $b \notin \mathsf{L}$. Thus $\mathsf{Exp}_C^{\mathsf{NM}} = 1$.

Therefore, such a ballot b exists with probability at most $2|\mathsf{BB}|\,\mathrm{P}\left[\mathsf{Exp}_C^{\mathsf{NM}} = 1\right]$, which is smaller than $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_C^{\mathsf{NM}} = 1\right]$ since $|\mathsf{BB}| \leq q(\lambda)$.

For any β, $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda) = 1$ if and only if $\mathcal{B}_2$ returns 1 in this game.

Assume $\mathsf{BB}\backslash\mathsf{hL}$ does not contain any ballot associated with a honest credential (*i.e.* a credential in $\mathsf{U}_1\backslash\mathsf{CU}_1$), which, as we have established, holds except with probability at most $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_C^{\mathsf{NM}} = 1\right]$.

Let us then show that $\mathcal{A}_4$ is accurately simulated by $\mathcal{B}_2$, *i.e.* it is provided by $\mathcal{B}_2$ with the same input as when it is called in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}$, that is the actual tally of the board BB (which $\mathcal{A}_2$ returned) if $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$) and if $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$; and $\perp$ otherwise.

It is clear from the definition of $\mathcal{B}$ that when either the equality condition $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$ or the voter verification condition $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ do not hold, $\mathcal{A}_4$ is indeed given $\perp$ as an argument.

Let us now study the case where both these conditions are met. Among the ballots in BB, some are also in the list hL of ballots created by the oracle O_{c}^{i}. We will thus see BB as an interleaving $(\mathsf{BB}\backslash\mathsf{hL}) \uplus (\mathsf{BB} \cap \mathsf{hL})$ of the ballots in $\mathsf{BB}\backslash\mathsf{hL}$ and $\mathsf{BB} \cap \mathsf{hL}$ (keeping the same order within each of these two lists).

By assumption, $\mathsf{BB}\backslash\mathsf{hL}$ contains no ballot for honest credentials, whereas by construction hL (and thus $\mathsf{BB} \cap \mathsf{hL}$) only contains ballots for honest credentials. In addition, by construction, hL contains at most one ballot for each credential. Thus, $\mathsf{BB} \cap \mathsf{hL}$ also contains at most one distinct ballot (of which there can be several copies) for each credential. The list of distinct ballots with honest credentials in BB (not necessarily in the same order) is thus a subset of hL.

Moreover, we assumed the voter verifications succeeded, *i.e.* $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ holds. By construction, each $b \in \mathsf{hL}$ was added when simulating a call to $O_{\mathsf{vote}}^{p}(id, v_0, v_1)$ for some honest $id \in \mathsf{U}_1\backslash\mathsf{CU}_1$ and some v_0, v_1. Therefore $(id, v_0) \in \mathsf{V}_0$, $(id, v_1) \in \mathsf{V}_1$, and $\mathsf{L}_{id} = [b]$. Hence, $id \in \mathsf{H}$. By definition of $\mathcal{B}$ (which performs the voter verifications), this means that b is in BB.

Consequently, all ballots in hL are also in BB. Thus the list of distinct ballots with honest credentials in BB (not necessarily in the same order) is actually equal to hL.

The revote policy specified for Civitas is to count only the first ballot corresponding to each credential. Since BB can be separated into the lists $\mathsf{BB}\backslash\mathsf{hL}$ and $\mathsf{BB} \cap \mathsf{hL}$ (whose ballots do not share any credential, by assumption), and since the ballots of each of these two lists occur in the same order in BB, we have

$$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{BB} \cap \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}\backslash\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1).$$

Then, by the previous observation that the list of distinct ballots in $\mathsf{BB} \cap \mathsf{hL}$ is hL (regardless of the order, which does not matter since all ballots in hL have distinct credentials by construction):

$$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}\backslash\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1).$$

By construction, hL contains ballots for either the votes in V_0 or those in V_1, depending on β. Since at that point $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$, we have $\rho(\mathsf{V}_0) = \mathsf{Tally}(\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1)$. In addition, the oracle O_{d}^{i} returns the list L of the credentials and votes of each ballot in $\mathsf{BB}\backslash\mathsf{hL}$. Since L' is the list of the first vote for each credential in L, we thus have $\mathsf{Tally}(\mathsf{BB}\backslash\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) = \rho(L')$. Therefore, $\rho(\mathsf{V}_0) * \rho(L')$, which is the result computed by $\mathcal{B}_2$, is indeed $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}_1)$, which concludes the proof that $\mathcal{A}_4$ is accurately simulated by $\mathcal{B}_2$.

Hence, unless $\mathsf{cL}\backslash\mathsf{hL}$ contains a ballot associated with a honest credential, $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda) = 1$ if and only if the accurately simulated $\mathcal{A}_4$ returns 1, *i.e.* if and only if $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},\beta}(\lambda) = 1$.

Thus

$$\left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},\beta}(\lambda) \neq 1\right]\right| \leq 2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_C^{\mathsf{NM}} = 1\right]. \tag{7}$$

We thus have:

$$\begin{aligned}
&\left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},1}(\lambda) = 1\right]\right| \\
&\quad = \quad \left|\left(\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},0}(\lambda) = 1\right]\right) + \left(\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},1}(\lambda) = 1\right]\right)\right. \\
&\qquad\quad \left. + \left(\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},1}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},1}(\lambda) = 1\right]\right)\right| \\
&\quad \leq \quad \left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},0}(\lambda) = 1\right]\right| + \left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},1}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},0}(\lambda) = 1\right]\right| \\
&\qquad\quad + \left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},1}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},1}(\lambda) = 1\right]\right| \\
&\quad \leq \quad \left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},1}(\lambda) = 1\right]\right| + 4q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_C^{\mathsf{NM}} = 1\right]
\end{aligned}$$

Therefore, if $\mathcal{A}$ breaks privacy with careful voters, *i.e.* if $\left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},0}(\lambda)=1\right]-\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},1}(\lambda)=1\right]\right|$ is not negligible, then $\mathcal{B}$ breaks $\mathsf{Exp}^{\mathsf{ind}}$, or C breaks $\mathsf{Exp}^{\mathsf{NM}}$.

□

C.2.3 *Belenios is private for* $\mathsf{Exp}^{\mathsf{priv-careful}}$.

THEOREM C.3. *Assuming no adversary wins* $\mathsf{Exp}^{\mathsf{ind}}$ *nor* $\mathsf{Exp}^{\mathsf{NM}}$ *with non-negligible probability, Belenios is private for* $\mathsf{Exp}^{\mathsf{priv-careful}}$.

PROOF. Let $\mathcal{A}=\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3,\mathcal{A}_4$ be an adversary that wins $\mathsf{Exp}^{\mathsf{priv-careful}}$. We consider an adversary $\mathcal{B}=\mathcal{B}_1,\mathcal{B}_2,\mathcal{B}_3$ that plays $\mathsf{Exp}^{\mathsf{ind}}$:

- $\mathcal{B}_1^{O_{\mathsf{reg}},O_{\mathsf{corr}}}(\mathsf{pk})$ first simulates $\mathcal{A}_1^{O_{\mathsf{reg}},O_{\mathsf{corr}}}(\mathsf{pk})$, *i.e.* $\mathcal{B}$ registers and corrupts the same identities as $\mathcal{A}$, while keeping lists U_1, CU_1 of the identities it declares and corrupts by calling O_{reg} and O_{corr}. $\mathcal{A}$ returns some $state'_1$. $\mathcal{B}_1$ then corrupts each user $\mathcal{A}_1$ has registered, *i.e.*, $\mathcal{B}_1$ calls $O_{\mathsf{corr}}(id)$ for each id $\mathcal{A}_1$ has declared, and stores each id's credential in a list CU_2.
- $\mathcal{B}_2^{O^i_{\mathsf{c}},O^i_{\mathsf{d}}}(state_1,\mathsf{pk})$ maintains lists $\mathsf{V}_0,\mathsf{V}_1,\mathsf{BB}$, initially empty, which will be used to simulate the lists with the same name in $\mathsf{Exp}^{\mathsf{priv-careful}}$. $\mathcal{B}_2$ will also use lists hL, H, and a list L_{id} for each $id\in\mathsf{U}_1$, all of them initially empty.
  $\mathcal{B}_2$ first simulates $\mathcal{A}_2^{O^{p,c}_{\mathsf{vote}}}(state'_1,\mathsf{pk})$. For each call to $O^{p,c}_{\mathsf{vote}}(id,v_0,v_1)$, provided $id\in\mathsf{U}_1\backslash\mathsf{CU}_1$, $\mathcal{B}$ retrieves id's credential $cred_{id}$ from CU_2, and calls $O^i_{\mathsf{c}}(id,cred_{id},v_0,v_1)$. $\mathcal{B}$ obtains a ballot b. $\mathcal{B}$ then removes from V_0 and V_1 all elements of the form $(id,*)$, and appends (id,v_0) to V_0, (id,v_1) to V_1, b to hL, and b to L_{id}. Finally $\mathcal{B}$ returns b to $\mathcal{A}$.
  $\mathcal{A}_2$ eventually returns a board BB. We write $\mathsf{BB}\backslash\mathsf{hL}$ (resp. $\mathsf{BB}\cap\mathsf{hL}$) the sublist of BB (in the same order) of ballots that do not occur (resp. do occur) in hL.
- $\mathcal{B}_2$ calls $O^i_{\mathsf{d}}(\mathsf{BB}\backslash\mathsf{hL})$, and obtains a list L of pairs of credentials and votes. $\mathcal{B}_2$ then computes the list L' of the last vote for each credential in L. We will see later that under the right assumptions, O^i_{d} accepts to open all (valid) ballots in $\mathsf{BB}\backslash\mathsf{hL}$.
- $\mathcal{B}_2$ then simulates $\mathcal{A}_3^{O_{\mathsf{happyBB}}}$. For each call to $O_{\mathsf{happy}_{\mathsf{BB}}}(id)$, provided $id\in\mathsf{U}_1\backslash\mathsf{CU}_1$, following the specification of the voter verification for Belenios, $\mathcal{B}$ retrieves id's credential $cred$ from CU_2, and checks whether the last ballot b in L_{id} is the last ballot associated to $cred$ (*i.e.* signed by $cred$) in BB. If so, $\mathcal{B}$ appends id to H. In any case, $\mathcal{B}$ then resumes the execution of $\mathcal{A}_3$.
- $\mathcal{B}_2$ then computes $\rho(\mathsf{V}_0)$, $\rho(\mathsf{V}_1)$, and checks that they are equal, and that every id occurring in $\mathsf{V}_0,\mathsf{V}_1$ is also an element of H. If so, $\mathcal{B}_2$ computes $r=\rho(\mathsf{V}_0)*\rho(L')$. Otherwise, $\mathcal{B}_2$ lets $r=\perp$.
- $\mathcal{B}_2$ then calls $\mathcal{A}_4$ on r, and obtains a bit β'. $\mathcal{B}_2$ returns β'.

Note that the lists BB, V_0, V_1, H in $\mathcal{B}$ are equal to the lists of the same name in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},\beta}$. Then, by construction, $\mathcal{A}_2$ is always accurately simulated by $\mathcal{B}_2$, *i.e.* it is called on the same inputs, and provided with the same oracles as what would happen in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}$. Given the specification of VerifVoter for Belenios, $\mathcal{A}_3$ is also accurately simulated by $\mathcal{B}_2$.

We also construct an adversary C, who plays the game $\mathsf{Exp}^{\mathsf{NM}}$.

- C_1 simulates $\mathcal{A}_1$, similarly to $\mathcal{B}_1$. However, unlike $\mathcal{B}_1$, it stops there, and does not corrupt all identities.
- C_2 first draws at random a bit β'', Similarly to $\mathcal{B}_2$, C_2 then simulates $\mathcal{A}_2$ and uses lists hL, V. For each call to $O^{p,c}_{\mathsf{vote}}(id,v_0,v_1)$, provided $id\in\mathsf{U}_1\backslash\mathsf{CU}_1$, C calls $O_{\mathsf{c}}(id,v_{\beta''})$. C obtains a ballot b, removes all elements of the form $(id,*)$ from V, and appends $(id,v_{\beta''})$ to V, and b to hL. Finally C returns b.
- $\mathcal{A}_2$ eventually returns a board BB. C then draws at random a ballot of BB and returns it.

Note that the lists BB, hL are the same for C in $\mathsf{Exp}^{\mathsf{NM}}$ at the point C_2 returns, and for $\mathcal{B}$ in $\mathsf{Exp}^{\mathsf{ind}}$ at the point the simulated $\mathcal{A}_2$ returns. Similarly V in C is the same as $\mathsf{V}_{\beta''}$ in $\mathcal{B}$. Let us also notice that hL in C is equal to the list L in the game $\mathsf{Exp}^{\mathsf{NM}}$.

We will now prove that if $\mathcal{A}$ wins $\mathsf{Exp}^{\mathsf{priv-careful}}$, then $\mathcal{B}$ wins $\mathsf{Exp}^{\mathsf{ind}}$ provided C does not win $\mathsf{Exp}^{\mathsf{NM}}$.

The adversary C is polynomial, *i.e.* there exists a polynomial $q(\lambda)$ bounding its number of operations. $q(\lambda)$ necessarily also bounds the length of the board BB that C computes.

For any β, assume $\mathsf{BB}\backslash\mathsf{hL}$ contains a ballot b such that there exists honest $(id,cred)\in\mathsf{U}\backslash\mathsf{CU}$, and a vote v, such that $\mathsf{open}(b,\mathsf{sk})=(cred,v)$ (note that U,CU in $\mathsf{Exp}^{\mathsf{NM}}$ are equal to U_1, CU_1 in $\mathcal{B}$). Thus, provided C correctly guesses $\beta''=\beta$, and chooses this b from BB, the condition $b\notin\mathsf{L}\ \wedge\ \exists(id,cred)\in\mathsf{U}\backslash\mathsf{CU}.\ \mathsf{open}(b,\mathsf{sk})=(cred,*)$ in $\mathsf{Exp}_C^{\mathsf{NM}}$ holds. Indeed, since $b\notin\mathsf{hL}$, $b\notin\mathsf{L}$. and thus $\mathsf{Exp}_C^{\mathsf{NM}}=1$.

Therefore, such a ballot b exists with probability at most $2|\mathsf{BB}|\,\mathrm{P}\left[\mathsf{Exp}_C^{\mathsf{NM}}=1\right]$, which is smaller than $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_C^{\mathsf{NM}}=1\right]$ since $|\mathsf{BB}|\leq q(\lambda)$.

For any β, $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda)=1$ if and only if $\mathcal{B}_2$ returns 1 in this game.

Assume $\mathsf{BB}\backslash\mathsf{hL}$ does not contain any ballot associated with a honest credential (*i.e.* a credential in $\mathsf{U}_1\backslash\mathsf{CU}_1$), which, as we have established, holds except with probability at most $2q(\lambda)\,\mathrm{P}\left[\mathsf{Exp}_C^{\mathsf{NM}}=1\right]$.

Note that this assumption notably implies that no ballot in $\mathsf{BB} \backslash \mathsf{hL}$ has the same ciphertext as a ballot generated by a call to O_{c}^{i}, that is, for all $b \in \mathsf{BB} \backslash \mathsf{hL}$, $\mathsf{extract}(b) \notin \mathsf{L}$. Indeed, the ciphertext part of the ballot, for Belenios, is signed with the credential. Hence any $c \in \mathsf{L}$ is (by definition of O_{c}^{i}) signed with a honest credential, which, by the assumption, is not the case of any of the ballots in $\mathsf{BB} \backslash \mathsf{hL}$. Therefore, as stated earlier, O_{d}^{i} accepts to open all (valid) ballots in $\mathsf{BB} \backslash \mathsf{hL}$.

Let us then show that $\mathcal{A}_4$ is accurately simulated by $\mathcal{B}_2$, *i.e.* it is provided by $\mathcal{B}_2$ with the same input as when it is called in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}$, that is the actually tally of the board BB (which $\mathcal{A}_2$ returned) if $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1))$ and if $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$; and $\perp$ otherwise.

It is clear from the definition of $\mathcal{B}$ that when either the equality condition $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1))$ or the voter verification condition $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ do not hold, $\mathcal{A}_4$ is indeed given $\perp$ as an argument.

Let us now study the case where both these conditions are met. Among the ballots in BB, some are also in the list hL of ballots created by the oracle O_{c}^{i}. We will thus see BB as an interleaving of the ballots in $\mathsf{BB} \backslash \mathsf{hL}$ and $\mathsf{BB} \cap \mathsf{hL}$ (keeping the same order within each of these two lists).

By assumption, $\mathsf{BB} \backslash \mathsf{hL}$ contains no ballot for honest credentials, whereas by construction hL (and thus $\mathsf{BB} \cap \mathsf{hL}$) only contains ballots for honest credentials.

Moreover, we assumed the voter verifications succeeded, *i.e.* $\forall id.\ (id, *) \in \mathsf{V}_0, \mathsf{V}_1 \implies id \in \mathsf{H}$ holds. According to the specification of VerifVoter for Belenios, this means that for all id occurring in $\mathsf{V}_0, \mathsf{V}_1$, the last ballot b in L_{id}, *i.e.* the last ballot produced by $O_{\mathsf{vote}}^{p,c}(id, *, *)$, is also the last ballot signed by $cred_{id}$ in BB. ($cred_{id}$ being the credential associated with id in CU_2). Since id (and thus $cred_{id}$) is honest by definition of $\mathsf{V}_0, \mathsf{V}_1$, b is actually the last ballot signed by $cred_{id}$ in $\mathsf{BB} \cap \mathsf{hL}$.

Consider a credential $cred$ such that $\mathsf{BB} \cap \mathsf{hL}$ contains at least one ballot signed by $cred$. This ballot can only have been added to hL by a call to $O_{\mathsf{vote}}^{p,c}$, and therefore, by definition of this oracle, the associated id was at one point added to V_0 and V_1. Since no identity is ever removed from these lists, at the time of tallying, id still occurs in $\mathsf{V}_0, \mathsf{V}_1$.

Hence, the list hL' of the last ballots signed by each credential in $\mathsf{BB} \cap \mathsf{hL}$ is exactly the list of the last ballots produced by $O_{\mathsf{vote}}^{p,c}$ for each id in $\mathsf{V}_0, \mathsf{V}_1$. By construction, this list contains ballots for either the votes in V_0 or those in V_1, depending on β. Since at that point $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$, we have $\rho(\mathsf{V}_0) = \mathsf{Tally}(\mathsf{hL}', \mathsf{sk}, \mathsf{U}_1)$. In addition, the revote policy specified for Belenios is to count only the last ballot signed by each credential. Therefore, $\mathsf{Tally}(\mathsf{BB} \cap \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{hL}', \mathsf{sk}, \mathsf{U}_1) = \rho(\mathsf{V}_0)$.

Besides, the oracle O_{d}^{i} returns the list L of the credentials and votes of each ballot in $\mathsf{BB} \backslash \mathsf{hL}$. Since L' is the list of the last votes associated with each credential in L, we thus have $\mathsf{Tally}(\mathsf{BB} \backslash \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) = \rho(L')$.

Since BB can be separated into the lists $\mathsf{BB} \backslash \mathsf{hL}$ and $\mathsf{BB} \cap \mathsf{hL}$ (whose ballots do not share any credential, by assumption), and since the ballots of each of these two lists occur in the same order in BB, we have

$$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{BB} \cap \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB} \backslash \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) = \rho(\mathsf{V}_0) * \rho(L').$$

Therefore, $\rho(\mathsf{V}_0) * \rho(L')$, which is the result computed by $\mathcal{B}_2$, is indeed $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}_1)$, which concludes the proof that $\mathcal{A}_4$ is accurately simulated by $\mathcal{B}_2$.

Hence, unless $\mathsf{BB} \backslash \mathsf{hL}$ contains a ballot associated with a honest credential, $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda) = 1$ if and only if the accurately simulated $\mathcal{A}_4$ returns 1, *i.e.* if and only if $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},\beta}(\lambda) = 1$.

Thus

$$\left| \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda) = 1 \right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},\beta}(\lambda) \neq 1 \right] \right| \leq 2q(\lambda)\, \mathrm{P}\left[\mathsf{Exp}_{C}^{\mathsf{NM}} = 1 \right]. \tag{8}$$

We thus have:

$$\begin{aligned}
&\left| \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},0}(\lambda) = 1 \right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},1}(\lambda) = 1 \right] \right| \\
= \quad & \left| \left(\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},0}(\lambda) = 1 \right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},0}(\lambda) = 1 \right] \right) + \left(\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},0}(\lambda) = 1 \right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},1}(\lambda) = 1 \right] \right) \right. \\
& \left. + \left(\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},1}(\lambda) = 1 \right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},1}(\lambda) = 1 \right] \right) \right| \\
\leq \quad & \left| \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},0}(\lambda) = 1 \right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},0}(\lambda) = 1 \right] \right| + \left| \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},1}(\lambda) = 1 \right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},0}(\lambda) = 1 \right] \right| \\
& + \left| \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},1}(\lambda) = 1 \right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},1}(\lambda) = 1 \right] \right| \\
\leq \quad & \left| \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},0}(\lambda) = 1 \right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},1}(\lambda) = 1 \right] \right| + 4q(\lambda)\, \mathrm{P}\left[\mathsf{Exp}_{C}^{\mathsf{NM}} = 1 \right]
\end{aligned}$$

Therefore, if $\mathcal{A}$ breaks privacy with careful voters, *i.e.* if $\left| \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},0}(\lambda) = 1 \right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv-careful},1}(\lambda) = 1 \right] \right|$ is not negligible, then $\mathcal{B}$ breaks $\mathsf{Exp}^{\mathsf{ind}}$, or C breaks $\mathsf{Exp}^{\mathsf{NM}}$.

□

C.2.4 *Helios is private for* $\mathsf{Exp}^{\mathsf{priv}}$.

Theorem C.4. *Assuming no adversary wins* $\mathsf{Exp}^{\mathsf{ind}}$ *with non-negligible probability, Helios is private for* $\mathsf{Exp}^{\mathsf{priv}}$.

Proof. Let $\mathcal{A} = \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ be an adversary that wins $\mathsf{Exp}^{\mathsf{priv}}$. We consider an adversary $\mathcal{B} = \mathcal{B}_1, \mathcal{B}_2$ that plays $\mathsf{Exp}^{\mathsf{ind}}$:

- $\mathcal{B}_1^{O_{\mathsf{reg}}, O_{\mathsf{corr}}}(\mathsf{pk})$ first simulates $\mathcal{A}_1^{O_{\mathsf{reg}}, O_{\mathsf{corr}}}(\mathsf{pk})$, *i.e.* $\mathcal{B}$ registers and corrupts the same identities as $\mathcal{A}$, while keeping lists U_1, CU_1 of the identities it declares and corrupts by calling O_{reg} and O_{corr}. $\mathcal{A}$ returns some $state'_1$. $\mathcal{B}_1$ then corrupts each user $\mathcal{A}_1$ has registered, *i.e.*, $\mathcal{B}_1$ calls $O_{\mathsf{corr}}(id)$ for each id $\mathcal{A}_1$ has declared, and stores each id's credential in a list CU_2.

- $\mathcal{B}_2^{O^i_{\mathsf{c}}, O^i_{\mathsf{d}}}(state_1, \mathsf{pk})$ maintains lists $\mathsf{V}_0, \mathsf{V}_1, \mathsf{BB}$, initially empty, which will be used to simulate the lists with the same name in $\mathsf{Exp}^{\mathsf{priv}}$. $\mathcal{B}_2$ will also use a list hL, initially empty.

  $\mathcal{B}_2$ first simulates $\mathcal{A}_2^{O^p_{\mathsf{vote}}, O_{\mathsf{cast}}}(state'_1, \mathsf{pk})$:
  - for each call to $O^p_{\mathsf{vote}}(id, v_0, v_1)$, provided $id \in \mathsf{U}_1 \backslash \mathsf{CU}_1$, $\mathcal{B}$ retrieves id's credential $cred_{id}$ from CU_2, and calls $O^i_{\mathsf{c}}(id, cred_{id}, v_0, v_1)$. $\mathcal{B}$ obtains a ballot b. $\mathcal{B}$ then removes from V_0 and V_1 all elements of the form $(id, *)$, and appends (id, v_0) to V_0, (id, v_1) to V_1, b to hL and to BB. Finally, $\mathcal{B}$ returns b to $\mathcal{A}$.
  - for each call to $O_{\mathsf{cast}}(id, b)$, provided $id \in \mathsf{CU}_1$ and $\mathsf{Valid}(id, b, \mathsf{BB}, \mathsf{pk})$ (which implies, for Helios, that $\mathsf{extract}(b)$ does not already occur in BB, and that $\mathsf{open}_{id}(b) = id$). $\mathcal{B}$ appends b to BB. Finally $\mathcal{B}$ returns b to $\mathcal{A}$.
- $\mathcal{B}_2$ then computes $\rho(\mathsf{V}_0)$, $\rho(\mathsf{V}_1)$, and checks they are equal. If not, $\mathcal{B}_2$ blocks.
- $\mathcal{B}$ then checks whether two ballots in BB have the same ciphertext, *i.e.* if there exist two ballots b, b' in BB such that $\mathsf{extract}(b) = \mathsf{extract}(b')$. If so, following the specification of Helios, $\mathcal{B}$ lets $r = \perp$.
  Otherwise, we write $\mathsf{BB} \backslash \mathsf{hL}$ (resp. $\mathsf{BB} \cap \mathsf{hL}$) the sublist of BB of ballots that do not occur (resp. do occur) in hL. Note that by construction all ballots in hL occur in BB.

- $\mathcal{B}_2$ calls $O^i_{\mathsf{d}}(\mathsf{BB} \backslash \mathsf{hL})$, and obtains a list L of votes. Note that, if that point is reached, no two ballots in BB have the same ciphertext. Hence, no ballot in $\mathsf{BB} \backslash \mathsf{hL}$ has the same ciphertext as a ballot in hL, which is the list of all ballots produced by O^i_{c}. Thus O^i_{d} accepts to open all ballots in $\mathsf{BB} \backslash \mathsf{hL}$. $\mathcal{B}_2$ computes (using BB and L) the list L' of the last vote from each id.
- $\mathcal{B}_2$ computes $r = \rho(\mathsf{V}_0) * \rho(L')$.
- In any case, *i.e.* even if $r = \perp$, $\mathcal{B}_2$ calls $\mathcal{A}_3$ on r, and obtains a bit β'. $\mathcal{B}_2$ returns β'.

Note that the lists BB, V_0, V_1 in $\mathcal{B}$ are equal to the lists of the same name in $\mathsf{Exp}^{\mathsf{priv},\beta}_{\mathcal{A}}$. Then, by construction, $\mathcal{A}_2$ is always accurately simulated by $\mathcal{B}_2$, *i.e.* it is called on the same inputs, shown the same board, and provided with the same oracles as what would happen in $\mathsf{Exp}^{\mathsf{priv},\beta}_{\mathcal{A}}$.

We will now prove that if $\mathcal{A}$ breaks privacy, then $\mathcal{B}$ wins $\mathsf{Exp}^{\mathsf{ind}}$.

For any β, $\mathsf{Exp}^{\mathsf{ind},\beta}_{\mathcal{B}}(\lambda) = 1$ if and only if $\mathcal{B}_2$ returns 1 in this game.

Let us then show that $\mathcal{A}_3$ is accurately simulated by $\mathcal{B}_2$, *i.e.* it is simulated by $\mathcal{B}_2$ only when it is called in $\mathsf{Exp}^{\mathsf{priv},\beta}_{\mathcal{A}}$ (that is, when $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$); and it is provided with the actually tally of the board BB (which $\mathcal{A}_2$ interacted with).

It is clear from the construction of $\mathcal{B}$ that if $\rho(\mathsf{V}_0) \neq \rho(\mathsf{V}_1)$, $\mathcal{A}_3$ is not simulated. Hence, it is simulated by $\mathcal{B}_2$ only when it is called in $\mathsf{Exp}^{\mathsf{priv},\beta}_{\mathcal{A}}$.

When $\mathcal{A}_3$ is called:

- either BB contains duplicate ciphertexts, and $\mathcal{A}_3$ is called on $r = \perp$, which corresponds to what happens in $\mathsf{Exp}^{\mathsf{priv},\beta}_{\mathcal{A}}$;
- or BB does not contain duplicate ciphertexts.

In the first case, $\mathcal{A}_3$ is accurately simulated. Let us study the second case. Let us first partition BB into two boards $\mathsf{BB}_h \uplus \mathsf{BB}_d$, containing respectively the ballots whose id is honest and dishonest (in the same order). By construction, BB_h is equal to hL, and BB_d is equal to $\mathsf{BB} \backslash \mathsf{hL}$.

By assumption, BB contains no duplicate ciphertexts. Hence, the lists BB_h and BB_d do not have any identity or ciphertexts in common, and according to the specification of Helios we have

$$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{BB}_h, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB}_d, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB} \backslash \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1).$$

By construction of $\mathcal{B}_2$, the list of the last ballot associated with each honest id in hL contains ballots for either the votes in V_0 or those in V_1, depending on β. Since at that point $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$, we thus have $\rho(\mathsf{V}_0) = \mathsf{Tally}(\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1)$.

In addition, the oracle O^i_{d} returns the list L of the votes of each ballot in $\mathsf{BB} \backslash \mathsf{hL}$. Since L' is the list of the last vote in L associated in $\mathsf{BB} \backslash \mathsf{hL}$ to each id, we thus have $\mathsf{Tally}(\mathsf{BB} \backslash \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) = \rho(L')$.

Therefore, we indeed have

$$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB} \backslash \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) = \rho(V_0) * \rho(L'),$$

which is the result computed by $\mathcal{B}_2$. This concludes the proof that $\mathcal{A}_3$ is accurately simulated by $\mathcal{B}_2$.

Hence $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda) = 1$ if and only if the accurately simulated $\mathcal{A}_3$ returns 1, *i.e.* if and only if $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}(\lambda) = 1$.
Thus

$$\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda) = 1\right] = \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}(\lambda) \neq 1\right]. \tag{9}$$

We thus have:

$$\left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},1}(\lambda) = 1\right]\right| = \left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},1}(\lambda) = 1\right]\right|.$$

Therefore, if $\mathcal{A}$ breaks privacy, *i.e.* if $\left|\mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},0}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},1}(\lambda) = 1\right]\right|$ is not negligible, then $\mathcal{B}$ breaks $\mathsf{Exp}^{\mathsf{ind}}$.

□

C.2.5 *Simple is private for* $\mathsf{Exp}^{\mathsf{priv}}$.

Theorem C.5. *Assuming no adversary wins* $\mathsf{Exp}^{\mathsf{ind}}$ *with non-negligible probability, Simple is private for* $\mathsf{Exp}^{\mathsf{priv}}$.

Proof. Let $\mathcal{A} = \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ be an adversary that wins $\mathsf{Exp}^{\mathsf{priv}}$. We consider an adversary $\mathcal{B} = \mathcal{B}_1, \mathcal{B}_2$ that plays $\mathsf{Exp}^{\mathsf{ind}}$:

- $\mathcal{B}_1^{O_{\mathsf{reg}}, O_{\mathsf{corr}}}(\mathsf{pk})$ first simulates $\mathcal{A}_1^{O_{\mathsf{reg}}, O_{\mathsf{corr}}}(\mathsf{pk})$, *i.e.* $\mathcal{B}$ registers and corrupts the same identities as $\mathcal{A}$, while keeping lists U_1, CU_1 of the identities it declares and corrupts by calling O_{reg} and O_{corr}. $\mathcal{A}$ returns some $state_1'$. $\mathcal{B}_1$ then corrupts each user $\mathcal{A}_1$ has registered, *i.e.*, $\mathcal{B}_1$ calls $O_{\mathsf{corr}}(id)$ for each id $\mathcal{A}_1$ has declared, and stores each id's credential in a list CU_2.

- $\mathcal{B}_2^{O_{\mathsf{c}}^i, O_{\mathsf{d}}^i}(state_1, \mathsf{pk})$ maintains lists $\mathsf{V}_0, \mathsf{V}_1$, BB, initially empty, which will be used to simulate the lists with the same name in $\mathsf{Exp}^{\mathsf{priv}}$. $\mathcal{B}_2$ will also use a list hL, initially empty.
  $\mathcal{B}_2$ first simulates $\mathcal{A}_2^{O_{\mathsf{vote}}^p, O_{\mathsf{cast}}}(state_1', \mathsf{pk})$:
  - for each call to $O_{\mathsf{vote}}^p(id, v_0, v_1)$, provided $id \in \mathsf{U}_1 \backslash \mathsf{CU}_1$ does not already occur in $\mathsf{V}_0, \mathsf{V}_1$, $\mathcal{B}$ retrieves id's credential $cred_{id}$ from CU_2, and calls $O_{\mathsf{c}}^i(id, cred_{id}, v_0, v_1)$. $\mathcal{B}$ obtains a ballot b. $\mathcal{B}$ then appends (id, v_0) to V_0, (id, v_1) to V_1, b to hL and to BB. Finally $\mathcal{B}$ returns b to $\mathcal{A}$.
  - for each call to $O_{\mathsf{cast}}(id, b)$, provided $id \in \mathsf{CU}_1$ and $\mathsf{Valid}(id, b, \mathsf{BB}, \mathsf{pk})$ (which implies, for Simple, that b does not already occur in BB). $\mathcal{B}$ appends b to BB. Finally $\mathcal{B}$ returns b to $\mathcal{A}$.
- $\mathcal{B}_2$ then computes $\rho(\mathsf{V}_0)$, $\rho(\mathsf{V}_1)$, and checks they are equal. If not, $\mathcal{B}_2$ blocks.
- $\mathcal{B}$ then checks whether two ballots in BB are equal. If so, following the specification of Simple, $\mathcal{B}$ lets $r = \perp$.
  Otherwise, we write $\mathsf{BB} \backslash \mathsf{hL}$ (resp. $\mathsf{BB} \cap \mathsf{hL}$) the sublist of BB of ballots that do not occur (resp. do occur) in hL. Note that by construction all ballots in hL occur in BB in that order.
- $\mathcal{B}_2$ calls $O_{\mathsf{d}}^i(\mathsf{BB} \backslash \mathsf{hL})$, and obtains a list L of votes. Note that, if that point is reached, no two ballots in BB are equal. Hence, no ballot in $\mathsf{BB} \backslash \mathsf{hL}$ is equal to a ballot in hL, which is the list of all ballots produced by O_{c}^i. Thus O_{d}^i accepts to open all ballots in $\mathsf{BB} \backslash \mathsf{hL}$.
- $\mathcal{B}_2$ computes $r = \rho(\mathsf{V}_0) * \rho(L)$.
- In any case, *i.e.* even if $r = \perp$, $\mathcal{B}_2$ calls $\mathcal{A}_3$ on r, and obtains a bit β'. $\mathcal{B}_2$ returns β'.

Note that the lists BB, V_0, V_1 in $\mathcal{B}$ are equal to the lists of the same name in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}$. Then, by construction, $\mathcal{A}_2$ is always accurately simulated by $\mathcal{B}_2$, *i.e.* it is called on the same inputs, shown the same board, and provided with the same oracles as what would happen in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}$.

We will now prove that if $\mathcal{A}$ breaks privacy, then $\mathcal{B}$ wins $\mathsf{Exp}^{\mathsf{ind}}$.

For any β, $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{ind},\beta}(\lambda) = 1$ if and only if $\mathcal{B}_2$ returns 1 in this game.

Let us then show that $\mathcal{A}_3$ is accurately simulated by $\mathcal{B}_2$, *i.e.* it is simulated by $\mathcal{B}_2$ only when it is called in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}$ (that is, when $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$); and it is provided with the actually tally of the board BB (which $\mathcal{A}_2$ interacted with).

It is clear from the construction of $\mathcal{B}$ that if $\rho(\mathsf{V}_0) \neq \rho(\mathsf{V}_1)$, $\mathcal{A}_3$ is not simulated. Hence, it is simulated by $\mathcal{B}_2$ only when it is called in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}$.

When $\mathcal{A}_3$ is called:

- either BB contains duplicate ballots, and $\mathcal{A}_3$ is called on $r = \perp$, which corresponds to what happens in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{priv},\beta}$;
- or BB does not contain duplicate ballots.

In the first case, $\mathcal{A}_3$ is accurately simulated. Let us study the second case. Let us first partition BB into two lists $(\mathsf{BB} \cap \mathsf{hL}) \uplus (\mathsf{BB} \backslash \mathsf{hL})$.

By construction of $\mathcal{B}$, all ballots in hL are present in BB in the same order, and, since BB contains no duplicates, this means $\mathsf{BB} \cap \mathsf{hL} = \mathsf{hL}$.

By assumption, BB contains no duplicate ballots. Hence, the lists hL and BB\hL do not have any ballot in common, and according to the specification of Simple we have

$$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB} \backslash \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1).$$

By construction of $\mathcal{B}_2$, the list hL contains ballots for either the votes in V_0 or those in V_1, depending on β. Since at that point $\rho(\mathsf{V}_0) = \rho(\mathsf{V}_1)$, we thus have $\rho(\mathsf{V}_0) = \mathsf{Tally}(\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1)$.

In addition, the oracle O^i_{d} returns the list L of the votes of each ballot in BB\hL. We thus have $\mathsf{Tally}(\mathsf{BB} \backslash \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) = \rho(L)$.

Therefore, we indeed have

$$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}, \mathsf{U}_1) = \mathsf{Tally}(\mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) * \mathsf{Tally}(\mathsf{BB} \backslash \mathsf{hL}, \mathsf{sk}, \mathsf{U}_1) = \rho(V_0) * \rho(L),$$

which is the result computed by $\mathcal{B}_2$. This concludes the proof that $\mathcal{A}_3$ is accurately simulated by $\mathcal{B}_2$.

Hence $\mathsf{Exp}^{\mathsf{ind},\beta}_{\mathcal{B}}(\lambda) = 1$ if and only if the accurately simulated $\mathcal{A}_3$ returns 1, *i.e.* if and only if $\mathsf{Exp}^{\mathsf{priv},\beta}_{\mathcal{A}}(\lambda) = 1$.

Thus

$$\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},\beta}_{\mathcal{B}}(\lambda) = 1\right] = \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},\beta}_{\mathcal{A}}(\lambda) \neq 1\right]. \tag{10}$$

We thus have:

$$\left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{A}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{A}}(\lambda) = 1\right]\right| = \left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},0}_{\mathcal{B}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{ind},1}_{\mathcal{B}}(\lambda) = 1\right]\right|.$$

Therefore, if $\mathcal{A}$ breaks privacy, *i.e.* if $\left|\mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},0}_{\mathcal{A}}(\lambda) = 1\right] - \mathrm{P}\left[\mathsf{Exp}^{\mathsf{priv},1}_{\mathcal{A}}(\lambda) = 1\right]\right|$ is not negligible, then $\mathcal{B}$ breaks $\mathsf{Exp}^{\mathsf{ind}}$.

□