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 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 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 tellers, who determine the private key, possibly split in shares. The tellers 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 revive 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 $a, b, c$ is denoted $\{a, b, c\}$. 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 $V$ of votes and a set $R$ of possible results, equipped with an associative and commutative operator $*$ (e.g. addition of vectors). A counting function is a function $\rho$ that associates a result $r \in R$ 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 $C = \{a_1, \ldots, a_k\}$. In case voters should select between $k$ candidates or vote blank, we can represent valid votes by vectors representing the selection of candidates

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

where $v^{\text{blank}}$ is the null vector $(0, \ldots, 0)$, representing a blank vote. In a mixnet-based tally, all the individual votes are revealed. Thus $R$ 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 $\rho_{\text{mix}}(V) = V$, 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 $R = \mathbb{N}^k$, the set of vectors of $k$ elements, and $*$ is the addition of vectors. The corresponding counting function is $\rho_{\text{hom}}(V) = \sum_{v \in V} v$. Both $\rho_{\text{mix}}$ and $\rho_{\text{hom}}$ have the partial tally property. The vote $v^{\text{blank}}$ is a neutral vote w.r.t. $\rho_{\text{hom}}$ but not $\rho_{\text{mix}}$.

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 $r \in R$ 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 $\rho_{\text{mix}}$ and $\rho_{\text{hom}}$ 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 $N$ 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 $V = X \uplus \mathcal{AX}$ where $X$ 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.

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

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

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

Given a signature $\mathcal{F}$, a set of names $N$, a set of variables $V$, the set of terms $T(\mathcal{F}, V, N)$ is the set inductively defined by applying functions to variables in $V$ and names in $N$. The set of names resp. variables occurring in $t$ is denoted $\text{names}(t)$ (resp. $\text{vars}(t)$). A term is ground if it does not contain any variable. The set of terms $T(\mathcal{F}, \mathcal{AX}, \mathcal{FN})$ 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 $\sigma = \{M_1/x_1, \ldots, M_k/x_k\}$ maps variables $x_1, \ldots, x_k \in V$ to messages $M_1, \ldots, M_k$. Its domain is denoted $\text{dom}(\sigma) = \{x_1, \ldots, x_k\}$. The application of $\sigma$ to a term $t$ is denoted $t\sigma$ and is defined as usual. A substitution $\sigma$ is ground if its messages $M_1, \ldots, M_k$ 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 $\lambda \equiv \eta$ where $\lambda, \eta \in T(\mathcal{F}, V, 0)$ are messages without names. Equality modulo $E$, denoted by $\equiv_E$, is defined as the smallest equivalence relation on terms that is closed under context and substitution. We denote disequalities modulo $E$ by $M \not\equiv_E N$.

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:
\[ P, Q \mapsto \begin{align*}
\emptyset & \quad \text{for } n \in \mathbb{N} \text{ (n bound in } P) \\
\nu n.P & \\
\text{out}(c, M).P & \\
\text{in}(c, x).P & \quad \text{for } x \in X \text{ (x bound in } P) \\
n & \mapsto (M_1, \ldots, M_n).P & \quad \text{for event } \text{ev} \text{ of arity } n \\
| & \\
\text{let } x = M \text{ in } P & \quad \text{for } x \in X \text{ (x bound in } P) \\
| & \\
\text{if } M = N \text{ then } P \text{ else } Q & \\
| & !P \\
\end{align*} \]
where \( M, N, M_1, \ldots, M_n \) are messages and \( c \in C \) is a channel.

**Figure 1: Syntax for processes.**

characterises + as an associative and commutative operator.

\[
\begin{align*}
\text{dec}(\text{enc}(x, y), y) & = x \\
\text{adec(} & \text{aenc}(x, pk(y)), y) = x \\
\pi_1((x, y)) & = x \\
\pi_2((x, y)) & = y \\
x + (y + z) & = (x + y) + z \\
x + y & = y + x
\end{align*}
\]

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 protocols. Protocols are modelled through a process algebra, whose syntax is displayed in Figure 1.

As usual, we identify processes up to \( \alpha \)-renaming, to avoid capture of bound names and variables.

A configuration of the system is a triple \( (E; P; \phi) \) where:

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

The semantics of processes is given through a transition relation \( \xrightarrow{\alpha} \) provided in Figure 2, where \( \alpha \) is the action associated to the transition. \( \tau \) 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 \( E \), and behaves like \( P \). Process \( \text{out}(c, M).P \) emits \( M \) on the channel \( c \) and behaves like \( P \). Process \( \text{in}(c, x).P \) inputs a term computed by the attacker (that is a term built from \( \phi \) using an attacker term) on channel \( c \) and then behaves like \( P \). Process \( \text{event}(M_1, \ldots, M_n).P \) triggers the event \( \text{ev} \) \( (M_1, \ldots, M_n) \) and then behaves like \( P \). Process \( P \mid Q \) corresponds to the parallel composition of \( P \) and \( Q \). Process \( \text{let } x = M \text{ in } P \) behaves like \( P \) in which \( x \) is replaced with \( M \). Process \( \text{if } M = N \text{ then } P \text{ else } Q \) behaves like \( P \) if \( M \) and \( N \) are equal modulo \( E \), and behaves like \( Q \) otherwise. The replicated process \( P \mid Q \) behaves as an unbounded number of copies of \( P \).

We denote by \( w \xrightarrow{\tau} \), 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 as follows.

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 \( \text{trace}(P) \) is defined as follows.

\[
\text{trace}(P) = \{(w, \nu E.\phi)|\langle\emptyset; \{P\}; \emptyset\rangle \xrightarrow{w} \langle E; P; \phi\rangle\}
\]

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

\[
\text{blocking}(t, P) \overset{\text{def}}{=} \forall \phi. (t, \phi) \not\in \text{trace}(P).
\]

**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.

\[
P_{\text{Helios}}(v_a, v_b) =
\nu \, k_{ax}, k_{bx}, k_{cx}, k_e.
\text{out}(c, k_e) . \text{out}(c, \text{pk}(k_e)) \mid
\text{Voter}(A, v_a, c_a, c'_a, k_{ax}, k_e) \mid
\text{Voter}(B, v_b, c_b, c'_b, k_{bx}, k_e) \mid
\text{Tally}_{\text{Helios}}(c_a, c_b, c_e, c'_a, c'_b, k_{ax}, k_{bx}, k_{cx}, k_e)
\]

where \( \text{Voter}(a, v, c, c', k, k_e) \) 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 \( \text{Tally}_{\text{Helios}} \) represents the voting server.

\( \text{Voter}(a, v, c, c', k, k_e) \) 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.

\[
\text{Voter}(a, v, c, c', k, k_e) =
\nu \, r. \text{out}(c, \text{enc}(\text{aenc}((v, r), \text{pk}(k_e)), k_e)). \text{Voted}(a, v).
\text{out}(c', \text{aenc}((v, r), \text{pk}(k_e)))
\]

The voting server receives ballots from voters \( A, B, \) and \( C \) and then outputs the decrypted ballots, after some mixing, modelled
We write $P$ while private names are stored in $\tau$ through the + operator.

$$\text{Tally}_{\text{Helios}}(c_0, c_1, c_2, c_3, k_{x_0}, k_{x_1}, k_{x_2}, k_e) = in(c_0, x_1).in(c_1, x_2).in(c_2, x_3).$$

$$\text{let } y_1 = \text{dec}(x_1, k_{x_0}) \text{ in }$$
$$\text{let } y_2 = \text{dec}(x_2, k_{x_1}) \text{ in }$$
$$\text{let } y_3 = \text{dec}(x_3, k_{x_2}) \text{ in }$$

$$\text{if } x_1 \neq x_2 \land x_1 \neq x_3 \land x_2 \neq x_3 \text{ then }$$
$$\text{out}(c_3, \pi_1(\text{dec}(y_1, k_e)) + \pi_1(\text{dec}(y_2, k_e)) + \pi_1(\text{dec}(y_3, k_e)))$$

where we omit the null else-branches. $\land$ is syntactic sugar for a success of tests and if $M \neq N$ then $P$ is syntactic sugar for if $M = N$ then $\phi$ else $P$.

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

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 $\text{new } E.\phi$ and $\text{new } E'.\phi'$ are statically equivalent if and only if they have the same domain, and for all attacker terms $R, S$ with variables in $\text{dom}(\phi) = \text{dom}(\phi')$, we have

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

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 $P \vdash T Q$ if for all $(s, \psi) \in \text{trace}(P)$, there exists $(s', \psi') \in \text{trace}(Q)$ such that $s =_{E} s'$ and $\psi$ and $\psi'$ are statically equivalent. We say that $P$ and $Q$ are trace equivalent, and we write $P \approx_{T} Q$, if $P \vdash T Q$ and $Q \vdash T P$.

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:

$$P_{\text{Helios}}(0, 1) \approx_{T} P_{\text{Helios}}(1, 0)$$

3.2 Voting protocols

We consider two disjoint, infinite subsets of $C$: a set $A$ of agent names or identities, and a set $V$ 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

$$P = v_{\text{cred}1}.v_{\text{cred}4}.\ldots.v_{\text{cred}p}.(\text{Voter}(a_{1}, v_{a_{1}}, e_{1}, \text{cred}, \text{cred}_{1}) | \ldots | \text{Voter}(a_{n}, v_{a_{n}}, e_{n}, \text{cred}, \text{cred}_{n}) | \text{Tally}_{P}(e_{1}, \text{cred}_{1}, \ldots, \text{cred}_{p}) | \text{Others}_{P}(e_{1}, \text{cred}_{1}, \ldots, \text{cred}_{p}))$$

where $a_{i} \in A$, $v_{a_{i}} \in V$, $e_{1}$, $e_{2}$, $e_{3}$ are (distinct) channels, $\text{cred}$ and $\text{cred}_{i}$ are (distinct) names.

A voting process may be instantiated by various voters and vote selections. Given $A = \{a_{1}, \ldots, a_{n}\} \subseteq A$ a finite set of voters, and $\alpha : A \rightarrow V$ that associates a vote to each voter, we define $P_{\alpha}$ by replacing $a_{i}$ by $b_{i}$ and $v_{i}$ by $a(b_{i})$ in $P$. 

Figure 2: Semantics
Moreover, P must satisfy the following properties:

- Process Voter(a, υ, τ, cred) models an honest voter a willing to vote for υ, using the channels τ, credentials cred (e.g., a signing key) and election credentials cred. It is assumed to contain an event Voted(a, v) that models that a has voted for v. This event is typically placed at the end of process Voter(a, υ, τ, cred). This event cannot appear in process Tally_p nor Others_p.

- Process Tally_p(τ, cred, cred₁, . . . , cred_p) models the tally. It is parametrised by the total number of voters p (honest and dishonest), with p ≥ 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.

∀a. ∀(tr, ϕ) ∈ trace(P). out(c_r, r) ∈ tr ⇒ ∃V. ϕ(r) ∈ R(ρ(V))

Tally_p may of course contain input/output actions on other channels.

- Process Others_p(τ, cred, cred₁, . . . , 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: out(c, kec).out(c, pk(kec)).

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

Voters(tr) = {a ∈ A | ∃v ∈ V. Voted(a, v) ∈ 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 result(t, ϕ, V) holds if the election result in (t, ϕ) corresponds to V.

result(t, ϕ, V) def = ∃x. t′ = t′.out(c_r, x) ∧ ϕ(x) ∈ R(ρ(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, ϕ) ∈ trace(P), of the form t′.out(c_r, x), there exists V_c such that the result in t corresponds to V_c ⊇ V_c, that is:

V_a = {v | ∃a. Voted(a, v) ∈ 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 substitution α from voters to votes, for any two voters a, b ∈ A \ dom(α) and any two votes v_1, v_2 ∈ V, we have

P_{α∪{a→v_1,b→v_2}} ≈ P_{α∪{a→v_2,b→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 = t′.out(c_r, x), ϕ ∈ trace(P_α), and (t′.out(c_r, x), ϕ′) ∈ trace(P_α), then

ϕ(x) ∈ R(ρ(V)) ⇒ ϕ′(x) ∈ R(ρ(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 process P is voting friendly if for all voter a ∈ A, there exists t″ (the honest voting trace) such that for all α satisfying α ∉ dom(α),

- for all (t, ϕ) ∈ trace(P_α), such that t = t″.out(c_r, x) for some t′, x, for all α, there exists tr, ψ such that tr = t″, Voted(a, v) ∈ tr, (t′.tr.out(c_r, x), ψ) ∈ trace(P_{α∪{a→v}}), and ∀V. (ϕ(x) ∈ R(ρ(V))) ⇒ ψ(x) ∈ R(ρ(V∪{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 blocking(t′.out(c_r, x), P_α), for all α, tr, ψ such that tr = t″, we have blocking(t′.tr.out(c_r, x), P_{α∪{a→v}}). Intuitively, the fact that a voter does not suddenly unlock the tally.

In practice, most voting systems are voting friendly since voters vote independently. In particular, process 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

\[
[t \equiv_t t' \land (t, \phi) \in \text{trace}(P_{\Delta U}((a \rightarrow v_1) \cup (a \rightarrow v_2))) \land (t', \phi') \in \text{trace}(P_{\Delta U}((a \rightarrow v_1) \cup (a \rightarrow v_2))) \land \text{result}(t, \phi, V) \land \text{result}(t', \phi', V') \Rightarrow \rho(V' \cup \{v_1\}) = \rho(V \cup \{v_2\})].
\]

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 $V_0$ denote the multiset of honest votes, and $V_1$ the multiset containing the same number of neutral votes. Thanks to Lemma 3.12, we can show that $r \equiv \rho(V_0) = r' \equiv \rho(V_1)$. Since $V_0$ only contains neutral votes, we have $r \equiv r' \equiv \rho(V_0)$. 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 $(id, \ast) \in L$ 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

- (Setup, Credential, Vote, VerifVoter, Tally, Valid)

- Setup($\lambda$), given a security parameter $\lambda$, returns a pair of election keys $(pk, sk)$.
- Credential($id$, $id$) 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 $U$.
- Vote($id$, $cred$, $pk$, $v$) constructs a ballot containing the vote $v$ for voter $id$ with credential $cred$, using the election public key $pk$.
- VerifVoter($id$, $cred$, $L$, $BB$) 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.
- Tally($BB$, $sk$, $U$) 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 text $ValidTally(BB, sk, U)$ that typically checks that the ballots of $BB$ are valid. Tally may return $\perp$ if the tally procedure fails (invalid or decryption failure for example). If Tally($BB$, $sk$, $U$) $\neq \perp$ then it must correspond to a valid result, that is, there exists $V$ such that $Tally(BB, sk, U) = \rho(V)$.
- Valid($id$, $b$, $BB$, $pk$) 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 Valid($id$, $b$, $BB$, $pk$) succeeds.

We will always assume a correct voting scheme, that is, tallying honestly generated ballots yields the expected result. Formally, for all distinct identities $U = id_1, \ldots, id_n$, and credentials $cred_1, \ldots, cred_n$, for all votes $v_1, \ldots, v_n$, for all election keys $(pk, sk)$, if $BB = [Vote(id_1, cred_1, pk, v_1), \ldots, Vote(id_n, cred_n, pk, v_n)]$, then

\[
Tally(BB, sk, U) = \rho(\{v_1, \ldots, v_n\})
\]

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 two vote counts (typically when revote is possible).
2. the first vote counts (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 function $open_{id}$ which, given a ballot $b$, retrieves the associated identity. Formally, for any $id$, $cred$, $pk$, $v$,

\[
open_{id}(Vote(id, cred, pk, v)) = id
\]

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 $id$, $cred$, $sk$, $pk$, $v$,

\[
open_{cred}(Vote(id, cred, pk, v), sk, U) = cred
\]
\[ O_{\text{reg}}(id) \]
\[ \text{if } (id, *) \in U \text{ then stop} \]
\[ \text{else} \]
\[ \text{cred}_id \leftarrow \text{Credential}(1^\lambda, id) \]
\[ U \leftarrow U \|(id, \text{cred}_id) \]
\[ O_{\text{corr}}(id) \]
\[ \text{if } (id, *) \notin U \lor (id, *) \in CU \text{ then stop} \]
\[ \text{else} \]
\[ \text{CU} \leftarrow \text{CU} \|(id, \text{cred}_id) \]
\[ \text{return cred}_id \]
\[ \text{where } (id, \text{cred}_id) \notin U \]

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 re-vote 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 definitions to a dishonest tally, and whether the implication still holds.

Verifiability is defined through the game Exp\textsuperscript{verif}(λ) displayed on Figure 4. In a first step, the adversary may use oracles \( O_{\text{reg}}(id) \) and \( O_{\text{corr}}(id) \) (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 \( O_{\text{vote}}(id, v) \). 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 \( O_{\text{cast}}(id, b) \). 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 \( \mathcal{A} \),

\[ P[\text{Exp}_{\mathcal{A}}^{\text{verif}}(\lambda) = 1] \]

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 have 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 \( O_{\text{reg}}(id) \) and \( O_{\text{corr}}(id) \) 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 \( O_{\text{vote}}^{\text{prv}}(id, v_0, v_1) \). Voter \( id \) will vote \( v_\beta \) depending on the bit \( \beta \). The adversary may also cast an arbitrary ballot \( b \) in the name of a dishonest voter \( id \) through oracle \( O_{\text{cast}}(id, b) \). 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 \( \beta \). Formally, privacy is defined through the game \( \text{Exp}_{\mathcal{A}}^{\text{prv}}(\lambda) \) displayed on Figure 5.

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

\[ P[\text{Exp}_{\mathcal{A}}^{\text{prv}, 0}(\lambda) = 1] - P[\text{Exp}_{\mathcal{A}}^{\text{prv}, 1}(\lambda) = 1] \]

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 both boards BB\(_1\), BB\(_2\) are independant then Tally(BB\(_1\) ⊕ BB\(_2\)) = Tally(BB\(_1\)) + Tally(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
Tally defined through the game tally: if more voters does not change the tally: if $\forall b, b' \in \mathbb{B}; \text{open}_{id}(b) \neq \text{open}_{id}(b')$, then their tally can be computed separately:

$\text{Tally}(\mathbb{B}_1 \uplus \mathbb{B}_2, sk, U) = \text{Tally}(\mathbb{B}_1, sk, U) \uplus \text{Tally}(\mathbb{B}_2, sk, U)$. (*)

We also assume that the tally only counts ballots cast with registered ids, i.e. $\forall b, b' \in \mathbb{B}_b; \text{open}_{id}(b) = \text{open}_{id}(b')$ where $\mathbb{B}_b = \{ b \in \mathbb{B} \mid \text{open}_{id}(b) \neq \text{open}_{id}(b') \}$. (\text{Property} (*)).

Similarly, a cred-based voting scheme has the piecewise tally property if for any two boards $\mathbb{B}_1$ and $\mathbb{B}_2$ that contain ballots associated to different credentials, that is $\forall b, b' \in \mathbb{B}_b; \text{open}_{id}(b, sk, U) \neq \text{open}_{id}(b', sk, U)$ 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 $\forall b, b' \in \mathbb{B}_b, (\text{open}_{id}(b), *) \in U'$ and $\text{Tally}(\mathbb{B}_b, sk, U) = \text{Tally}(\mathbb{B}_b, sk, U \cup U')$.

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 $\text{Exp}^\text{ValidTally}(\lambda)$ displayed in Figure 6. A similar assumption was introduced in [7]. For example, Helios is strongly correct.

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

---

**Figure 4: Verifiability**

\[
\begin{align*}
\text{Exp}^\text{priv}_{\mathcal{A}}(\lambda) & \quad \text{O}^\text{vote}(id, v) \\
(pk, sk) & \leftarrow \text{Setup}(\lambda^\lambda) \\
U, \text{CU} & \leftarrow (U) \\
\text{state} & \leftarrow \mathcal{A}^\text{priv}_{\mathcal{A}}(pk) \\
\mathbb{B}, \mathbb{V}_0, \mathbb{V}_1 & \leftarrow (U) \\
\text{state}_0 & \leftarrow \mathcal{A}^\text{valid}_{\mathcal{A}}(\text{state}, pk) \\
\beta' & \leftarrow \mathcal{A}_v(\text{state}_0, pk, r) \\
& \text{return } \beta' \text{ if } \mathbb{V}_0 = \mathbb{V}_1 \text{ then} \\
\mathbb{B} & \leftarrow \mathbb{B} \parallel \mathbb{V}_0 \parallel (id, v_0) \\
\mathbb{V}_1 & \leftarrow (id, v_1) \\
& \text{return } \beta' \text{ if } \mathbb{V}_0 = \mathbb{V}_1 \text{ then} \\
\mathbb{B} & \leftarrow \mathbb{B} \parallel \mathbb{V}_0 \parallel (id, v_0) \\
\mathbb{V}_1 & \leftarrow (id, v_1) \\
& \text{return } b \text{ where } (id, cre_{id}) \in U \\
& \text{and } V'_0 \text{ (resp. } V'_1) \text{ is obtained from } V_0 \text{ (resp. } V_1) \\
& \text{by removing all instances of } (id, *)
\end{align*}
\]

**Figure 5: Privacy**

\[
\begin{align*}
\text{Exp}^\text{priv}_{\mathcal{A}}(\lambda) & \quad \text{O}^\text{vote}(id, v_0, v_1) \\
(pk, sk) & \leftarrow \text{Setup}(\lambda^\lambda) \\
U, \text{CU} & \leftarrow (U) \\
\text{state}_0 & \leftarrow \mathcal{A}^\text{valid}_{\mathcal{A}}(pk) \\
\mathbb{B}, \mathbb{V}_0, \mathbb{V}_1 & \leftarrow (U) \\
\text{state}_1 & \leftarrow \mathcal{A}^\text{valid}_{\mathcal{A}}(\text{state}, pk) \\
\beta' & \leftarrow \mathcal{A}_v(\text{state}_1, pk, r) \\
& \text{return } b \text{ where } (id, cre_{id}) \in U \\
& \text{and } V'_0 \text{ (resp. } V'_1) \text{ is obtained from } V_0 \text{ (resp. } V_1) \\
& \text{by removing all instances of } (id, *)
\end{align*}
\]

**Figure 6: ValidTally game**

\[
\begin{align*}
\text{Exp}^\text{valid}_{\mathcal{A}}(\lambda) & \quad \text{O}^\text{vote}(id, v) \\
(pk, sk) & \leftarrow \text{Setup}(\lambda^\lambda) \\
U, \text{CU} & \leftarrow (U) \\
\text{state} & \leftarrow \mathcal{A}^\text{valid}_{\mathcal{A}}(pk) \\
\mathbb{B}, id, v & \leftarrow (\text{state}, pk) \\
\beta' & \leftarrow \mathcal{A}_v(id, cre_{id}, pk, v) \\
& \text{return } \text{Vote}(id, cre_{id}, pk, v) \\
& \text{if } (id, *) \in \text{CU} \text{ then} \\
& \text{return } V_o \text{ (resp. } V_1) \text{ is obtained from } V_0 \text{ (resp. } V_1) \\
& \text{by removing all instances of } (id, *)
\end{align*}
\]
where, in the result of the election, they are no longer compensated by the new votes. This allows the attacker to break privacy.

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 not be compensated 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 \( \text{Exp}_{\text{verif}}^{\mathcal{A}} \) with non negligible probability. We construct an attacker \( \mathcal{B} \) that breaks privacy, i.e., wins \( \text{Exp}_{\text{priv},\beta}^{\mathcal{B}} \). \( \mathcal{B} \) starts by registering, and corrupting, the same voters as \( \mathcal{A} \), using oracles \( \mathcal{O}_{\text{reg}} \) and \( \mathcal{O}_{\text{corr}} \). Let \( id_1, \ldots, id_n \) be this first set of voters. \( \mathcal{B} \) then registers another set of \( n \) voters

\[
\text{Exp}_{\text{priv},\beta}^{\mathcal{A}}(\lambda) = \begin{cases} 1 & \text{if } (id, v) \in U \text{ or } \mathcal{A} \text{ wins new game} \\ 0 & \text{otherwise} \end{cases}
\]

\[
\text{Exp}_{\text{verif}}^{\mathcal{A}}(\lambda) = \begin{cases} 1 & \text{if } (id, v) \in U \text{ or } \mathcal{A} \text{ wins new game} \\ 0 & \text{otherwise} \end{cases}
\]

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, \( r \neq \rho(v_1, \ldots, v_n) \) if \( r_1 \). 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 \( \beta \), and wins \( \text{Exp}_{\text{priv}}^{\mathcal{B}} \).

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 as \( \text{Exp}_{\text{priv},\beta}^{\mathcal{A}}(\lambda) \) except that the adversary arbitrarily controls the ballot box. Unfortunately, an adversary can always win this new game. Indeed, he may simply query

\[
\text{Exp}_{\text{vote}}^{\mathcal{A}}(\lambda) = \begin{cases} 1 & \text{if } (id, v_0) \in U \text{ or } \mathcal{A} \text{ wins new game} \\ 0 & \text{otherwise} \end{cases}
\]

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 \( \text{Exp}_{\text{vote}}^{\mathcal{B}}(\lambda, v_0, v_1) \) as before. Note that there is no need for the \( \mathcal{O}_{\text{cor}} \) oracle since the adversary may add directly his own
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 box 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.

**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 \(\beta\)) through the oracle \(\mathcal{O}_{\text{vote}}^{\text{BS}}(id, v_0, v_1)\). 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 $O^{bs}_{vote}(id_1, 1, 0)$, followed by $O^{bs}_{vote}(id_1, 0, 1)$, obtaining ballots $b_1, b'_1$, and return the board $BB = \{b_1, b'_1\}$. The equality condition on the number of ballots in $BB$ produced by $O^{bs}_{vote}$ holds, since for $v = 0, 1$: 

$$|\{b \in BB|3v'. (b, v, v') \in L\}| = |\{b \in BB|3v'. (b, v, v') \in L\}| = 1$$

where $L = \{(b_1, 1, 0), (b'_1, 0, 1)\}$. 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 $\text{Exp}^{BS}$.

### 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 $\mathcal{E} = (\text{gen}, \text{enc}, \text{dec})$ 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:

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

$$\text{VerifVote}(id, cred, L_{id}, BB) =$$

the last element in $L_{id}$ is the last ballot registered for $id$ in $BB$.

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

$$\text{Valid}(id, (id', v), BB, pk) = (id = id') \land \leftarrow v \text{ does not occur in } BB$$

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:

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

where $\text{signElGamal}(\cdot, \cdot, \cdot) = \text{comb}(\cdot, \cdot)$ 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

$$\text{Vote}(id, cred, pk, v) = (\text{enc}(pk, cred), \text{enc}(pk, v), π).$$

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.

$$\text{VerifVote}(id, cred, L_{id}, BB) = b \in BB$$

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 vote, 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 $\text{VerifVote}(id, cred, L_{id}, BB) = \text{true}$. Voters have to trust the voting server (or other internal components) on this aspect.
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 $O_{\text{vote}}^\rho(id_1, 0, 1)$ twice and $O_{\text{vote}}^\rho(id_2, 1, 0)$ 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 modeling 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

National recommendation on e-voting.


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 \( c \), i.e.

   \[
   V.a. \forall (tr, \phi) \in \text{trace}(P_a). \text{out}(c_r, r) \in tr \Rightarrow \exists \mathcal{V}. \phi(r) \in R(\rho(V)).
   \]

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: \( \forall \mathcal{V}, V'. \rho(V \cup V') = \rho(V) * \rho(V') \). * is an associative, commutative operation.

4. The voting processes must be election determinate, i.e. satisfy

   \[
   V.a, t, t', \phi, \phi', x, V .
   \]

   \[
   (t = t' \land (t. \text{out}(c_r, x), \phi) \in \text{trace}(P_a) \land (t'. \text{out}(c_r, x), \phi') \in \text{trace}(P_a) \land \phi(x) \in R(\rho(V)) \Rightarrow \phi'(x) \in R(\rho(V))
   \]

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 \( a \) satisfying \( a \notin \text{dom}(\alpha) \),

   - for all \( (t, \phi) \in \text{trace}(P_a) \), such that \( t = t''. \text{out}(c_r, x) \) for some \( t', x \), for all \( v \), there exists \( tr, \psi \) such that \( tr = t'' \), \( \text{Voted}(a, v) \in tr \), such that \( tr = t''. \text{out}(c_r, x), \psi \) \in \text{trace}(P_a \cup (a \rightarrow r \rightarrow c_r)), \) and \( \mathcal{V}. \phi(x) \in R(\rho(V)) \Rightarrow \psi(x) \in R(\rho(V \cup [v])) \).

   - and for all \( t', x \) such that blocking\((t'. \text{out}(c_r, x), P_a)\), for all \( v \), \( tr, \psi \) such that \( tr = t'' \), blocking\((t'. \text{tr.out}(c_r, x), P_a \cup (a \rightarrow r \rightarrow c_r))\).

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

6. There exists a neutral vote \( v_{\text{neutral}} \in \mathcal{V} \), such that \( \rho([v_{\text{neutral}}]) \) is neutral for *.

7. There exists a special vote \( v_{\text{special}} \in \mathcal{V} \), which is counted separately in the result, as briefly sketched in the core of the paper. Formally, \( v_{\text{special}} \) must enjoy the following properties.

   - the result associated with a multiset determines the number of instances of \( v_{\text{special}} \) in it

   \[
   \forall \mathcal{V}, V'. \rho(V) = \rho(V') \Rightarrow V(v_{\text{special}}) = V'(v_{\text{special}}).
   \]

   - the count of \( v_{\text{special}} \) can be simplified

   \[
   \forall \mathcal{V}, V', k. \rho(V \cup k \cdot v_{\text{special}}) = \rho(V' \cup k \cdot v_{\text{special}}) \Rightarrow \rho(V) = \rho(V').
   \]

For example, for \( \rho_{\text{hom}} \), all the votes are special, therefore this property always holds. For \( \rho_{\text{mix}} \), 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:

\[
V_1(t, \phi) \overset{\text{def}}{=} \forall t', x. (t = t'. \text{out}(c_r, x)) \Rightarrow \exists \mathcal{V}. \phi(x) \in R(\rho([v \mid \exists a. \text{Voted}(a, v) \in t \cup \mathcal{V}]))
\]

\[
V \overset{\text{def}}{=} \forall a. V(t, \phi) \in \text{trace}(P_a). V_1(t, \phi).
\]

- Privacy:

\[
P = \forall a, b \in \mathcal{A} \backslash \text{dom}(\alpha). \forall V_1, V_2. P_{a \cup \{a \rightarrow v_1, b \rightarrow v_2\}} \approx_{t} P_{a \cup \{a \rightarrow v_2, b \rightarrow v_1\}}
\]

- First pivot property:

\[
F \overset{\text{def}}{=} \forall a. \forall a \in \mathcal{A} \backslash \text{dom}(\alpha). \forall V_1, V_2 \in \mathcal{V}. \forall t, t', \phi, \phi', V, V'.
\]

\[
[ t = t' \land (t, \phi) \in \text{trace}(P_{a \cup \{a \rightarrow v_1\}}) \land (t', \phi') \in \text{trace}(P_{a \cup \{a \rightarrow v_2\}}) \land \text{result}(t, \phi, V) \land \text{result}(t', \phi', V') ] \Rightarrow \rho(V' \cup [v_1]) = \rho(V \cup [v_2])
\]
The idea of the proof is to compare $\rho(t, \phi, V) \Rightarrow |V_{\text{change}}| = |V_{\text{wanted}}| \Rightarrow \exists V_{\text{c}}. \rho(V) \ast \rho(V_{\text{change}}) = \rho(V_{\text{wanted}}) \ast \rho(V_c)$. 

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 affection of votes $a$, an agent $a \not\in \text{dom}(a)$, votes $v_1, v_2 \in V$, traces $(t, \phi) \in \text{trace}(P_{a \cup \{a \rightarrow v_1\}})$ and $(t', \phi') \in \text{trace}(P_{a \cup \{a \rightarrow v_2\}})$, such that $t = t'$ and two multisets $V$, $V'$, such that result($t, \phi, V$), result($t', \phi', V'$), and $\rho(V' \uplus \{v_1\}) \neq \rho(V \uplus \{v_2\})$.

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

Note that we necessarily have $v_1 \neq v_2$: indeed, if $v_1 = v_2$ then $(t, \phi)$ and $(t', \phi')$ 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 $v_1 \neq v_2$, we have $\rho(V') \ast \rho(\{v_2\}) = \rho(V') \ast \rho(\{v_1\})$, which is contradictory. Hence $v_1 \neq v_2$.

The attack on privacy consists in the fact that, since changing a voter’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 $b \notin \text{dom}(\alpha) \cup \{a\}$. By assumption 5, there exist sequences of actions $t_b, t_b'$, and frames $\psi, \psi'$, such that

$$(t_1, t_b.out(c_r, x), \psi) \in \text{trace}(P_{a \cup \{a \rightarrow v_1, b \rightarrow v_2\}}), (t_1', t_b'.out(c_r, x), \psi') \in \text{trace}(P_{a \cup \{a \rightarrow v_2, b \rightarrow v_1\}}), Voted(b, v_2) \in t_b, Voted(b, v_1) \in t_b', t_b = t, t_b' = t', \psi(x) \in R(\rho(V \uplus \{v_2\})), \psi'(x) \in R(\rho(V' \uplus \{v_1\})).$$

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 $P_{a \cup \{a \rightarrow v_1, b \rightarrow v_2\}}$ and $P_{a \cup \{a \rightarrow v_2, b \rightarrow v_1\}}$, which yield different results for the election. Using assumption 4, this lets us prove that these two processes are not $\equiv_t$-equivalent.

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

Thus, $P_{a \cup \{a \rightarrow v_1, b \rightarrow v_2\}} \not\equiv t P_{a \cup \{a \rightarrow v_2, b \rightarrow v_1\}}$. This violates $P$, which concludes the proof.

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

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

Proof. Assume that both $P$ and $F$ hold. Let $a$ be an affection of votes, let $(t, \phi) \in \text{trace}(P_a)$, let $V$ be such that result($t, \phi, V$), i.e. there exist $t', x$ such that $t = t'.out(c_r, x)$ and $\phi(x) \in R(\rho(V))$. Let $V_{\text{wanted}} \subseteq \{v | \exists a. Voted(a, v) \in t\}$, and $V_{\text{change}}$ such that $|V_{\text{change}}| = |V_{\text{wanted}}|$. To prove $FF$, we need to show that in this trace augmented with $V_{\text{change}}$ contains at least the subset $V_{\text{wanted}}$ of the intended votes of the honest voters. That is to say, we must show that there exists $V_{\text{c}}$ such that $\rho(V) \ast \rho(V_{\text{change}}) = \rho(V_{\text{wanted}}) \ast \rho(V_c)$.

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

Let us denote the Voted events appearing in $t$ by

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

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

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

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

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

16
For $i \in [0, m]$, let $a_i$ denote the affection of votes obtained from $a$ by turning the first $i$ votes from $V_{\text{wanted}}$ to $V_{\text{change}}$, i.e.

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

Let $\beta \overset{\text{def}}{=} a_m$. Note that $a_0 = a$, and that all the $a_i$ have the same domain.

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

$$\forall i \in [0, k_1] . \exists t_i . t_i = t \land \neg \text{blocking}(t_i, P_{a_i}).$$

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

$$\forall i . \ t_i = t \Rightarrow \text{blocking}(t_i, P_{a_i}).$$ (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 $t_{i-1} = t$, and $\neg \text{blocking}(t_{i-1}, P_{a_{i-1}})$. Since $t = t'_1 . \text{out}(c_r, x)$, we also have $t_{i-1} = t'_{i-1} . \text{out}(c_r, x)$ for some $t'_{i-1}$ such that $t'_{i-1} = t'$.

Then, for all $t'_i = t'_{i-1} = t'$, by (1), blocking($t'_i . \text{out}(c_r, x)$, $P_{a_i}$) 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 $a_i$. Let us construct an attack on privacy, which constitutes a contradiction. Indeed, by assumption 5, we may add a voter $b \notin \text{dom}(a)$, who votes for $v'_i$ at step $i - 1$, and for $v_i$ at step $i$, and there exists $t_r$ such that

- there exists $t''_r = t_r$ and $\psi'$ such that $(t'_{i-1}, t''_r . \text{out}(c_r, x), \psi') \in \text{trace}(P_{a_i . \cup \{b \rightarrow v'_i\}})$.
- for all $t'_{i} = t'_{i-1} = t'$, we have shown that blocking($t'_{i} . \text{out}(c_r, x)$, $P_{a_i}$), and thus for all $t''_r = t_r$ and all $\psi'$, $(t''_r . t''_r . \text{out}(c_r, x), \psi') \notin \text{trace}(P_{a_i . \cup \{b \rightarrow v'_i\}})$.

Therefore, the processes $P_{a_i . \cup \{b \rightarrow v'_i\}}$ and $P_{a_i . \cup \{b \rightarrow v_i\}}$ 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 $t_i = t$, of the form $t'_i . \text{out}(c_r, x)$, such that $\neg$-blocking($t_i$, $P_{a_i}$). In other words, there exists $\phi_i$ such that $(t_i, \phi_i) \in \text{trace}(P_{a_i})$. By assumption 1, there exists $V_i$ such that $\phi_i(x) \in \text{R}(\rho(V_i))$, i.e. result($t_i, \phi_i, V_i$). Let $V'' \overset{\text{def}}{=} V_m$. Note that $V_0 = V$.

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

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,

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

By definition, this means

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

which concludes the proof. \hfill \Box

**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 $v_{\text{neutral}}$. Assume FF holds.

Let $a$ be an affection of votes, and let $(t' . \text{out}(c_r, x), \phi) \in \text{trace}(P_a)$. Let $t \overset{\text{def}}{=} t' . \text{out}(c_r, x)$. 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 $\phi(x) \in \text{R}(\rho(V | \exists a . \text{Voted}(a, v) \in t \lor V_c))$.

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

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

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 $\ast$, so is $\rho(V_{\text{change}})$.

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

**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),

$$FF \Rightarrow V$$
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 which concludes the proof. □

As before, since \( \rho \), by assumption 3, \( \rho(\bar{\phi} | \exists a. \text{Voted}(a, \nu) \in t) \cup \bar{V}_c \).

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

Let \( V_{\text{wanted}} \) be the multiset of all intended honest votes in \( t \) that are not \( v_{\text{special}} \). By FF, there exists a multiset \( V_c \) such that \( \rho(V) \ast \rho(V_{\text{change}}) = \rho(V_{\text{wanted}}) \ast \rho(V_c) \).

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

which implies by assumption 7 that \( \rho(V) = \rho(V_{\text{wanted}} \uplus V_c) \).

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

Let \( k' \) be the number of intended votes for \( v_{\text{special}} \) in \( t \) and \( V_{\text{wanted}} \) be the number of intended votes for \( v_{\text{special}} \) in \( t \) such that \( V_{\text{change}} \) is \( k' \cdot v_{\text{special}} \).

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

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 \( v_{\text{special}} \neq V_{\text{wanted}} \uplus V_c'(v_{\text{special}}) \), \( V_c' = V_c'' \uplus k' \cdot v_{\text{special}} \) for some \( V_c'' \). Therefore we have

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

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 \textit{piecewise tally} property. In the case of \textit{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

\[
\forall b \in BB_1, \forall b' \in BB_2, \text{open}_id(b) \neq \text{open}_id(b')
\]

then the tally can be computed separately:

\[
\text{Tally}((BB_1 \cup BB_2, sk, U)) = \text{Tally}((BB_1, sk, U)) + \text{Tally}((BB_2, sk, U)).
\]

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

\[
\forall b \in BB_1, \forall b' \in BB_2, \text{open}_cred(b, sk, U) \neq \text{open}_cred(b', sk, U)
\]

then the tally can be computed separately:

\[
\text{Tally}((BB_1 \cup BB_2, sk, U)) = \text{Tally}((BB_1, sk, U)) + \text{Tally}((BB_2, sk, U)).
\]

(2) In the case of \textit{id}-based schemes only, the tally only counts ballots cast with registered \textit{ids}, i.e. \(\forall BB, sk, U\. \text{Tally}(BB, sk, U) = \text{Tally}(BB', sk, U)\) where \(BB' = \{b \in BB \mid (\text{open}_id(b), *) \in U\}\).

(3) Registering more voters does not change the tally. In the case of \textit{id}-based schemes, the assumption is that for all board \(BB\), election key \(sk\) and list of voters \(U\), if \(U, U'\) have no \textit{id} in common and \(\forall b \in BB\. (open_id(b), *) \notin U'\), then \(\forall BB, sk, U\. \text{Tally}(BB, sk, U) = \text{Tally}(BB, sk, U \cup U')\).

In the case of credential-based schemes, the assumption is that if \(U, U'\) share no credentials and \(\forall b \in BB\. (*, \text{open}_cred(b, sk, U \cup U')) \notin U'\), then \(\forall BB, sk, U\. \text{Tally}(BB, sk, U) = \text{Tally}(BB, sk, U \cup U')\).

(4) The voting scheme is \textit{correct}, i.e., for all distinct identities \(U = id_1, \ldots, id_n\), and credentials \(cred_1, \ldots, cred_n\), for all votes \(v_1, \ldots, v_n\), for all election keys \((pk, sk)\), if \(BB = [\text{Vote}(id_1, cred_1, pk, v_1), \ldots, id_n, cred_n]\), then

\[
\text{Tally}(BB, pk, sk) = \rho(v_1, \ldots, v_n).
\]

(5) There exists a neutral vote \(v_{\text{neutral}} \in V\), such that \(\rho(v_{\text{neutral}})\) is neutral for \(\ast\).

(6) Given a multiset of valid votes \(V\) and a result \(r\), it is possible to efficiently decide whether \(r\) can be decomposed into \(\rho(V) \ast \rho(V')\) for some multiset \(V'\) of valid votes.

That is to say, there exists a PPTM \(D\) such that

\[
\forall r, V\. \quad D(r, V) = 1 \iff \exists V'. \quad r = \rho(V) \ast \rho(V').
\]

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 \textit{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 \((b, \ast)\), and the last (resp. first) ballot \(b' \in BB\) such that \(\text{open}_id(b') = id\) is \(b\), then \(\text{VerifVoter}(id, cred, L, BB)\) 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 \((b, \ast)\), and the last (resp. first) ballot \(b' \in BB\) such that \(\text{open}_cred(b', sk, U) = cred\) is \(b\), then \(\text{VerifVoter}(id, cred, L, BB)\) 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 \textit{id}-based schemes: the assumption is that if \(\forall b \in BB'\. \text{open}_id(b) \neq id\) then \(\text{VerifVoter}(id, cred, L, BB) \iff \text{VerifVoter}(id, cred, L, BB \cup BB')\).

- For credential-based schemes: the assumption is that if \(\forall b \in BB'\. \text{open}_cred(b, sk, U) \neq cred\) then \(\text{VerifVoter}(id, cred, L, BB) \iff \text{VerifVoter}(id, cred, L, BB \cup BB')\).

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

\[
\forall A\. \quad P\left[\text{Exp}^{\text{ValidTally}}_A(\lambda) = 1\right] \text{ is negligible,}
\]

or that no polynomial adversary wins \(\text{Exp}^{\text{NM}}\) with non-negligible probability, i.e.

\[
\forall A\. \quad P\left[\text{Exp}^{\text{NM}}_A(\lambda) = 1\right] \text{ is negligible,}
\]

where \(\text{Exp}^{\text{ValidTally}}\) is defined on Figure 6 and \(\text{Exp}^{\text{NM}}\) is defined on Figure 7.
We consider the case of protocols where the revote policy is to count only the last ballot (for each 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 \( O^{\text{vote}} \) and \( O^{\text{vote}}_P \) with \( O^{\text{vote},i} \) and \( O^{\text{vote},f} \) described on Figure 12. These two oracles are analogous to \( O^{\text{vote}} \) and \( O^{\text{vote}}_P \), but keep only the first votes from each voter in the lists \( V_b, V_1 \), 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 \( \text{Exp}^{\text{ValidTally}} \) with non-negligible probability,
- and for credential-based schemes, assuming that no polynomial adversary wins \( \text{Exp}^{\text{NM}} \) with non-negligible probability,

\[
\forall \mathcal{A}, \quad \left| \Pr[A^{\text{priv},0}(\lambda) = 1] - \Pr[A^{\text{priv},1}(\lambda) = 1] \right| \text{ is negligible.}
\]

then

\[
\forall \mathcal{A}, \quad \Pr[A^{\text{verif}}(\lambda) = 1] \text{ is negligible.}
\]

**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 \( \text{Exp}^{\text{verif}} \). We consider an adversary \( \mathcal{B} = \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3 \) that attacks privacy, i.e. plays \( \text{Exp}^{\text{priv},B} \).

- \( \mathcal{B}_1^{O_{\text{vote}}^{\text{vote}}, O_{\text{cast}}}(\text{pk}) \) first simulates \( \mathcal{A}_1^{O_{\text{vote}}^{\text{vote}}, O_{\text{cast}}}(\text{pk}) \), 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 \( O_{\text{reg}} \). \( \mathcal{A} \) returns some state. \( \mathcal{B}_1 \) then calls \( O_{\text{reg}} \) another \( |U_1| \) times, on fresh identities that do not appear in \( U_1 \). It keeps a list \( U_2 \) of these fresh identities.

- \( \mathcal{B}_2^{O_{\text{vote}}^{\text{vote}}, O_{\text{cast}}}(\text{state}_1, \text{pk}) \) maintains a list \( L \), initially empty, which will be used to record the calls to \( O^{\text{vote}}_P \), and a representation of the current board \( BB' \). \( \mathcal{B}_2 \) first simulates \( \mathcal{A}_2^{O^{\text{vote}}_P, O_{\text{cast}}}(\text{state}, \text{pk}) \):
  - for each call to \( O^{\text{vote}}_P(id, v) \), provided \( id \in U_1 \), \( \mathcal{B} \) calls \( O^{\text{vote}}_P(id, v, \text{neutral}) \), 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, \( O^{\text{vote}}_P(id, v, \text{neutral}) \) returns nothing);
    - or adds \( id, v \) to \( L \) and removes any previous couple \( (id, v') \) (for any \( v' \)) from \( L \).
  - if \( O^{\text{vote}}_P(id, v, \text{neutral}) \) returns a ballot \( b \), \( \mathcal{B} \) then adds it to \( BB' \), and passes it to \( \mathcal{A}_2 \).
  - for each call to \( O_{\text{cast}}(id, b) \), provided \( id \in U_1 \), \( \mathcal{B} \) calls \( O_{\text{cast}}(id, b, \text{pk}) \), and, if \( \text{Valid}(id, b, BB', \text{pk}) \), adds it to \( BB' \).

Let then Voted be the list of the votes in \( L \): Voted = \( \{v \mid (id, v) \in L \} \). Let also BB\(_a\) be the value of the board BB\(_a\) 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 \( O^{\text{vote}}_P(id_1, \text{neutral}, v_1), \ldots, O^{\text{vote}}_P(id_l, \text{neutral}, v_l), \) 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 \( |U_1| = |U_2| \). Let 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 \( \text{neutral} \) if \( \beta = 0 \) and \( v_1, \ldots, v_l \) if \( \beta = 1 \).

At this point, we have \( BB = BB_a \uplus BB_b \).
\[ B_2 \text{ then returns a state indicating whether } A_2 \text{ has failed. Intuitively, } A \text{ is accurately simulated only if } \beta = 0, \text{ i.e. is shown a board where the votes it wanted to cast have indeed been cast. Hence whenever } A \text{ wins } \text{Exp}_\text{verif}, \text{ the simulated } A_2 \text{ failing can only mean that } \beta = 1. \]

- \text{Exp}_\text{priv} \text{ will then check that } \rho(V_0) = \rho(V_1), \text{ where } V_0 \text{ and } V_1 \text{ are the lists it keeps, which contain the first (or last, depending on the revote policy) votes } O_{\text{vote}}^b \text{ has been called on for each } id. \text{ Considering the definition of } B_2, \text{ at this point we always have }
\[
\rho(V_0) = \rho(V_1) = \rho(v_1, \ldots, v_l, v_{\text{neutral}}, \ldots, v_{\text{neutral}})
\]
\[ l \text{ times} \]

Hence this check necessarily succeeds, and \text{Exp}_\text{priv} \text{ computes } Tally(BB, sk, U). \]

- \[ B_1 \text{ obtains a result } r. \text{ If } r = \bot, \text{ then } B_1 \text{ returns } 1. \text{ If } A_1 \text{ blocked previously, } B_1 \text{ returns } \beta' = 1. \text{ Otherwise, } B_1 \text{ computes } D(r, \text{Voted}) \text{ and: } \]
  - if \[ \exists V_c. \ r = \rho(\text{Voted}) \ast \rho(V_c) \text{ then } B_1 \text{ returns } \beta' = 1 \]
  - otherwise, \[ B_1 \text{ returns } \beta' = 0. \]

We also construct an adversary \( C \), who plays the game \( \text{Exp}^\text{ValidTally} \).

- \( C_1 \) is identical to \( B_1 \).
- \( C_2 \) first draws at random a bit \( \beta'' \), and then simulates \( B_2 \) up to the point where \( B_2 \) has finished simulating \( A_2 \). \( C_2 \) keeps a list \( BB \), initially empty. It simulates each call to \( O_{\text{vote}}^b(id, v_0, v_1) \) \( B \) does by calling \( O_{\text{vote}}^b(id, v_{\beta''}) \), and appending the obtained ballot to \( BB \). It simulates each call to \( O_{\text{cast}}(id, b) \) by appending \( b \) to \( BB \), provided \( id \) is dishonest and \( \text{Valid}(id, b, BB, pk) \).
- Once \( C_2 \) has finished simulating \( B_2 \), it draws at random a number \( k \in [1, l] \). Recall that \( l \) is the number of different \( id \)s \( O_{\text{vote}}^b \) (and thus \( O_{\text{vote}}^v \)) has been called on, and is also the number of additional calls to \( O_{\text{vote}}^\nu \) \( B \) will make. Note that, at this point, \( BB \) in \( \text{Exp}^\text{ValidTally} \) is the same as \( BB_a \) in \( \text{Exp}^\text{priv, } c \). \( C \) then simulates the first \( k - 1 \) calls to \( O_{\text{vote}}(id, v_0, v_1) \), again by calling \( O_{\text{vote}}(id, v_{\beta''}) \). If the \( k \)th call is \( O_{\text{vote}}(id, v_0, v_1) \), \( C \) returns \( (BB, id, v_{\beta''}) \).

We will now prove that if \( A \) breaks individual verifiability, then \( \beta \) breaks privacy provided \( C \) does not break \( \text{Exp}^\text{ValidTally} \).

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

For any \( \beta \), assume \( \text{ValidTally}(BB_a, sk, U_1) \text{ holds and } \text{ValidTally}(BB_a \cup BB_b, sk, U_1 \cup U_2) \text{ does not}. \text{ Thus, by assumption 3, } \text{ValidTally}(BB_a \cup BB_b, sk, U_1 \cup \{id_1, cred_{id_1}, \ldots, (id_{k'}, cred_{id_{k'}})\}) \text{ does not hold either. Hence, there exists a smallest } k \in [1, l] \text{ such that } \text{ValidTally}(BB_a \cup \{id_{b_k+1}, \ldots, id_{b_{k'}}\}, sk, U_1 \cup U_2) \text{ holds and } \text{ValidTally}(BB_a \cup \{id_{b_k+1}, \ldots, id_{b_{k'}}\}, sk, U_1 \cup U_2 \cup \{(id_{b_k}, cred_{id_{b_k}})\}) \text{ does not, where } U_{b_k} = \{(id_{b_k}, cred_{id_{b_k}}), \ldots, (id_{b_{k'}-1}, cred_{id_{b_{k'}}})\}. \)

\( b_k \) has been added to \( BB_b \) by the \( k \)th call to \( O_{\text{vote}}^b \) by \( B_2 b^+_k \) = \( \text{Vote}(id, cred_{id_{b_k}}, pk, v_{\beta''}) \) for some \( \beta'' \).

Thus, provided \( C \) correctly guesses \( \beta'' = \beta \) and \( k \), \( C \) returns \( (BB_a \cup \{id_{b_k+1}, \ldots, id_{b_{k'}-1}\}, id_k, v_{\beta''}, \text{ the conditions on } BB \text{ in } \text{Exp}^\text{ValidTally} \) holds, and thus \( \text{Exp}^\text{ValidTally} = 1 \). Therefore, \( \text{ValidTally}(BB_a, sk, U_1) \text{ holds and } \text{ValidTally}(BB_a \cup BB_b, sk, U_1 \cup U_2) \text{ does not with probability at most } 2lP\left(\text{Exp}^\text{ValidTally} = 1\right) \), which is smaller than \( 2q(\lambda)P\left(\text{Exp}^\text{ValidTally} = 1\right) \) since \( l \leq q(\lambda) \).

Since \( BB_a \) only contains ballots cast for identities in \( U_1 \) and \( BB_b \) for identities in \( U_2 \), and \( U_1 \cap U_2 = \emptyset \), if \( \text{ValidTally}(BB_a \cup BB_b, sk, U_1 \cup U_2) \), by assumption 1 we have (regardless of \( \beta \))

\[ r = \text{Tally}(BB, sk, U) = \text{Tally}(BB_a \cup BB_b, sk, U) = \text{Tally}(BB_a, sk, U) \ast \text{Tally}(BB_b, sk, U) \]

In addition, by assumption 3, \( \text{Tally}(BB_a, sk, U) = \text{Tally}(BB_a, sk, U_1) \).

Hence
\[ r = \text{Tally}(BB_a, sk, U_1) \ast \text{Tally}(BB_b, sk, U). \]

- If \( \beta = 0 \): then \( \text{Exp}^\text{priv}(\lambda) = 1 \) if and only if \( B_3 \) returns 1 in this game, which happens either when \( A \) (simulated by \( B \)) blocks, or when \( A \) does not and \( \exists V_c. \ r = \rho(\text{Voted}) \ast \rho(V_c) \) or \( r = \bot \). Assume \( \text{ValidTally}(BB_a, sk, U_1) \Rightarrow \text{ValidTally}(BB_a \cup BB_b, sk, U) \), which, as we have established, holds except with probability at most \( 2q(\lambda)P\left(\text{Exp}^\text{ValidTally} = 1\right) \).

Let us first examine the case where \( A \) does not block and \( r \neq \bot \). Since \( r \neq \bot \), \( \text{ValidTally}(BB_a, sk, U_1) \) holds. Hence, \( \text{ValidTally}(BB_a \cup BB_b, sk, U) \) also holds, and as explained previously we thus know that \( r = \text{Tally}(BB_a, sk, U_1) \ast \text{Tally}(BB_b, sk, U) \). Since \( \beta = 0 \), \( BB_b \) only contains ballots for \( v_{\text{neutral}} \). Hence, by assumption 4, \( \text{Tally}(BB_b, sk, U) = \rho(v_{\text{neutral}}) = \rho(v_{\text{neutral}}) \). Thus \( r = \text{Tally}(BB_a, sk, U_1) \).

The condition \( \exists V_c. \ r = \rho(\text{Voted}) \ast \rho(V_c) \) is therefore equivalent to \( \exists V_c. \ r = \rho(\text{Voted}) \ast \rho(V_c) \). Since, in this case, \( A \) 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 \( \text{Exp}^\text{verif}(\lambda) \) does not return 1.
Hence $\text{Exp}^\text{Priv,0}_B(\lambda) = 1$ if and only if either $A$ (simulated by $B$) blocks, or constructs a board whose tally is $\perp$, or it does not and $\text{Exp}^\text{Verif}_A(\lambda) \neq 1$.

Since $\text{Exp}^\text{Verif}_A(\lambda)$ also does not return 1 when $A$ blocks or when the tally is $\perp$, this implies that, unless the implication $\text{ValidTally}(B_B, sk, U_1) \Rightarrow \text{ValidTally}(B_B \cup B_B, sk, U)$ is false, $\text{Exp}^\text{Priv,0}_B(\lambda)$ if and only if $\text{Exp}^\text{Verif}_A(\lambda) \neq 1$. Thus

$$P[\text{Exp}^\text{Priv,0}_B(\lambda) = 1] - P[\text{Exp}^\text{Verif}_A(\lambda) \neq 1] \leq 2q(\lambda)P[\text{Exp}_{C}^{\text{ValidTally}} = 1]. \quad (2)$$

- If $\beta = 1$: then $\text{Exp}^\text{Priv,1}_B(\lambda) = 1$ if and only if $B$ returns 1 in this game, which happens either when $A$ (simulated by $B$) blocks, or when it does not and $\exists V_c. r = \rho(V)$ or $r = \perp$.

Assume $\text{ValidTally}(B_B, sk, U_1) \Rightarrow \text{ValidTally}(B_B \cup B_B, sk, U)$, which, as we have established, holds except with probability at most $2q(\lambda)P[\text{Exp}_{C}^{\text{ValidTally}} = 1]$. Let us first examine the case where $A$ does not block and $r \neq \perp$. As in the $\beta = 0$ case, we thus have $r = \text{Tally}(B_B, sk, U_1) \star \text{Tally}(B_B, sk, U)$. Since $\beta = 1$, $B_B$ contains ballots for $v_1, \ldots, v_j$, i.e. for $V$. Hence, by assumption 4, $\text{Tally}(B_B, sk, U) = \rho(V)$, and therefore $r = \text{Tally}(B_B, sk, U_1) = \rho(V)$. By definition of $\text{Tally}$, there exists $V$ such that $\text{Tally}(B_B, sk, U_1) = \rho(V)$. Hence the condition $\exists V_c. r = \rho(V) \Rightarrow \rho(V)$ necessarily holds.

Therefore, unless the implication $\text{ValidTally}(B_B, sk, U_1) \Rightarrow \text{ValidTally}(B_B \cup B_B, sk, U)$ is false, $\text{Exp}^\text{Priv,1}_B(\lambda) = 1$ if and only if either $A$ (simulated by $B$) 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

$$1 - P[\text{Exp}^\text{Priv,1}_B(\lambda) = 1] \leq 2q(\lambda)P[\text{Exp}_{C}^{\text{ValidTally}} = 1]. \quad (3)$$

We thus have, using 4 and 5:

$$P[\text{Exp}^\text{Verif}_A(\lambda) = 1] = \left( 1 - P[\text{Exp}^\text{Verif}_A(\lambda) \neq 1] \right) + \left( P[\text{Exp}^\text{Priv,0}_B(\lambda) = 1] - P[\text{Exp}^\text{Priv,0}_B(\lambda) = 1] \right)$$

$$+ \left( P[\text{Exp}^\text{Priv,1}_B(\lambda) = 1] - P[\text{Exp}^\text{Priv,1}_B(\lambda) = 1] \right)$$

$$\leq P[\text{Exp}^\text{Priv,1}_B(\lambda) = 1] - P[\text{Exp}^\text{Priv,0}_B(\lambda) = 1] + 4q(\lambda)P[\text{Exp}_{C}^{\text{ValidTally}} = 1].$$

Therefore, if $A$ breaks individual verifiability, i.e. if $P[\text{Exp}^\text{Verif}_A(\lambda) = 1]$ is not negligible, then $B$ breaks privacy, or $C$ breaks $\text{Exp}^{\text{ValidTally}}$.

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

$$\forall A. P[\text{Exp}^{\text{NM}}_A(\lambda) = 1] \text{ is negligible}$$

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

Let $A = A_1, A_2$ be an adversary that breaks individual verifiability, i.e. wins $\text{Exp}^{\text{Verif}}$. We construct the adversary $B$, that plays $\text{Exp}^{\text{Priv}}$, as in the id-based case. We also construct $D$, that plays $\text{Exp}^{\text{NM}}$, and is similar to $C$ in the id-based case, except that $D$ simulates calls to $O^{\text{vote}}_B$ by calling $O_c$ instead of $O^{\text{vote}}_C$:

- $D_1$ is identical to $B_1$.
- $D_2$ first draws at random a bit $\beta''$, and then simulates $B_2$ up to the point where $B_2$ has finished simulating $A_2$. It simulates each call to $O^{\text{vote}}(id, v_0, v_1) B$ makes by calling $O_c(id, v, \beta''')$. Since $B_B$ is obtained by simulating $A$, which is polynomial, in all executions of $D$ the length of $B_B$ is bounded by some polynomial $p(\lambda)$. $D$ then draws at random a ballot in $B_B$, and returns it.

Similarly to the proof for the id-based case, and keeping the same notations, it then follows that $B_B$ contains a ballot with a credential in $U_2$ the same credential with probability at most $2p(\lambda)P[\text{Exp}^{\text{NM}}_D = 1]$. 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 $O_c(id, v)$ for some $id, v$ such that $(id, cred) \in U_2$, and the only calls to this oracle simulate calls made by $B$ to $O^{\text{vote}}$ when simulating $A$. Since $B$ only calls $O^{\text{vote}}_D$ on ids in $U_1$ when simulating $A$, this is contradictory. Thus, a ballot in $B_B$ with a credential in $U_2$ cannot have been produced by $O_c$. Hence, provided $D$ picked $\beta'' = \beta$, and picks the right ballot in $B_B$, which happens with probability at least $\frac{1}{p(\lambda)}$, $D$ wins $\text{Exp}^{\text{NM}}$.

Therefore, $B_B$ contains no ballot with a credential in $U_2$, except with probability at most $2p(\lambda)P[\text{Exp}^{\text{NM}}_D(\lambda) = 1]$.
We consider the case of protocols where the revote policy is to count only the last ballot (for each id).

Thus if A verifiability game to the case of a dishonest board and careful voters as follows.

Figure 14. These two oracles are analogous to O

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 proof proves that the result also holds if both the games

We adapt these definitions by replacing the oracles O

Note that by construction, all ballots in BB\beta have credentials in U2. Consequently, unless BB\beta contains a ballot with a credential in U2, by assumption 1, we have (regardless of β)

Note that, in these reductions, for each id in U1, B makes at most as many calls to O

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

Note that, in these reductions, for each id in U1, B makes at most as many calls to O

In the case of the first ballot, we adapt these definitions by replacing the oracles O

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 \lambda, we consider the game \text{Exp}^\text{verif–careful}_\mathcal{A}(\lambda) defined on Figure 13. The voting system is verifiable against a dishonest board with careful voters if

∀\mathcal{A}. P \left[ \text{Exp}^\text{verif–careful}_\mathcal{A}(\lambda) = 1 \right] is negligible.

In the case of the first ballot, we adapt these definitions by replacing the oracles O

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 Exp\text{ValidTally} with non-negligible probability,

• and for credential-based schemes, assuming that no polynomial adversary wins Exp\text{NM} with non-negligible probability,
Figure 14: Oracles for the verifiability and privacy games with dishonest boards and careful voters (revote policy = first vote)

\[
\begin{align*}
O^{v,c,f}_{\text{vote}} (id, v) & \quad \text{if } (id,*) \in U \cup \{\text{CU}\} \land (id,*) \notin \text{Voted} \\
& b \leftarrow \text{Vote}(id, \text{cred}_a, pk, v) \\
& \text{Voted} \leftarrow \text{Voted} \cup \{(id, v)\} \\
& L_{id} \leftarrow L_{id} \cup \{(b, v)\} \\
& \text{return } b \\
& \text{where } (id, \text{cred}_a) \in U
\end{align*}
\]

\[
\begin{align*}
O^{p,c,f}_{\text{vote}} (id, v, v_1) & \quad \text{if } (id,*) \in U \cup \{\text{CU}\} \land (id,*) \notin \text{Voted}, V_1 \text{ then} \\
& b \leftarrow \text{Vote}(id, \text{cred}_id, pk, v_0) \\
& V_0 \leftarrow V_0 \cup \{(id, v_0)\} \\
& V_1 \leftarrow V_1 \cup \{(id, v_1)\} \\
& L_{id} \leftarrow L_{id} \cup \{(b, v_0)\} \\
& \text{return } b \\
& \text{where } (id, \text{cred}_id) \in U
\end{align*}
\]

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

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

**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 \( \text{Exp}_{\mathcal{A}}^{\text{verif-careful}} \). 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 \( \text{Exp}_{\mathcal{B}}^{\text{priv-careful}} \).

- \( \mathcal{B}_1^{\text{O}_{\text{reg}}, \text{O}_{\text{cor}}}(pk) \) first simulates \( \mathcal{A}_1^{\text{O}_{\text{reg}}, \text{O}_{\text{cor}}}(pk) \), 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 \( \text{O}_{\text{reg}} \). \( \mathcal{A} \) returns some state \( \mathcal{B}_1 \) then calls \( \text{O}_{\text{reg}} \) another \( |U_1| \) times, on fresh identities that do not appear in \( U_1 \). It keeps a list \( U_2 \) of these fresh identities.

- \( \mathcal{B}_2^{\text{O}_{\text{vote}}}(\text{state}_0, pk) \) maintains a list \( L \), initially empty, which will be used to record the calls to \( \text{O}_{\text{vote}}^{c} \). \( \mathcal{B}_2 \) first simulates \( \mathcal{A}_2^{\text{O}_{\text{vote}}}(\text{state}, pk) \). For each call to \( \text{O}_{\text{vote}}^{c}(id, v) \), provided \( id \in U_1 \), \( \mathcal{B} \) calls \( \text{O}_{\text{vote}}^{c}(id, v, v_{\text{neutral}}) \) 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, \( \text{O}_{\text{vote}}^{c} \) 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, \( \text{O}_{\text{vote}}^{c} \) indeed returns a ballot).

The simulated \( \mathcal{A}_2 \) returns a board \( \mathcal{B}_2' \) and a state \( \text{state}'_2 \). Let \( \mathcal{B}_2'_a \) be the list of ballots in \( \mathcal{B}_2' \) that \( \mathcal{A}_2 \) was allowed to cast, i.e. \( \mathcal{B}_2'_a = \{ b \in \mathcal{B}_2' | \text{open}_a(b) \in U_1 \} \).

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

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

In any case, \( \mathcal{B}_2 \) calls \( \text{O}_{\text{vote}}^{c}(id_1, v_{\text{neutral}}, v_1), \ldots, \text{O}_{\text{vote}}^{c}(id_l, v_{\text{neutral}}, v_l) \), where \( v_1, \ldots, v_l \) are the elements of \( \text{Voted} \), 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 \( |U_1| = |U_2| \). \( \mathcal{B}_2 \) thus obtains \( l \) ballots \( b_1, \ldots, b_l \) for \( v_{\text{neutral}} \) if \( \beta = 0 \) and \( v_1, \ldots, v_l \) if \( \beta = 1 \). Let \( \mathcal{B}_2 \) be the list of these ballots. Let then \( \mathcal{B} = \mathcal{B}_2' \cup \mathcal{B}_2 \).

\( \mathcal{B}_2 \) then returns a \( \text{state}'_2 \) indicating whether \( \mathcal{A}_2 \) has failed, and the board \( \mathcal{B} \). 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 \( \text{Exp}_{\mathcal{A}}^{\text{verif-careful}} \), the simulated \( \mathcal{A}_2 \) failing can only mean that \( \beta = 1 \).

- \( \mathcal{B}_3^{\text{O}_{\text{happy}}}(\text{state}'_2) \) first simulates \( \mathcal{A}_3^{\text{O}_{\text{happy}}}(\text{state}'_2) \), unless \( \mathcal{A}_2 \) has failed, in which case it simply calls \( \text{O}_{\text{happy}}(\text{open}_a(b)) \) for each \( b \) occurring in \( \mathcal{B}_2' \).
$B_1$ then calls $O_{\text{happy}_{BB}}(id)$ for each $i \in [1, l]$. Since $BB'_{\beta}$ only contains ballots $b$ such that open_{id}(b) \in U_1$ (by definition), only $BB_b$ contains ballots for the $id_i \in U_2$. $BB_b$, by construction, contain exactly one ballot for each $id_i$, which is the ballot $b_i$ produced by the (only) call to $O_{\text{vote}}^c(id_i, *, *)$. Since the revote policy is to count the first (or last) ballot for each $id$, by assumption 7, $\text{VerifVoter}(id_i, cred_{id_i}, -id_i, BB)$ holds, and thus the call to $O_{\text{happy}_{BB}}(id_i)$ adds $id_i$ to $H$. After that step, $H$ thus necessarily contains at least $id_1, \ldots, id_l$.

- **Exp$^{\text{priv-careful}}$** 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 $O_{\text{vote}}^c$ has been called on for each $id$, and $H$ is the list of identities $O_{\text{happy}_{BB}}$ has successfully been called on. If $H_a$ and $H_b$ denote the value of the list $H$ at this point respectively in Exp$^{\text{verif-careful}}_A$ and Exp$^{\text{priv-careful}}_B$, we have $H_b = H_a \cup \{id_1, \ldots, id_l\}$.

  - we have established that $H_b$ contains $id_1, \ldots, id_l$,
  - and for all $(id, cred) \in U_1, id \in H_b$ if and only if $O_{\text{happy}_{BB}}(id)$ has been called by $B$ and VerifVoter$(id, cred, L_{id}, BB)$ succeeds, i.e. if and only if $O_{\text{happy}_{BB}}(id)$ was called by the simulated $A$ and VerifVoter$(id, cred, L_{id}, BB)$ succeeds. Since $BB = BB'_\beta \uplus BB_b$, and given the definition of $BB_b$, by assumption 8, VerifVoter$(id, cred, L_{id}, BB)$ is equivalent to VerifVoter$(id, cred, L_{id}, BB'_\beta)$, which is itself equivalent to VerifVoter$(id, cred, L_{id}, BB_a)$. Thus $id \in H_b$ if and only if $id \in H_a$.

Therefore, the test $\forall id. (id, *) \in V_0, V_1 = id \in H$ succeeds in Exp$^{\text{verif-careful}}_A$ if and only if it succeeds in Exp$^{\text{priv-careful}}_B$.

- **Exp$^{\text{priv-careful}}$** will then check that $\rho(V_0) = \rho(V_1)$. Considering the definition of $B_2$, if this point is reached, we always have

$$\rho(V_0) = \rho(V_1) = \rho(v_1, \ldots, v_l, v_{\text{neutral}}, \ldots, v_{\text{neutral}})$$

$l$ times

Here the check necessarily succeeds, and Exp$^{\text{priv-careful}}_B$ computes $\text{Tally}(BB, sk, U)$.

- $B_2$ obtains a result $r$ (or $\perp$, if the previous tests by Exp$^{\text{priv}}$ failed or the tally returned $\perp$). If
  - $A_2$ blocked previously;
  - or $r = \perp$, which means, as we have established that $\rho(V_0) = \rho(V_1)$ always holds, that the test $\forall id. (id, *) \in V_0, V_1 = id \in H$ fails, or the tally returns $\perp$;
  - $B$ returns $\beta' = 1$.
  
  Otherwise, $B$ computes $\text{Dr}(r, \text{Voted})$ and:
  - $r = \rho(\text{Voted}) \neq \rho(V_2)$ then $B$ returns $\beta' = 1$;
  - otherwise, $B$ returns $\beta' = 0$.

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

- $C_1$ is identical to $B_1$.
- $C_2$ first draws at random a bit $\beta''$, and then simulates $B_2$ up to the point where $B_2$ has finished simulating $A_2$. It simulates each call to $O_{\text{vote}}^c(id, v_0, v_1) B$ does by calling $O_{\text{vote}}^c(id, v_{\beta''})$.
- Once $C_2$ has finished simulating $B_2$ up to the point $B_2$ has simulated $A_2$ and obtained a board $BB'_\beta$, it draws at random a number $k \in [1, l]$ Recall that $l$ is the number of different $ids$ $O_{\text{vote}}^c$ has been called on, and is also the number of additional calls to $O_{\text{vote}}^c B$ will make. $C$ then simulates the first $k-1$ calls to $O_{\text{vote}}^c(id, v_0, v_1)$, again by calling $O_{\text{vote}}^c(id, v_{\beta''})$. It obtains $k-1$ ballots, and stores them in a board $BB_b = [(id_1, b_1), \ldots, (id_{k-1}, b_{k-1})]$. If the $k$th call is $O_{\text{vote}}^c(id, v_0, v_1), C$ returns $(BB'_\beta \uplus BB_b, id, v_{\beta''})$.

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

The adversary $C$ is polynomial, i.e. there exists a polynomial $g(l)$ bounding its number of operations.

For any $\beta$, assume ValidTally$(BB_{\alpha}, sk, U_1)$ holds and ValidTally$(BB'_\beta \uplus BB_b, sk, U_1 \cup U_2)$ does not. Thus, by assumptions 3 and 2, ValidTally$(BB'_\beta \uplus BB_b, sk, U_1 \cup \{(id_1, cred_{id_1}), \ldots, (id_l, cred_{id_l})\})$ does not hold either, but ValidTally$(BB_{\alpha}, sk, U_1)$ does. Hence, there exists a smallest $k \in [1, l]$ such that ValidTally$(BB_{\alpha}' \cup \{(id_1, b_1), \ldots, (id_{k-1}, b_{k-1})\}, sk, U_1 \cup U_2)$ holds and ValidTally$(BB'_\beta \cup \{(id_1, b_1), \ldots, (id_{k-1}, b_{k-1})\}, sk, U_1 \cup U_2 \cup \{(id_k, cred_{id_k})\})$ does not, where $U'_2 = \{(id_1, cred_{id_1}), \ldots, (id_{k-1}, cred_{id_{k-1}})\}$, $b_k$ was produced by the $k$th call to $O_{\text{vote}}^c$ by $B$: $b_k = \text{Vote}(id_k, cred_{id_k}, pk, v_\beta)$. Thus, provided $C$ correctly guesses $\beta'' = \beta$ and $k$, $C$ returns $(BB'_\beta \uplus \{(b_k, \perp, \perp, id_k, v_\beta)\})$ the conditions on $BB$ in Exp$^{\text{ValidTally}}_C$ holds, and thus Exp$^{\text{ValidTally}}_C = 1$. Therefore, ValidTally$(BB_{\alpha}, sk, U_1)$ holds and ValidTally$(BB_{\alpha} \uplus BB_b, sk, U_1 \cup U_2)$ does not with probability at most $2lP[\text{Exp}^{\text{ValidTally}}_C = 1]$, which is smaller than $2g(l)P[\text{Exp}^{\text{ValidTally}}_C = 1]$ since $l \leq g(l)$.

Since $BB'_\beta$ only contains ballots cast for identities in $U_1$ (by definition) and $BB_b$ for identities in $U_2$, and $U_1 \cap U_2 = \emptyset$, if ValidTally$(BB'_\beta \uplus BB_b, sk, U_1 \cup U_2)$, by assumption 1 we have (regardless of $\beta$)

$$r = \text{Tally}(BB, sk, U) = \text{Tally}(BB'_\beta \uplus BB_b, sk, U) = \text{Tally}(BB_{\alpha}, sk, U) + \text{Tally}(BB_b, sk, U)$$

25
In addition, by assumption 3, $\text{Tally}(BB'_a, sk, U) = \text{Tally}(BB'_a, sk, U_1)$. Hence

$$r = \text{Tally}(BB'_a, sk, U_1) * \text{Tally}(BB_b, sk, U).$$

Moreover, since $BB_a \setminus BB_a'$ contains only ballots for identities not in $U_1$, by assumption 2, $\text{Tally}(BB_a', sk, U_1) = \text{Tally}(BB_a, sk, U_1)$, and thus

$$r = \text{Tally}(BB_a, sk, U_1) * \text{Tally}(BB_b, sk, U).$$

- If $\beta = 0$: then $\text{Exp}_{BB}^{\text{priv-careful},0}(\lambda) = 1$ if and only if $B_4$ (called on the result $r$) returns 1 in this game, which happens
  - either when $A$ (simulated by $B$) blocks;
  - or when $r = \bot$, i.e., as already mentioned, if the tally returns $\bot$ or the test $\forall \beta \cdot (id, \ast) \in V_0, V_1 \Rightarrow id \in H$ fails in $\text{Exp}_{BB}^{\text{priv-careful},\beta}$;
  - or when $A$ does not block, the previous test succeeds, and $\exists V_c. r = \rho(\text{Voted}) \ast \rho(V_c)$.

Assume $\text{ValidTally}(BB_a, sk, U_1) \Rightarrow \text{ValidTally}(BB_a' \cup BB_b, sk, U)$, which, as we have established, holds except with probability at most

$$2q(\lambda) \Pr_{\text{Exp}_{C}^{\text{ValidTally}}} = 1.$$ 

We have already established that the test $\forall \beta \cdot (id, \ast) \in V_0, V_1 \Rightarrow id \in H$ succeeds in $\text{Exp}_{BB}^{\text{priv-careful},\beta}$ if and only if it succeeds in $\text{Exp}_{A}^{\text{verif-careful}}$.

Let us first examine the case where $A$ does not block, the tally does not return $\bot$, and the test $\forall \beta \cdot (id, \ast) \in V_0, V_1 \Rightarrow id \in H$ succeeds. Since $r \neq \bot$, $\text{ValidTally}(BB_a, sk, U_1)$ holds. Hence, $\text{ValidTally}(BB_a \cup BB_b, sk, U)$ also holds, and as explained previously we thus know that $r = \text{Tally}(BB_a, sk, U_1) * \text{Tally}(BB_b, sk, U)$.

Since $\beta = 0$, $BB_b$ only contains ballots for $\nu_{\text{neutral}}$, and then by assumption 4, $\text{Tally}(BB_b, sk, U) = \rho(\nu_{\text{neutral}}) \ast \rho(\nu_{\text{neutral}})$. Thus $r = \text{Tally}(BB_a, sk, U_1)$.

The condition $\exists V_c. r = \rho(\text{Voted}) \ast \rho(V_c)$ is therefore equivalent to $\exists V_c. \text{Tally}(BB_a, sk, U_1) = \rho(\text{Voted}) \ast \rho(V_c)$. Since, in this case,

- $A$ has been accurately simulated without blocking,
- $BB_a'$ is the board it returns,
- the test $\forall \beta \cdot (id, \ast) \in V_0, V_1 \Rightarrow id \in H$ succeeds in $\text{Exp}_{A}^{\text{verif-careful}}$,

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

Hence, in that case, $\text{Exp}_{BB}^{\text{priv-careful},0}(\lambda) = 1$ if and only if

- either $A$ (simulated by $B$) blocks;
- or the test $\forall \beta \cdot (id, \ast) \in V_0, V_1 \Rightarrow id \in H$ fails in $\text{Exp}_{A}^{\text{verif-careful}}$;
- or the tally returns $\bot$;
- or $A$ does not block, the previous test succeeds, the tally does not return $\bot$, and $\text{Exp}_{A}^{\text{verif-careful}}(\lambda)$ does not return 1.

Since $\text{Exp}_{A}^{\text{verif-careful}}(\lambda)$ also does not return 1 when $A$ blocks, or when the test fails, or when the tally returns $\bot$, this implies that, unless the implication $\text{ValidTally}(BB_a, sk, U_1) \Rightarrow \text{ValidTally}(BB_a' \cup BB_b, sk, U)$ is false, $\text{Exp}_{BB}^{\text{priv-careful},0}(\lambda) = 1$ if and only if

$$\Pr_{\text{Exp}_{C}^{\text{ValidTally}}} = 1.$$ (4)

- If $\beta = 1$: then $\text{Exp}_{BB}^{\text{priv-careful},1}(\lambda) = 1$ if and only if $B_4$ (called on the result $r$) returns 1 in this game, which happens
  - either when $A$ (simulated by $B$) blocks;
  - or when $r = \bot$, i.e., as already mentioned, if the tally returns $\bot$ or the test $\forall \beta \cdot (id, \ast) \in V_0, V_1 \Rightarrow id \in H$ fails in $\text{Exp}_{BB}^{\text{priv-careful},\beta}$;
  - or when $A$ does not block, the previous test succeeds, the tally returns $r \neq \bot$, and $\exists V_c. r = \rho(\text{Voted}) \ast \rho(V_c)$.

Assume $\text{ValidTally}(BB_a, sk, U_1) \Rightarrow \text{ValidTally}(BB_a' \cup BB_b, sk, U)$, which, as we have established, holds except with probability at most

$$2q(\lambda) \Pr_{\text{Exp}_{C}^{\text{ValidTally}}} = 1.$$ 

We have already established that the test $\forall \beta \cdot (id, \ast) \in V_0, V_1 \Rightarrow id \in H$ succeeds in $\text{Exp}_{BB}^{\text{priv-careful},\beta}$ if and only if it succeeds in $\text{Exp}_{A}^{\text{verif-careful}}$.

Let us first examine the case where $A$ does not block, the tally does not return $\bot$, and the test $\forall \beta \cdot (id, \ast) \in V_0, V_1 \Rightarrow id \in H$ succeeds.

As in the $\beta = 0$ case, we thus have $r = \text{Tally}(BB_a, sk, U_1) * \text{Tally}(BB_b, sk, U)$. Since $\beta = 1$, $BB_b$ contains ballots for $v_1, \ldots, v_l$, i.e. for Voted. Hence, by assumption 4, $\text{Tally}(BB_b, sk, U) = \rho(\text{Voted})$, and therefore $r = \text{Tally}(BB_a, sk, U_1) \ast \rho(\text{Voted})$. By definition of Tally, there exists $V$ such that $\text{Tally}(BB_a, sk, U_1) = \rho(V)$. Hence the condition $\exists V_c. r = \rho(\text{Voted}) \ast \rho(V_c)$ necessarily holds.

In addition we know that the test $\forall \beta \cdot (id, \ast) \in V_0, V_1 \Rightarrow id \in H$ succeeds in $\text{Exp}_{A}^{\text{verif-careful}}$. Therefore, unless the implication $\text{ValidTally}(BB_a, sk, U_1) \Rightarrow \text{ValidTally}(BB_a' \cup BB_b, sk, U)$ is false, $\text{Exp}_{BB}^{\text{priv-careful},1}(\lambda) = 1$ if and only if

- either $A$ (simulated by $B$) blocks;
- or the test $\forall \beta \cdot (id, \ast) \in V_0, V_1 \Rightarrow id \in H$ fails in $\text{Exp}_{A}^{\text{verif-careful}}$;
- or the tally returns $\bot$;
which means, by assumption 3, that

\[ 1 - P\left[ \Exp^{\text{priv-careful},1}_{\beta}(\lambda) = 1 \right] \leq 2q(\lambda) P\left[ \Exp^{\text{ValidTally}}_{\beta} = 1 \right]. \]  

(5)

We thus have, using 4 and 5:

\[
P\left[ \Exp^{\text{verif-careful}}_{\beta}(\lambda) = 1 \right] = \left(1 - P\left[ \Exp^{\text{verif-careful,0}}_{\beta} = 1 \right] \right) + \left(P\left[ \Exp^{\text{verif-careful,1}}_{\beta}(\lambda) = 1 \right] - P\left[ \Exp^{\text{verif-careful,1}}_{\beta}(\lambda) = 1 \right] \right)
+ \left(P\left[ \Exp^{\text{verif-careful,0}}_{\beta} = 1 \right] - P\left[ \Exp^{\text{verif-careful,0}}_{\beta} = 1 \right] \right) + 4q(\lambda) P\left[ \Exp^{\text{ValidTally}}_{\beta} = 1 \right]
\]

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

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

\[ \forall \beta. \ P\left[ \Exp^{\text{NM}}_{\beta}(\lambda) = 1 \right] \text{ is negligible.} \]

Let \( A = A_1, A_2, A_3 \) be an adversary that breaks individual verifiability, i.e. wins \( \Exp^{\text{verif-careful}}_{\beta} \). We construct the adversary \( B \), that plays \( \Exp^{\text{priv-careful}}_{\beta} \), as in the id-based case. However, contrary to the id-based case, \( B \) does not remove any ballots from \( \BB_{id} \), and simply uses \( \BB'_{id} = \BB_{id} \). We also construct \( D \), that plays \( \Exp^{\text{NM}}_{\beta} \), and is similar to \( C \) in the id-based case, except that \( D \) simulates calls to \( \Opt{\text{vote}}^{\text{NM}} \) by calling \( O_{\beta} \) instead of \( \Opt{\text{vote}}^{\text{NM}} \):

- \( D_1 \) is identical to \( B_1 \).
- \( D_2 \) first draws at random a bit \( \beta'' \), and then simulates \( B_2 \) up to the point where \( B_2 \) has finished simulating \( A_2 \). It simulates each call to \( \Opt{\text{vote}}^{\text{NM}}(\beta''') \) by calling \( O_{\beta}(\beta'', \nu_0, \nu_1) \) \( B \) makes by calling \( O_{\beta}(\nu_0, \nu_1) \).
- Once \( D_2 \) has finished simulating \( B_2 \) up to the point \( B_2 \) has simulated \( A_2 \), \( D \) obtains a board \( B_{id} \) (which is the same as \( B_{id} \) in \( \Exp^{\text{priv-careful}}_{\beta} \)). Since \( B_{id} \) is obtained by simulating \( A \), which is polynomial, in all executions of \( D \) the length of \( B_{id} \) is bounded by some polynomial \( p(\lambda) \). \( D \) then draws at random a ballot in \( B_{id} \), and returns it.

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

Therefore, \( B_{id} \) contains no ballot with a credential in \( U_2 \), except with probability at most \( 2p(\lambda) P\left[ \Opt{\text{vote}}^{\text{NM}} = 1 \right] \).

Note that by construction, all ballots in \( B_{id} \) have credentials in \( U_2 \). Consequently, unless \( B_{id} \) contains a ballot with a credential in \( U_2 \), by assumption 1, we have (regardless of \( \beta \))

\[ r = \text{Tally}(B_{id}, sk, U) = \text{Tally}(B_{id} \cup B_{id}, sk, U) = \text{Tally}(B_{id}, sk, U) \]

which means, by assumption 3, that

\[ r = \text{Tally}(B_{id}, sk, U_1) \]

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

Note that, in these reduction, for each \( id \) in \( U_2 \), \( B \) makes at most as many calls to \( \Opt{\text{vote}}^{\text{NM}} \) as \( A \) makes to \( \Opt{\text{vote}}^{\text{NM}} \). In addition, for each \( id \) in \( U_2 \), \( B \) makes at most one call to \( \Opt{\text{vote}}^{\text{NM}} \).

Thus, the exact same proof proves that the result also holds if both the games \( \Exp^{\text{priv-careful}}_{\beta} \) and \( \Exp^{\text{verif-careful}}_{\beta} \) are modified to prevent revote, by allowing only one call to \( \Opt{\text{vote}}^{\text{NM}} \) for each \( id \).
where (presented on Figure 7) with non-negligible probability. We will also in some cases use the assumption that the ballot creation function has a

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 Exp^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

$$\forall \mathcal{A}. \ |P[\text{Exp}^{\text{ind},0}(\lambda) = 1] - P[\text{Exp}^{\text{ind},1}(\lambda) = 1]|$$

is negligible

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

$$\forall \text{id}, \text{cred}, \text{v}. \ \text{open}($$

It also assumes a function 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 open(b) = id and extract(b) = c. This corresponds to the case of Helios and Belenos in our case study. For the other protocols we study, extract(b) = b.

C.2 Proofs

C.2.1 Civitas is private for Exp^priv.

**Theorem C1.** Assuming no adversary wins Exp^ind nor Exp^NM with non-negligible probability, Civitas is private for Exp^priv.

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

- \( \mathcal{B}_1 \) first simulates \( \mathcal{A}_1 \) in \( \text{open}(\text{cred}, \text{v}) \):
  - \( \mathcal{B}_1 \) maintains lists \( V_0, V_1, BB \), initially empty. It will be used to simulate the lists with the same name in Exp^priv.
  - \( \mathcal{B}_2 \) will use lists hL, cl, initially empty. \( \mathcal{B}_2 \) first simulates \( \mathcal{A}_2 \) in \( \text{open}(\text{cred}, \text{v}) \):
    - for each call to \( \text{open}(\text{cred}, \text{v}) \), provided id \( \in U_1 \cup U_2 \), \( \mathcal{B} \) checks whether id is already present in \( V_0, V_1 \). Provided it is not, \( \mathcal{B} \) then retrieves id’s credential cred from \( U_2 \), and calls \( \mathcal{A}_1 \) in \( \text{open}(\text{cred}, \text{v}) \). \( \mathcal{B} \) obtains a ballot b. \( \mathcal{B} \) then appends b to BB and hL, and checks they are equal.
    - for each call to \( \text{open}(\text{cred}, \text{v}) \), provided id \( \in U_1 \) and Valid(id, v, BB, pk), \( \mathcal{B} \) appends b to BB, and to cl.
    - We write \( \text{cl} \cup \text{hL} \) (resp. \( \text{cl} \cap \text{hL} \)) the sublist of cl (in the same order) of ballots that do not occur (resp. do occur) in hL.
  - \( \mathcal{B}_2 \) then computes \( \rho(V_0), \rho(V_1) \), and check they are equal.
  - \( \mathcal{B}_2 \) calls \( \mathcal{A}_3 \) in \( \text{cl} \cup \text{hL} \), 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 \( \text{cl} \cup \text{hL} \) has been generated by a call to \( \mathcal{A}_1 \), which means that \( \mathcal{A}_2 \) accepts to open all ballots in \( \text{cl} \cup \text{hL} \).
  - \( \mathcal{B}_2 \) computes \( \rho(V_0) \rho(L') \), calls \( \mathcal{A}_3 \) on r, and obtains a bit \( \beta' \). \( \mathcal{B}_2 \) returns \( \beta' \).
Note that the lists \( BB, V_0, V_1 \) in \( B \) are equal to the lists of the same name in \( \text{Exp}^\text{priv,}\beta_{\mathcal{A}} \). 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 \( \text{Exp}^\text{priv,}\beta_{\mathcal{A}} \).

We also construct an adversary \( C \), that plays the game \( \text{Exp}^\text{NM} \).

- \( C_1 \) 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 \( \beta'' \). Similarly to \( \mathcal{B}_2 \), \( C_2 \) then simulates \( \mathcal{A}_2 \) and uses lists \( BB, cl, hL, V \):
  - for each call to \( O_{\text{vote}}'(id, v_0, v_1) \), provided \( id \in U_1 \setminus C U_1 \), \( C \) calls \( O_c(id, v_{\beta''}) \).\( C \) 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 \( O_{\text{cast}}(id, b) \), provided \( id \in C U_1 \) and Valid\( (id, b, BB, pk) \), \( C \) 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 \( \text{Exp}^\text{NM} \) at the point \( C_2 \) returns, and for \( B \) in \( \text{Exp}^\text{ind} \) at the point \( B_2 \) returns. Similarly \( V \) in \( C \) is the same as \( V_{\beta''} \) in \( B \). Let us also notice that \( hL \) in \( C \) is equal to the list \( L \) in the game \( \text{Exp}^\text{NM} \).

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

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 \( \beta \), assume \( cl \setminus hL \) contains a ballot \( b \) such that there exists a honest \( (\ast, \text{cred}) \in U \setminus C U \) such that open\( _{\text{cred}}(b, sk, U) = \text{cred} \) (note that \( U, C U \) in \( \text{Exp}^\text{NM} \) are equal to \( U_1, C U_1 \) in \( B \)). Thus, provided \( C \) correctly guesses \( \beta'' = \beta \), and chooses this \( b \) from \( cl \), the condition \( b \not\in L \land \exists (\ast, \text{cred}) \in U \setminus C U \). open\( _{\text{cred}}(b, sk, U) = \text{cred} \in \text{Exp}^\text{NM} \) holds. Indeed, since \( b \not\in hL, b \not\in L \) Thus \( \text{Exp}^\text{NM} = 1 \).

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

For any \( \beta \), \( \text{Exp}^\text{ind,}\beta_{\mathcal{B}}(\lambda) = 1 \) if and only if \( B_2 \) returns 1 in this game.

Assume \( cl \setminus hL \) does not contain any ballot associated with a honest credential (i.e. a credential in \( U_1 \setminus C U_1 \)), which, as we have established, holds except with probability at most \( 2q(\lambda) \cdot P \left[ \text{Exp}^\text{NM} = 1 \right] \).

Let us then show that \( \mathcal{A}_3 \) is accurately simulated by \( B_2 \), i.e. it is simulated by \( B_2 \) only when it is called in \( \text{Exp}^\text{priv,}\beta_{\mathcal{A}} \), that is, when \( \rho(V_0) = \rho(V_1) \); and it is provided with the actually tally of the board \( BB \) (which \( \mathcal{A}_2 \) interacted with).

Indeed, by construction of \( B_2 \), \( BB = hL \cup cl \) is an interleaving of the ballots from \( hL \) and \( cl \).

By assumption, \( cl \setminus hL \) contains no ballot from honest credentials, while by construction \( hL \) (and thus \( cl \cap hL \)) only contains ballots from honest credentials. Hence, the list \( BB \) of ballots in \( BB \) associated with dishonest credentials is \( cl \setminus hL \) (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 \( cl \). 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

\[
\text{Tally}(BB, sk, U_1) = \text{Tally}(BB_h, sk, U_1) \ast \text{Tally}(BB_d, sk, U_1) = \text{Tally}(hL, sk, U_1) \ast \text{Tally}(cl \setminus hL, sk, U_1).
\]

\( hL \) contains ballots for either the votes in \( V_0 \) or those in \( V_1 \), depending on \( \beta \). Since at that point \( \rho(V_0) = \rho(V_1) \), we have \( \rho(V_0) = \text{Tally}(hL, sk, U_1) \). In addition, the oracle \( O'_L \) returns the list \( L \) of the credentials and votes of each ballot in \( cl \setminus hL \). Since \( L' \) is the list of the first vote for each credential in \( L \), we thus have \( \text{Tally}(cl \setminus hL, sk, U_1) = \rho(L') \). Therefore, \( \rho(V_0) = \rho(L') \), which is the result computed by \( B_2 \), is indeed \( \text{Tally}(BB, sk, U_1) \), which concludes the proof that \( \mathcal{A}_3 \) is accurately simulated by \( B_2 \).

Hence, unless \( cl \setminus hL \) contains a ballot associated with a honest credential, \( \text{Exp}^\text{ind,}\beta_{\mathcal{B}}(\lambda) = 1 \) if and only if the accurately simulated \( \mathcal{A}_3 \) returns 1, i.e. if and only if \( \text{Exp}^\text{priv,}\beta_{\mathcal{A}}(\lambda) = 1 \).

Thus
\[
\left| P \left[ \text{Exp}^\text{ind,}\beta_{\mathcal{B}}(\lambda) = 1 \right] - P \left[ \text{Exp}^\text{priv,}\beta_{\mathcal{A}}(\lambda) \neq 1 \right] \right| \leq 2q(\lambda) \cdot P \left[ \text{Exp}^\text{NM} = 1 \right]. \tag{6}
\]

We thus have:
We will now prove that if $A$ wins $\text{Exp}_{A}$, then $B$ breaks $\text{Exp}^\text{ind}$, or $C$ breaks $\text{Exp}^\text{NM}$.

\begin{proof}

Let $A = A_1, A_2, A_3, A_4$ be an adversary that wins $\text{Exp}_{A}^\text{priv-careful}$. We consider an adversary $B = B_1, B_2, B_3$ that plays $\text{Exp}_{B}^\text{ind}$:

- $B_1^{O_{\text{ind}}, O_{\text{corr}}}(pk)$ first simulates $A_1^{O_{\text{ind}}, O_{\text{corr}}}(pk)$, i.e., $B$ registers and corrupts the same identities as $A$, while keeping lists $U_1, V_1$ of the identities it declares and corrupts by calling $O_{\text{reg}}$ and $O_{\text{corr}}$. $A$ returns some state $\text{state}_1$. $B_1$ then corrupts each user $A_1$ that has registered, i.e., $B_1$ calls $O_{\text{corr}}(id)$ for each $A_1$ that has declared, and stores each $id$'s credential in a list $C_1$.

- $B_2^{O_{\text{state}}, O_{\text{corr}}}(\text{state}_1, pk)$ maintains lists $V_0, V_1, B_2$, initially empty, which will be used to simulate the lists with the same name in $\text{Exp}_{B}^\text{priv-careful}$. $B_2$ will also use lists $hL, H$, and a list $L_{id}$ for each $id \in U_1$, all of them initially empty.

$B_1$ first simulates $A_2^{O_{\text{corr}}}(\text{state}_1, pk)$. For each call to $O_{\text{reg}}(id, v_0, v_1)$, provided $id \in U_1 \setminus V_1$, $B$ checks whether $id$ is already present in $V_0, V_1$. Provided it is not, $B$ then retrieves $id$'s credential $\text{cred}_id$ from $C_1$, and calls $O_{\text{reg}}(id, \text{cred}_id, v_0, v_1)$. $B$ obtains a ballot $b$. $B$ then appends $(id, v_0)$ to $V_0$, $(id, v_1)$ to $V_1$, $b$ to $hL$, and $b$ to $L_{id}$. Finally $B$ returns $b$ to $A$. $A_2$ eventually returns a board $BB$. $B$ write $BB\mid hL$ (resp. $BB \setminus hL$) the sublist of $BB$ (in the same order) of ballots that do not occur (resp. do occur) in $hL$.

- $B_2$ calls $O_{\text{state}}^{\text{ind}}(BB\mid hL)$, and obtains a list $L$ of pairs of credentials and votes. $B_2$ then computes the list $L'$ of the first vote for each credential in $L$. Note that, by construction, no ballot in $BB\mid hL$ has been generated by a call to $O_{\text{state}}$, which means that $O_{\text{state}}^{\text{ind}}$ accepts to open all ballots in $BB\mid hL$.

- $B_2$ then simulates $A_3^{O_{\text{corr}} \cap \text{happ}}$. For each call to $O_{\text{happ}}(id)$, provided $id \in U_1 \setminus V_1$, following the specification of the voter verification for Civitas, $B$ 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, $B$ appends $id$ to $hL$. In any case, $B$ then resumes the execution of $A_1$.

- $B_2$ then computes $\rho(V_0), \rho(V_1)$, and checks that they are equal, and that every $id$ occurring in $V_0, V_1$ is also an element of $H$. If so, $B_2$ computes $r = \rho(V_0) \cdot \rho(L')$. Otherwise, $B_2$ sets $r = \bot$.

- $B_2$ then calls $A_4$ on $r$, and obtains a bit $\beta'$. $B_2$ returns $\beta'$.

Note that the lists $BB, V_0, V_1$ in $B$ are equal to the lists of the same name in $\text{Exp}_{A}^\text{priv-careful}$. Then, by construction, $A_2$ is always accurately simulated by $B_2$, i.e., it is called on the same inputs, and provided with the same oracles as what would happen in $\text{Exp}_{A}^\text{priv-careful}$. Given the specification of VerifVoter for Civitas, $A_3$ is also accurately simulated by $B_2$.

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

- $C_1$ simulates $A_1$, similar to $B_1$. Unlike $B_1$, $C_1$ stops here, and does not corrupt all identities.

- $C_2$ first draws at random a bit $\beta''$. Similarly to $B_2$, $C_2$ then simulates $A_2$ and uses lists $hL, V$. For each call to $O_{\text{happ}}^{\text{ind}}(id, v_0, v_1)$, provided $id \in U_1 \setminus V_1$ does not already occur in $V$, $C$ calls $O_{\text{ind}}(id, v_0, v_1)$. $C$ obtains a ballot $b$ and appends $(id, v_0, v_1)$ to $V$, and $b$ to $hL$. Finally $C$ returns $b$.

$A_2$ eventually returns a board $BB$. $C$ then draws at random a ballot of $BB$ and returns it.

Note that the lists $BB, hL, V$ are the same for $C$ in $\text{Exp}^\text{NM}$ at the point $C_2$ returns, and for $B$ in $\text{Exp}_{B}^\text{ind}$ at the point the simulated $A_2$ returns. Similarly $V$ in $C$ is the same as $V_{\text{corr}}$ in $B$. Let us also notice that $hL$ in $C$ is equal to the list $L$ in the game $\text{Exp}^\text{NM}$.

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

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.

\end{proof}
For any \( \beta \), assume \( BB \setminus hL \) contains a ballot \( b \) such that there exists a honest (\( * \), cred) \( \in U \setminus CU \) such that open\(_{\text{cred}}(b, sk, U) = \text{cred} \) (note that \( U, CU \) in \( \text{Exp}_{NM} \) are equal to \( U_1, CU_1 \) in \( B \)). Thus, provided \( C \) correctly guesses \( \beta'' = \beta \), and chooses this \( b \) from \( BB \), the condition \( b \not\in L \land \exists (id, \text{cred}) \in U \setminus CU \). open\(_{\text{cred}}(b, sk, U) = \text{cred} \) in \( \text{Exp}_{NM} \) holds. Indeed, since \( b \not\in hL \), \( b \not\in L \). Thus \( \text{Exp}_{NM} = 1 \).

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

For any \( \beta \), \( \text{Exp}_{B}^{\text{ind}, \beta}(\lambda) = 1 \) if and only if \( B_2 \) returns 1 in this game.

Assume \( BB \setminus hL \) does not contain any ballot associated with a honest credential (\( i.e. \) a credential in \( U_1 \setminus CU_1 \)), which, as we have established, holds except with probability at most \( 2q(\lambda)P\left[ Exp_{NM} = 1 \right] \).

Let us then show that \( \mathcal{A}_4 \) is accurately simulated by \( B_2 \), \( i.e. \) it is provided by \( B_2 \) with the same input as when it is called in \( \text{Exp}_{\mathcal{A}}^{\text{priv}, \beta} \), that is the actual tally of the board \( BB \) (which \( \mathcal{A}_4 \) returned) if \( \rho(V_0) = \rho(V_1) \) and if \( \forall \text{ Vid.} \ (id, *) \in V_0, V_1 \Rightarrow id \in H \) and \( \bot \) otherwise.

It is clear from the definition of \( B \) that when either the equality condition \( \rho(V_0) = \rho(V_1) \) or the voter verification condition \( \forall \text{ Vid.} \ (id, *) \in V_0, V_1 \Rightarrow id \in H \) do not hold, \( \mathcal{A}_4 \) is indeed given \( \bot \) 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 \( O_{L}^{I} \). We will thus see \( BB \) as an interleaving \( (BB \setminus hL) \cup (BB \cap hL) \) of the ballots in \( BB \setminus hL \) and \( BB \cap hL \) (keeping the same order within each of these two lists).

By assumption, \( BB \setminus hL \) contains no ballot for honest credentials, whereas by construction \( hL \) (and thus \( BB \cap hL \)) only contains ballots for honest credentials. In addition, by construction, \( hL \) contains at most one ballot for each credential. Thus \( BB \cap hL \) 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, \( i.e. \) \( \forall \text{ Vid.} \ (id, *) \in V_0, V_1 \Rightarrow id \in H \) holds. By construction, each \( b \in hL \) was added when simulating a call to \( O_{\text{V}}^{\text{id}}(id, v_0, v_1) \) for some honest \( id \in U_1 \setminus CU_1 \) and some \( v_0, v_1 \). Therefore \( (id, v_0) \in V_0, (id, v_1) \in V_1, \) and \( k_{id} = [b] \). Hence, \( id \in H \). By definition of \( B \) (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 \( BB \setminus hL \) and \( BB \cap hL \) (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

\[
\text{Tally}(BB, sk, U_1) = \text{Tally}(BB \cap hL, sk, U_1) + \text{Tally}(BB \setminus hL, sk, U_1).
\]

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

\[
\text{Tally}(BB, sk, U_1) = \text{Tally}(hL, sk, U_1) + \text{Tally}(BB \setminus hL, sk, U_1).
\]

By construction, \( hL \) contains ballots for either the votes in \( V_0 \) or those in \( V_1 \), depending on \( \beta \). Since at that point \( \rho(V_0) = \rho(V_1) \), we have \( \rho(V_0) = \text{Tally}(hL, sk, U_1) \). In addition, the oracle \( O_{L}^{I} \) returns the list \( L \) of the credentials and votes of each ballot in \( BB \setminus hL \). Since \( L' \) is the list of the first vote for each credential in \( L \), we thus have \( \text{Tally}(BB \setminus hL, sk, U_1) = \rho(L') \). Therefore, \( \rho(V_0) = \rho(L') \), which is the result computed by \( B_2 \), is indeed \( \text{Tally}(BB, sk, U_1) \), which concludes the proof that \( \mathcal{A}_4 \) is accurately simulated by \( B_2 \).

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

Thus

\[
P\left[ \text{Exp}_{B}^{\text{ind}, \beta}(\lambda) = 1 \right] = P\left[ \text{Exp}_{\mathcal{A}}^{\text{priv, careful}, \beta}(\lambda) \neq 1 \right] \leq 2q(\lambda)P\left[ \text{Exp}_{C}^{\text{NM}} = 1 \right].
\]

We thus have:

\[
P\left[ \text{Exp}_{\mathcal{A}}^{\text{priv, careful}, 0}(\lambda) = 1 \right] = P\left[ \text{Exp}_{\mathcal{A}}^{\text{priv, careful}, 1}(\lambda) = 1 \right]
= P\left[ \text{Exp}_{\mathcal{A}}^{\text{priv, careful}, 0}(\lambda) = 1 \right] - P\left[ \text{Exp}_{\mathcal{A}}^{\text{priv, careful}, 1}(\lambda) = 1 \right] + P\left[ \text{Exp}_{B}^{\text{ind}, 0}(\lambda) = 1 \right] - P\left[ \text{Exp}_{B}^{\text{ind}, 1}(\lambda) = 1 \right]
\leq P\left[ \text{Exp}_{\mathcal{A}}^{\text{priv, careful}, 0}(\lambda) = 1 \right] - P\left[ \text{Exp}_{\mathcal{A}}^{\text{priv, careful}, 1}(\lambda) = 1 \right] + P\left[ \text{Exp}_{\mathcal{A}}^{\text{ind}, 0}(\lambda) = 1 \right] - P\left[ \text{Exp}_{\mathcal{A}}^{\text{ind}, 1}(\lambda) = 1 \right]
\leq P\left[ \text{Exp}_{B}^{\text{ind}, 0}(\lambda) = 1 \right] - P\left[ \text{Exp}_{B}^{\text{ind}, 1}(\lambda) = 1 \right] + 4q(\lambda)P\left[ \text{Exp}_{C}^{\text{NM}} = 1 \right]
\]
Therefore, if $A$ breaks privacy with careful voters, i.e. if $\Pr[\exp_{A}^{\text{priv-careful},0}(\lambda) = 1] - \Pr[\exp_{A}^{\text{priv-careful},1}(\lambda) = 1]$ is not negligible, then $B$ breaks $\exp^{\text{ind}}$, or $C$ breaks $\exp^{\text{NM}}$.

C.2.3 Belenios is private for $\exp^{\text{priv-careful}}$

Theorem C.3. Assuming no adversary wins $\exp^{\text{ind}}$ nor $\exp^{\text{NM}}$ with non-negligible probability, Belenios is private for $\exp^{\text{priv-careful}}$.

Proof. Let $A = A_1, A_2, A_3, A_4$ be an adversary that wins $\exp^{\text{priv-careful}}$. We consider an adversary $B = B_1, B_2, B_3$ that plays $\exp^{\text{ind}}$.

- $B_1^{O_{\text{reg}}, O_{\text{corr}}}(pk)$ first simulates $A_1^{O_{\text{reg}}, O_{\text{corr}}}(pk)$, i.e. $B$ registers and corrupts the same identities as $A$, while keeping lists $U_1, C_2$ of the identities it declares and corrupts by calling $O_{\text{reg}}$ and $O_{\text{corr}}$. $A$ returns some state $\beta'$. $B_1$ then corrupts each user $A$ has registered, i.e. $B_1$ calls $O_{\text{corr}}(id)$ for each $id \in A_1$, lists each id's credential in a list $C_{U_2}$.

- $B_2^{\beta', \beta''}(\text{state}_1, pk)$ maintains lists $V_0, V_1, B_3$, initially empty, which will be used to simulate the lists with the same name in $\exp^{\text{priv-careful}}$. $B_2$ will also use lists $hL, hR, and a list L_{id}$ for each id $\in U_1$, all of them initially empty.

- $B_2$ first simulates $A_2^{O_{\text{corr}}}(\text{state}_1, pk)$. For each call to $O_{\text{corr}}^{\text{vote}}(id, v_0, v_1)$, provided id $\in U_1 \setminus C_{U_2}$, $B$ retrieves id's credential $\text{cred}_id$ from $C_{U_2}$, and calls $O_{\beta'}^{\beta''}(id, \text{cred}_id, v_0, v_1)$. $B$ 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 $B_3$, and $b$ to $L_{id}$. Finally $B$ returns $b$ to $A$.

- $A_2$ eventually returns a board $B_3$. We write $B_3|hL$ (resp. $B_3 \cap hL$) the sublist of $B_3$ (in the same order) of ballots that do not occur (resp. do occur) in $hL$.

- $B_2$ calls $O_{\beta'}^{\beta''}(B_3|hL)$, and obtains a list $L$ of pairs of credentials and votes. $B_2$ then computes the list $L'$ of the last vote for each credential in $L$. We will see later that under the right assumptions, $O_{\beta'}^{\beta''}$ accepts to open all (valid) ballots in $B_3|hL$.

- $B_2$ then simulates $A_3^{O_{\text{happy}}}(\text{id}, hL)$. For each call to $O_{\text{happy}}^{\text{state}}(\text{id})$, provided id $\in U_1 \setminus C_{U_1}$, following the specification of the voter verification for Belenios, $B$ retrieves id's credential $\text{cred}_id$ from $C_{U_2}$, and checks whether the last ballot $b$ in $L_{id}$ is the last ballot associated to $\text{cred}_id$ (i.e. signed by $\text{cred}_id$) in $B_3$. If so, $B$ appends id to $hR$. In any case, $B$ then resumes the execution of $A_3$.

- $B_2$ then computes $\rho(V_0), \rho(V_1)$, and checks that they are equal, and that every id occurring in $V_0, V_1$ is also an element of $hR$. If so, $B_2$ computes $r = \rho(V_0) * \rho(L')$. Otherwise, $B_2$ sets $r = 1$.

- $B_2$ then calls $A_4$ on $r$, and obtains a bit $\beta'$. $B_2$ returns $\beta'$.

Note that the lists $B_3, V_0, V_1, hL$ in $B$ are equal to the lists of the same name in $\exp^{\text{priv-careful}, \beta'}$. Then, by construction, $A_2$ is always accurately simulated by $B_2$, i.e. it is called on the same inputs, and provided with the same oracles as what would happen in $\exp^{\text{priv}, \beta'}$. Given the specification of VerifVoter for Belenios, $A_3$ is also accurately simulated by $B_2$.

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

- $C_1$ simulates $A_1$, similarly to $B_1$. However, unlike $B_1$, it stops there, and does not corrupt all identities.

- $C_2$ then draws at random a bit $\beta''$. Similarly to $B_2, C_2$ then simulates $A_2$ and uses lists $hL, hR$. For each call to $O_{\text{corr}}^{\text{vote}}(id, v_0, v_1)$, provided id $\in U_1 \setminus C_{U_1}$, $C$ calls $O_{\beta''}(id, v_{\beta''})$. $C$ obtains a ballot $b$, removes all elements of the form $(id, *)$ from $V$, and appends $(id, v_{\beta''})$ to $V$, and $b$ to $B_3$. Finally $C$ returns $b$.

- $A_2$ eventually returns a board $B_3$. $C$ then draws at random a ballot of $B_3$ and returns it.

Note that the lists $B_3, V_0, V_1, hL$ in $B$ are equal to the same for $C$ in $\exp^{\text{NM}}$ at the point $C_2$ returns, and for $B$ in $\exp^{\text{ind}}$ at the point the simulated $A_2$ returns. Similarly $V$ in $C$ is the same as $V_{\beta''}$ in $B$. Let us also notice that $hL$ in $C$ is equal to the list $L$ in the game $\exp^{\text{NM}}$.

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

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 $\beta$, assume $BB|\text{hL}$ contains a ballot $b$ such that there exists honest (id, $\text{cred}_id$) $\in U \setminus C_{U_1}$, and a vote $v$, such that open($b, sk$) = $\text{cred}_id$ (note that $U \setminus C_{U_1}$ in $\exp^{\text{NM}}$ are equal to $U_1 \setminus C_{U_1}$ in $B$). Thus, provided $C$ correctly guesses $\beta'' = \beta$, and chooses this $b$ from $BB$, the condition $b \notin L \land \exists (id, \text{cred}_id) \in U \setminus C_{U_1}$ open($b, sk$) = $\text{cred}_id$, $*$ holds in $\exp^{\text{NM}}$. Indeed, since $b \notin hL$, $b \notin L$, and thus $\exp^{\text{NM}} = 1$.

Therefore, such a ballot $b$ exists with probability at most $2|BB| \Pr[\exp^{\text{NM}} = 1]$, which is smaller than $2q(\lambda) \Pr[\exp^{\text{NM}} = 1]$, since $|BB| \leq q(\lambda)$.

For any $\beta$, $\exp^{\text{ind}, \beta}(\lambda) = 1$ if and only if $B_2$ returns 1 in this game.

Assume $BB|\text{hL}$ does not contain any ballot associated with a honest credential (i.e. a credential in $U_1 \setminus C_{U_1}$), which, as we have established, holds except with probability at most $2q(\lambda) \Pr[\exp^{\text{NM}} = 1]$. 32
Note that this assumption notably implies that no ballot in $\mathbb{B}\backslash hL$ has the same ciphertext as a ballot generated by a call to $O^1_d$, that is, for all $b \in \mathbb{B}\backslash hL$, extract($b$) $\notin L$. Indeed, the ciphertext part of the ballot, for Belenos, is signed with the credential. Hence any $c \in L$ is (by definition of $O^1_d$) signed with a honest credential, which, by the assumption, is not the case of any of the ballots in $\mathbb{B}\backslash hL$. Therefore, as stated earlier, $O^1_d$ accepts to open all (valid) ballots in $\mathbb{B}\backslash hL$.

Let us then show that $A_4$ is accurately simulated by $B_2$, i.e. it is provided by $B_2$ with the same input as when it is called in $\text{Exp}_{\mathcal{A}}^{\text{priv},\beta}$, that is the actually tally of the board $\mathbb{B}$ (which $A_2$ returned) if $\rho(\mathcal{V}_0) = \rho(\mathcal{V}_1)$ and if $\forall \text{id. } (id, *) \in V_0, V_1 \Rightarrow id \in H$; and $\perp$ otherwise.

It is clear from the definition of $B$ that when either the equality condition $\rho(\mathcal{V}_0) = \rho(\mathcal{V}_1)$ or the voter verification condition $\forall \text{id. } (id, *) \in V_0, V_1 \Rightarrow id \in H$ do not hold, $A_4$ is indeed given $\perp$ as an argument.

Let us now study the case where both these conditions are met. Among the ballots in $\mathbb{B}$, some are also in the list $hL$ of ballots created by the oracle $O^1_d$. We will thus see $\mathbb{B}$ as an interleaving of the ballots in $\mathbb{B}\backslash hL$ and $\mathbb{B}\backslash hL$ (keeping the same order within each of these two lists).

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

Moreover, we assumed the voter verifications succeeded, i.e. $\forall \text{id. } (id, *) \in V_0, V_1 \Rightarrow id \in H$ holds. According to the specification of VerifVoter for Belenos, this means that for all $id$ occurring in $V_0, V_1$, the last ballot $b$ in $L_{id}$, i.e. the last ballot produced by $O^{\text{vote},c}_{\text{vote}}((id, *))$, is also the last ballot signed by $\text{cred}\text{,id}$ in $\mathbb{B}$. ($\text{cred}\text{,id}$ being the credential associated with $id$ in $CU_2$). Since $id$ (and thus $\text{cred}\text{,id}$) is honest by definition of $V_0, V_1$, $b$ is actually the last ballot signed by $\text{cred}\text{,id}$ in $\mathbb{B}\backslash hL$.

Consider a credential $\text{cred}$ such that $\mathbb{B}\backslash hL$ contains at least one ballot signed by $\text{cred}$. This ballot can only have been added to $\mathbb{B}$ by a call to $O^{\text{vote},c}_{\text{vote}}$ 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 $V_0, V_1$.

Hence, the list $hL'$ of the last ballots signed by each credential in $\mathbb{B}\backslash hL$ is exactly the list of the last ballots produced by $O^{\text{vote}}_{\text{vote}}$ for each $id$ in $V_0, V_1$. By construction, this list contains ballots for either the votes in $V_0$ or those in $V_1$, depending on $\beta$. Since at that point $\rho(\mathcal{V}_0) = \rho(\mathcal{V}_1)$, we have $\rho(\mathcal{V}_0) = \text{Tally}(hL', sk, U_1)$. In addition, the revote policy specified for Belenos is to count only the last ballot signed by each credential. Therefore, $\text{Tally}(\mathbb{B}\backslash hL, sk, U_1) = \text{Tally}(hL', sk, U_1) = \rho(\mathcal{V}_0)$.

Besides, the oracle $O^1_d$ returns the list $L$ of the credentials and votes of each ballot in $\mathbb{B}\backslash hL$. Since $L'$ is the list of the last votes associated with each credential in $L$, we thus have $\text{Tally}(\mathbb{B}\backslash hL, sk, U_1) = \rho(L')$.

Since $\mathbb{B}$ can be separated into the lists $\mathbb{B}\backslash hL$ and $\mathbb{B}\backslash hL$ (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 $\mathbb{B}$, we have

$$\text{Tally}(\mathbb{B}, sk, U_1) = \text{Tally}(\mathbb{B}\backslash hL, sk, U_1) \ast \text{Tally}(\mathbb{B}\backslash hL, sk, U_1) = \rho(\mathcal{V}_0) \ast \rho(L').$$

Therefore, $\rho(\mathcal{V}_0) \ast \rho(L')$, which is the result computed by $B_2$, is indeed $\text{Tally}(\mathbb{B}, sk, U_1)$, which concludes the proof that $A_4$ is accurately simulated by $B_2$.

Hence, unless $\mathbb{B}\backslash hL$ contains a ballot associated with a honest credential, $\text{Exp}^{\text{ind},\beta}_{\mathcal{B}}(\lambda) = 1$ if and only if the accurately simulated $A_4$ returns $1$, i.e. if and only if $\text{Exp}^{\text{priv}-\text{careful},\beta}_{\mathcal{A}}(\lambda) = 1$.

Thus

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

We thus have:

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

$$\leq \left| P \left[ \text{Exp}^{\text{priv}-\text{careful},0}_{\mathcal{A}}(\lambda) = 1 \right] - P \left[ \text{Exp}^{\text{ind},0}_{\mathcal{B}}(\lambda) = 1 \right] \right| + \left| P \left[ \text{Exp}^{\text{ind},1}_{\mathcal{B}}(\lambda) = 1 \right] - P \left[ \text{Exp}^{\text{priv}-\text{careful},1}_{\mathcal{A}}(\lambda) = 1 \right] \right|$$

$$\leq \left| P \left[ \text{Exp}^{\text{ind},0}_{\mathcal{B}}(\lambda) = 1 \right] - P \left[ \text{Exp}^{\text{ind},1}_{\mathcal{B}}(\lambda) = 1 \right] \right| \leq 4q(\lambda) P \left[ \text{Exp}^{\text{NM}}_{\mathcal{C}} = 1 \right].$$

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

$\square$
C.2.4 Helios is private for \(\text{Exp}^{\text{priv}}\).

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

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

- \(\mathcal{B}_1\) first simulates \(\mathcal{A}_1\) with \(\text{Exp}^{\text{priv}}\), i.e., \(\mathcal{B}_1\) registers and corrupts the same identities as \(\mathcal{A}_1\), while keeping lists \(U_1, U_2\) of the identities it declares and corrupts by calling \(\text{Exp}^{\text{reg}}\) and \(\text{Exp}^{\text{corr}}\). \(\mathcal{B}_1\) returns some state \(\mathcal{A}_1\), \(\mathcal{B}_1\) then corrupts each user \(\mathcal{A}_1\) that has registered, i.e., \(\mathcal{B}_1\) calls \(\text{Exp}^{\text{corr}}(id)\) for each id \(\mathcal{A}_1\) has declared, and stores each id’s credential in a list \(\mathcal{B}_2\).

- \(\mathcal{B}_2\) first simulates \(\mathcal{A}_2\) with \(\text{Exp}^{\text{priv}}\), i.e., \(\mathcal{B}_2\) maintains lists \(V_0, V_1, B\), initially empty, which will be used to simulate the lists with the same name in \(\text{Exp}^{\text{priv}}\). \(\mathcal{B}_2\) will also use a list \(L\), initially empty.

\(\mathcal{B}_2\) then checks whether two ballots in \(B\) have the same ciphertext, i.e., if there exist two ballots \(b, b'\) in \(B\) such that \(\text{extract}(b) = \text{extract}(b')\). If so, following the specification of Helios, \(\mathcal{B}_2\) lets \(r = 1\). Otherwise, we write \(B \cap L\) (resp. \(B \cap hL\)) the sublist of \(B\) of ballots that do not occur (resp. do occur) in \(L\). Note that by construction all ballots in \(B\) occur in \(B\).

- \(\mathcal{B}_2\) calls \(\text{Exp}^{\text{priv}}(\mathcal{B}_2)\) and obtains a list \(L\) of votes. Note that, if that point is reached, no two ballots in \(B\) have the same ciphertext. Hence, no ballot in \(B \cap L\) has the same ciphertext as a ballot in \(L\), which is the list of all ballots produced by \(\text{Exp}^{\text{priv}}\). Thus \(\text{Exp}^{\text{priv}}\) accepts to open all ballots in \(B \cap L\). \(\mathcal{B}_2\) computes (using \(B\) and \(L\)) the list \(L'\) of the last vote from each id.

- \(\mathcal{B}_2\) computes \(r = \rho(V_0) \cdot \rho(L')\).

In any case, i.e., even if \(r = 1\), \(\mathcal{B}_2\) calls \(\mathcal{A}_3\) on \(r\), and obtains a bit \(r'\). \(\mathcal{B}_2\) returns \(r'\).

Note that the lists \(B, V_0, V_1\) in \(\mathcal{B}\) are equal to the lists of the same name in \(\text{Exp}^{\text{priv}}_{\mathcal{A}_3}\). 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 \(\text{Exp}^{\text{priv}}_{\mathcal{A}_3}\).

We will now prove that if \(\mathcal{A}_3\) breaks privacy, then \(\mathcal{B}_2\) wins \(\text{Exp}^{\text{ind}}\).

For any \(\beta\), \(\text{Exp}^{\text{ind},\beta}_{\mathcal{B}_2}(\lambda) = 1\) 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., if it is simulated by \(\mathcal{B}_2\) only when it is called in \(\text{Exp}^{\text{priv},\beta}_{\mathcal{A}_3}\) (that is, when \(\rho(V_0) = \rho(V_1)\)), and it is provided with the actually tally of the board \(B\) (which \(\mathcal{A}_3\) interacted with).

It is clear from the construction of \(\mathcal{B}\) that if \(\rho(V_0) \neq \rho(V_1)\), \(\mathcal{A}_3\) is not simulated. Hence, it is simulated by \(\mathcal{B}_2\) only when it is called in \(\text{Exp}^{\text{priv},\beta}_{\mathcal{A}_3}\).

When \(\mathcal{A}_3\) is called:

- either \(B\) contains duplicate ciphertexts, and \(\mathcal{A}_3\) is called on \(r = 1\), which corresponds to what happens in \(\text{Exp}^{\text{priv},\beta}_{\mathcal{A}_3}\); or
- \(B\) 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 \(B\) into two boards \(BB_h \cup BB_d\), 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 \(BB \cap hL\).

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

\[
\text{Tally}(BB, sk, U_1) = \text{Tally}(BB_h, sk, U_1) + \text{Tally}(BB_d, sk, U_1) = \text{Tally}(hL, sk, U_1) + \text{Tally}(BB \setminus hL, sk, U_1).
\]

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 \(\beta\). Since at that point \(\rho(V_0) = \rho(V_1)\), we thus have \(\rho(V_0) = \text{Tally}(hL, sk, U_1)\).

In addition, the oracle \(\text{Exp}^{\text{priv}}\) returns the list \(L\) of the votes of each ballot in \(BB \setminus hL\) to each id, we thus have \(\text{Tally}(BB \setminus hL, sk, U_1) = \rho(L')\).

Therefore, we indeed have

\[
\text{Tally}(BB, sk, U_1) = \text{Tally}(hL, sk, U_1) + \text{Tally}(BB \setminus hL, sk, U_1) = \rho(V_0) + \rho(L'),
\]
which is the result computed by \( B_2 \). This concludes the proof that \( A_3 \) is accurately simulated by \( B_2 \).

Hence \( \text{Exp}^{\text{ind},\beta}_B(\lambda) = 1 \) if and only if the accurately simulated \( A_3 \) returns 1, i.e., if and only if \( \text{Exp}^{\text{priv},\beta}_A(\lambda) = 1 \).

Thus

\[
P\left[ \text{Exp}^{\text{ind},\beta}_B(\lambda) = 1 \right] = P\left[ \text{Exp}^{\text{priv},\beta}_A(\lambda) \neq 1 \right].
\]

We thus have:

\[
P\left[ \text{Exp}^{\text{priv},0}_A(\lambda) = 1 \right] - P\left[ \text{Exp}^{\text{priv},1}_A(\lambda) = 1 \right] = P\left[ \text{Exp}^{\text{ind},0}_B(\lambda) = 1 \right] - P\left[ \text{Exp}^{\text{ind},1}_B(\lambda) = 1 \right].
\]

Therefore, if \( A \) breaks privacy, i.e., if \( P\left[ \text{Exp}^{\text{priv},0}_A(\lambda) = 1 \right] - P\left[ \text{Exp}^{\text{priv},1}_A(\lambda) = 1 \right] \) is not negligible, then \( B \) breaks \( \text{Exp}^{\text{ind}} \).

\[\square\]

C.2.5 Simple is private for \( \text{Exp}^{\text{priv}} \).

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

**Proof.** Let \( A = A_1, A_2, A_3 \) be an adversary that wins \( \text{Exp}^{\text{priv}} \). We consider an adversary \( B = B_1, B_2 \) that plays \( \text{Exp}^{\text{ind}} \):

- \( B_1^{\mathcal{O}_{\text{reg}}, \mathcal{O}_{\text{corr}}}(pk) \) first simulates \( A^{\mathcal{O}_{\text{reg}}, \mathcal{O}_{\text{corr}}}(pk) \), i.e., \( B \) registers and corrupts the same identities as \( A \), while keeping lists \( U_1, \text{CU}_1 \) of the identities it declares and corrupts by calling \( \mathcal{O}_{\text{reg}} \) and \( \mathcal{O}_{\text{corr}} \). \( A \) returns some state \( \lambda \). \( B_1 \) then corrupts each user \( A_1 \) has registered, i.e., \( B_1 \) calls \( \mathcal{O}_{\text{corr}}(id) \) for each \( id \) \( A_1 \) has declared, and stores each \( id \)’s credential in a list \( \text{CU}_2 \).

- \( B_2^{\mathcal{O}^i_{\text{rot}}, \mathcal{O}^i_{\text{cast}}}(\text{state}_{\lambda}, pk) \) maintains lists \( V_0, V_1, \text{BB} \), initially empty, which will be used to simulate the lists with the same name in \( \text{Exp}^{\text{priv}} \). \( B_2 \) will also use a list \( h_L \), initially empty.

\( B_2 \) first simulates \( A^{\mathcal{O}^i_{\text{rot}}, \mathcal{O}^i_{\text{cast}}}(\text{state}_{\lambda}, pk) \):

- for each call to \( O^i_{\text{rot}}(id, v_0, v_1) \), provided \( id \in U_1 \setminus \text{CU}_1 \) does not already occur in \( V_0 \), \( V_1 \), \( B \) retrieves \( id \)’s credential \( \text{cred}_{id} \) from \( \text{CU}_2 \), and calls \( O^i_{\text{cast}}(id, \text{cred}_{id}, v_0, v_1) \). \( B \) obtains a ballot \( b \). \( B \) then appends \( (id, v_0) \) to \( V_0 \), \( (id, v_1) \) to \( V_1 \), \( b \) to \( h_L \), and to \( \text{BB} \). Finally \( B \) returns \( b \) to \( A \).

- for each call to \( \mathcal{O}_{\text{cast}}(id, b) \), provided \( id \in \text{CU}_1 \) and \( \text{Valid}(id, b, \text{BB}, pk) \) (which implies, for Simple, that \( b \) does not already occur in \( \text{BB} \)). \( B \) appends \( b \) to \( \text{BB} \). Finally \( B \) returns \( b \) to \( A \).

- \( B_2 \) then computes \( \rho(V_0), \rho(V_1) \), and checks they are equal. If not, \( B_2 \) blocks.

- \( B \) then checks whether both ballots in \( \text{BB} \) are equal. If so, following the specification of Simple, \( B \) lets \( r = \bot \). Otherwise, we write \( \text{BB} \setminus \text{hL} \) (resp. \( \text{BB} \cap \text{hL} \)) the sublist of \( \text{BB} \) of ballots that do not occur (resp. do occur) in \( \text{hL} \). Note that by construction all ballots in \( \text{hL} \) occur in \( \text{BB} \) in that order.

- \( B_2 \) calls \( O^i_{\text{BB}}(\text{BB} \setminus \text{hL}) \), and obtains a list \( L \) of votes. Note that, if that point is reached, no two ballots in \( \text{BB} \) are equal. Hence, no ballot in \( \text{BB} \setminus \text{hL} \) is equal to a ballot in \( \text{hL} \), which is the list of all ballots produced by \( O^i_{\text{BB}} \). Thus \( O^i_{\text{BB}} \) accepts to open all ballots in \( \text{BB} \setminus \text{hL} \).

- \( B_2 \) computes \( r = \rho(V_0) \ast \rho(L) \).

- In any case, i.e., even if \( r = \bot \), \( B_2 \) calls \( A_3 \) on \( r \), and obtains a bit \( \beta' \). \( B_2 \) returns \( \beta' \).

Note that the lists \( \text{BB}, V_0, V_1 \) in \( B \) are equal to the lists of the same name in \( \text{Exp}^{\text{priv},\beta}_A \). Then, by construction, \( A_2 \) is always accurately simulated by \( B_2 \), i.e., it is called on the same inputs, shown the same board, and provided with the same oracles as what would happen in \( \text{Exp}^{\text{priv},\beta}_A \).

We will now prove that if \( A \) breaks privacy, then \( B \) wins \( \text{Exp}^{\text{ind}} \).

For any \( \beta, \text{Exp}^{\text{ind},\beta}_B(\lambda) = 1 \) if and only if \( B_2 \) returns 1 in this game.

Let us then show that \( A_3 \) is accurately simulated by \( B_2 \), i.e., it is simulated by \( B_2 \) only when it is called in \( \text{Exp}^{\text{priv},\beta}_A \) (that is, when \( \rho(V_0) = \rho(V_1) \)), and it is provided with the actually tally of the board \( \text{BB} \) (which \( A_2 \) interacted with).

It is clear from the construction of \( B \) that if \( \rho(V_0) \neq \rho(V_1) \), \( A_3 \) is not simulated. Hence, it is simulated by \( B_2 \) only when it is called in \( \text{Exp}^{\text{priv},\beta}_A \).

When \( A_3 \) is called:

- either \( \text{BB} \) contains duplicate ballots, and \( A_3 \) is called on \( r = \bot \), which corresponds to what happens in \( \text{Exp}^{\text{priv},\beta}_A \);
- or \( \text{BB} \) 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 \( \mathbb{B} \) into two lists \( (\mathbb{B} \cap hL) \cup (\mathbb{B} \setminus hL) \).

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

By assumption, \( \mathbb{B} \) contains no duplicate ballots. Hence, the lists \( hL \) and \( \mathbb{B} \setminus hL \) do not have any ballot in common, and according to the specification of Simple we have

\[
\text{Tally}(\mathbb{B}, sk, U_1) = \text{Tally}(hL, sk, U_1) \ast \text{Tally}(\mathbb{B} \setminus hL, sk, U_1).
\]

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 \( \beta \). Since at that point \( \rho(V_0) = \rho(V_1) \), we thus have \( \rho(V_0) = \text{Tally}(hL, sk, U_1) \).

In addition, the oracle \( O_1 \) returns the list \( L \) of the votes of each ballot in \( \mathbb{B} \setminus hL \). We thus have \( \text{Tally}(\mathbb{B} \setminus hL, sk, U_1) = \rho(L) \).

Therefore, we indeed have

\[
\text{Tally}(\mathbb{B}, sk, U_1) = \text{Tally}(hL, sk, U_1) \ast \text{Tally}(\mathbb{B} \setminus hL, sk, U_1) = \rho(V_0) \ast \rho(L),
\]

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

Thus

\[
P\left[ \text{Exp}_{\mathcal{B}}^{\text{ind},\beta}(\lambda) = 1 \right] = P\left[ \text{Exp}_{\mathcal{A}}^{\text{ind},\beta}(\lambda) \neq 1 \right].
\]

We thus have:

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

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

\( \square \)