# Is the JCJ voting system really coercion-resistant?

Véronique Cortier, Pierrick Gaudry, Quentin Yang

# Is the JCJ voting system really coercion-resistant?

Véronique Cortier, Pierrick Gaudry, Quentin Yang
Université de Lorraine, Inria, CNRS

***Abstract*—Coercion-resistance is a security property of electronic voting, often considered as a must-have for high-stake elections. The JCJ voting scheme, proposed in 2005 by Juels, Catalano and Jakobsson, is still the reference paradigm when designing a coercion-resistant protocol. We highlight a weakness in JCJ that is also present in all the systems following its general structure. This comes from the procedure that precedes the tally, where the trustees remove the ballots that should not be counted. This phase leaks more information than necessary, leading to potential threats for the coerced voters. Fixing this leads to the notion of *cleansing-hiding*, that we apply to form a variant of JCJ that we call CHide. One reason for the problem not being seen before is the fact that the associated formal definition of coercion-resistance was too weak. We therefore propose a definition that takes into account more behaviors such as revoting or the addition of fake ballots by authorities. We then prove that CHide is coercion-resistant for this definition.**

## I. Introduction

Internet voting allows to take part in an election without being physically present at a polling station. It can be used for many reasons, such as providing people with low mobility or expatriates with a way to vote beside postal voting, or as a necessary alternative during a pandemic. As of today, electronic voting has been used for politically-binding elections in several countries (*e.g.* Australia, Switzerland, Estonia, to name but a few). For such high-stakes contexts, coercion may be an important threat. It occurs when an attacker forces a voter to vote in a specific way, using a threat or a reward. This phenomenon is known to exist in real-world elections, with traditional voting at polling stations. An electronic voting solution which is not designed to tackle coercion could allow the attacker to coerce a larger number of voters, or to gain a more convincing evidence that the coerced voters actually obeyed. Also, since Internet voting is a remote voting process, this introduces new attacks compared to polling station voting. For instance, the coercer can ask the voter to give all the voting materials that they received. The classical verifiability mechanisms will then provide a proof to the coercer that the voter did not cheat.

*The JCJ protocol.* A famous protocol which aims to counter coercion was proposed in 2005 by Juels, Catalano and Jakobsson [22]. They also provide a formalization of the notion, allowing to give security arguments. This is now called the JCJ protocol and remains the reference for the research on coercion-resistance in electronic voting. The key idea of JCJ is that voters can give fake voting material (a fake credential) to the coercer, and pretend that it is genuine. The coercer, who votes with the provided credential, has no way to detect whether the credential is valid or not. In order to guarantee that, during the voting phase, ballots are accepted in the ballot box regardless of their credentials; those which use an invalid or a duplicate one are removed later, during a *cleansing* phase. The output of this cleansing phase is a set of ballots that is tallied in the usual way. The main security feature is that, given a credential and all the publicly available information, the coercer is unable to tell whether the credential is real or fake. At the same time, for the legitimate voters, verifiability is preserved.

For coercion-resistance, the cleansing phase is critical. In JCJ, some information is leaked, namely the number of ballots sent to the public board and to the tallying procedure, before and after the cleansing phase. It is known that the difference $\Delta$ between those two can reveal some information to the coercer. Therefore it is important to ensure some "noise", coming from revotes or dummy ballots, that would mask the action of a voter resisting a coercer. Still, depending on its exact definition, the cleansing phase could leak more than just $\Delta$. For instance, in [32], the authors present a protocol where the coercer can deduce the number of ballots which pretend to be from each voter, and exploit this additional information (which is not available in the original JCJ). To mitigate this leakage, they propose that the authorities add a random number of dummy ballots for each voter, which mitigates the leakage.

*A weakness in JCJ.* We reveal that even in the original JCJ protocol, the cleansing step leaks more than the difference $\Delta$ between the sizes of its input and output. Indeed, first, the ballots with the same credentials (*i.e.* revotes) are handled, keeping only one ballot per credential. Second, the ballots corresponding to invalid credentials are removed. The size of the intermediate ballot box is leaked; moreover, anyone can observe the distribution of the number of revotes. We provide a few examples where this allows to fully break coercion-resistance, even if dummy ballots are used. Admittedly, these extreme scenarios are unlikely to occur in practice. Therefore, we explore more realistic scenarios where major revoting may occur, for example due to a technical incident where voters are encouraged to revote. Another case is when a discredit of a candidate happens during the voting phase, yielding voters to flip their votes. In such cases, coercion resistance is not fully broken but the coercer gains some non-negligible advantage due to the extra leakage of JCJ: they know whether it is more likely that the voter obeyed or not. In order to formally measure the loss of coercion-resistance, we follow the approach of Kuesters *et al* [25] and show that JCJ guarantees a lower level of coercion resistance than a more ideal protocol which only leaks $\Delta$. All the variants and improvements on JCJ

This work was partly supported by the PEPR ANR-22-PECY-0006 grant.

that we know of are also affected by this vulnerability.

*A cleansing-hiding protocol.* We propose a modification of JCJ, called CHide, that is not subject to this weakness. The key modification is the introduction of a *cleansing hiding* procedure, that replaces the original cleansing phase. More precisely, while JCJ uses plaintext equivalence tests (PET) as the main cryptographic tool, CHide relies on more complex MPC building blocks. Instead of PETs that return a bit telling whether two ciphertexts represent the same cleartext, we use a primitive Eq that returns an encrypted version of this bit, in order to hide which ballots are removed and for which reason (being a revote or having a fake credential). Then, a few logical gates primitives (*e.g.* Or, And, *etc.*) must be operated on these encrypted bits to obtain an encrypted validity bit. Finally, we use a Conditional_Set_Zero primitive that conditionally sets the ballots to (fixed) invalid ones, depending on their encrypted validity bit. Thanks to its cleansing hiding procedure, CHide only reveals the minimal information, namely the number $\Delta$ of ballots that have been removed. Of course, each step comes with a zero-knowledge proof that the expected operations have been performed, so that anyone can check that the result of the election is correct.

*A stronger notion of coercion-resistance.* The leakage in JCJ was not noticed before because their definition of coercion-resistance was too weak. A flaw was already noted in [17], which shows that the definition could not be realized. The fix proposed in [17] repairs this but still does not consider the cases where voters may revote and hence misses the situation where JCJ leaks too much information. These definitions also model the addition of ballots with fake credentials in a contrived way in the sense that each fake ballot must be cast by a voter that sacrifices their vote.

We propose a more generic definition of coercion-resistance that accounts for revoting as well as the addition of fake ballots by authorities. We prove that CHide is coercion-resistant according to this definition, and that JCJ is not, thus showing that we indeed capture the weakness of the leakage during the cleansing phase. We also prove that CHide ensures vote privacy (under weaker assumptions than for coercion-resistance) as well as universal verifiability.

*Summary of the contributions.*

- We discover a vulnerability in the JCJ scheme, which shows that it is not perfectly coercion-resistant.
- We formally measure the loss of coercion-resistance in JCJ in two realistic scenarios.
- We propose CHide, a cleansing-hiding variant of JCJ, that is not subject to this problem.
- We propose a new definition of coercion-resistance, which properly takes revoting and dummy ballots into account.
- We prove that CHide is coercion-resistant for this definition, while JCJ is not, thus showing the definition is precise enough to capture the leakage during cleansing.

*Related work.* To provide coercion-resistance, many authors used the core idea of JCJ, that is the fake credential paradigm. Civitas [11] is one of the most notable examples. It is widely considered as an important step towards a practical version of JCJ. Among other things, it introduces the notion of ballot blocks, that mitigates the quadratic cost of the cleansing phase of JCJ. This reduces somehow its coercion-resistance, which can be captured with our approach.

Other attempts were made to improve the efficiency of JCJ. In [32], Spycher *et al.* claim a linear time cleansing, but this comes with a deterioration of the coercion-resistance, as explained above. Later on, the same authors proposed other schemes with a clear trade-off between efficiency and coercion-resistance, thanks to anonymity sets [33]. Other improvements include [1], where Araújo *et al.* propose a way to perform the cleansing phase in linear time, and [10], where Clark and Hengartner introduce the idea of *over-the-shoulder* coercion-resistance. Both schemes would suffer from the same cleansing leakage as JCJ, but it can be observed directly from the public board, when the adversary can still submit ballots. In contrast, the leakage in JCJ can only be observed during the cleansing phase, after the end of the voting phase.

Apart from the fake credential paradigm, a natural approach is to use *deniable revoting*, where the coerced voter first complies with the coercer but revotes later. The ballot cast under coercion is invalidated by the subsequent revote, in a way that the coercer cannot detect. The Estonian voting system [29] completely relies on revoting to mitigate coercion, and examples of recent academic proposals based on revoting are VoteAgain by Lueks *et al.* [28] and the scheme of Locher *et al.* [27]. This approach assumes a weaker adversary which can no longer submit ballots passed a certain point.

All these schemes (including JCJ) do not address the so-called Italian attacks. The latter exist independently of the use of electronic voting and are based on the information given by the tally. They can occur when the ballots are complex enough, so that voters can "sign" them using a specific pattern on the low-stake parts of the answer. When using electronic voting, the typical way to prevent such attacks is *tally hiding*, *i.e.* to decrypt only the winners of the election, without decrypting the individual ballots. The challenge is then to preserve verifiability. Therefore tally-hiding usually relies on homomorphic encryption, or more generally on multiparty computation (MPC) techniques [5], [13], [19], [24]. Designing a tally-hiding scheme is out of scope of this paper but interestingly, CHide could be easily coupled with such tally-hiding schemes, right after the cleansing phase.

Regarding formal definition of coercion-resistance, we already mentioned the recent work of Haines and Smyth [17] that attempts to survey and unify the various definitions of the literature, starting with the one of JCJ where the adversary must distinguish between a real and an ideal game modeling the protocol. Another approach to define coercion-resistance is given by Küsters *et al.* in [25]. The authors define $\delta$-coercion-resistance with two conditions: first, the coerced voter must have a strategy to meet their objective with overwhelming probability (*i.e.* they can actually vote for the desired candidate); second, the adversary cannot distinguish the case when

the voter uses their evasion strategy from the case where the voter forwards all received messages to the adversary, with an advantage greater than $\delta$. Our definition of coercion-resistance can be seen as an instance of [25] where we compare the $\delta$-coercion-resistance of the real protocol with the $\delta'$-coercion-resistance of the ideal one, requiring them to be equal up to negligible difference. While [25] provides a very general and abstract definition, we instead consider a fixed objective for the coerced voter (voting for a given candidate), a fixed evasion strategy (the one considered in JCJ) and a formal framework to model the trust assumptions on how messages are delivered. This part is left unspecified in [25] and needs to be shaped for each protocol.

## II. Unveiling a shortcoming in JCJ

We provide a high level description of the JCJ protocol and show why coercion-resistance is undermined in case of revoting.

### A. Overview of JCJ

The JCJ voting system consists of the following phases.

*Setup.* The Election Trustees jointly generate the election public key $\mathsf{pk}_T$, that is sent to the public board. Then, the Registrars jointly compute one credential $c$ for each voter. Each credential is sent privately to the voter, possibly with designated zero-knowledge proofs to guarantee voters that their credential is valid [21]. The Registrars send to the public board the list $\boldsymbol{R} = \{\mathsf{Enc}(c_1, \mathsf{pk}_T), \ldots, \mathsf{Enc}(c_n, \mathsf{pk}_T)\}$ of encrypted credentials, in some random order.

*Voting.* To cast a ballot, a voter encrypts their vote $\nu$ with the election public key $\mathsf{pk}_T$. They also encrypt their credential $c$ and prove knowledge of $\nu$ and $c$. They prove that the encryptions are valid and linked, yielding a proof $\pi$. The resulting ballot $\mathbf{b} = (\mathsf{Enc}(\nu, \mathsf{pk}_T), \mathsf{Enc}(c, \mathsf{pk}_T), \pi)$ is sent anonymously to the bulletin board. The voting phase is depicted in Figure 1.

*Tallying.* The tally phase consists of four steps.

1. Ballots with duplicated credentials are detected using Plaintext Equivalence Tests [20] (PET). At most one ballot (typically, the last) is kept per credential.
2. The trustees mix the remaining ballots.
3. PETs are used again to remove ballots with invalid credential, *i.e.* whose credential is not encrypted in $\boldsymbol{R}$.
4. Finally, each remaining ballot is decrypted so that the result can be computed.

Each step includes a zero-knowledge proof that the correct operations are performed.

*a) Evading coercion.:* The JCJ voting system provides an *evasion strategy* that a voter under coercion must follow to safely disobey a coercer. If Alice is under coercion, she simply provides her coercer with a random (and fake) credential $c'$. She can later use her real credential $c$ to cast her vote. Ballots containing $c'$ will be removed at Step 3 of the tally phase. Thanks to the mixnet, the coercer is unable to learn that their ballot has been suppressed.

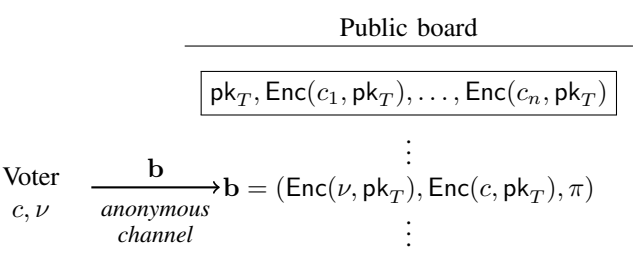


Fig. 1: Voting phase in JCJ.

### B. Leakage in case of revoting

For a verifiable voting system, it seems unavoidable to leak the number of received ballots in the public board. The number of valid ballots is also leaked unless more sophisticated tally methods are used such as tally-hiding schemes [13], [19], [24].

However, JCJ leaks much more information when revoting is allowed, namely:

- $n_b$, the total number of received ballots;
- $n_v$, the total number of valid (and counted) ballots;
- $n_r$, the total number of revotes;
- the complete distribution of revotes, per (encrypted) credential (hence, for all $k$, the number of credentials used to revote $k$ times).

This can be exploited by a coercer to detect when a coerced voter disobeys. Indeed, there is no reason to assume that revoting is independent from the choice of candidate. On the contrary, revoting is often due to voters changing their mind between candidates, for instance due to some late announcements in the press.

*a) An attack against coercion-resistance:* We consider an extreme case, with two candidates $A$ and $B$. Suppose that voters voting for $A$ do not revote while those voting for $B$ always revote, exactly once. We denote $r_A$ (resp. $r_B$) the number of votes for $A$ (resp. $B$). Due to the considered revoting behaviors, the number of revotes $n_r$ corresponds to the number of votes for $B$ sent by the honest voters.

Assume now that Alice wants to vote for $B$ but is instructed by her coercer to vote for $A$ (or abstain).

- If Alice obeys, the coercer will observe $r_B = n_r$.
- If Alice disobeys and casts one ballot for $B$, the coercer will observe that $r_B = n_r + 1$.

Hence the coercer will detect that Alice has disobeyed, which breaks coercion-resistance.

One could argue that Alice should follow a different evasion strategy and cast one ballot if she votes for $A$ and two if she votes for $B$. This does not work either. Indeed, assume now that Alice wants to vote for $A$, but is instructed to vote for $B$.

- If she obeys, she gives her real credential $c$ to her coercer. The latter then casts *exactly one* ballot for $B$ using $c$.
- Otherwise, she provides a fake credential $c'$, that the coercer uses to vote for $B$. Alice then votes for $A$ using $c$.

In the first case, $r_B = n_r + 1$, while in the second case, $r_B = n_r$. Once again, the coercer is able to detect that Alice disobeyed and coercion-resistance is broken.

### C. Discussion

*1) Considering other evasion strategies:* One possibility to correct JCJ's flaw is to define other evasion strategies in case of revoting. Indeed, if Alice wants to vote, JCJ's evasion strategy instructs her to do so exactly once. Consequently, if it is usual for everyone to revote several times, the leakage in JCJ allows the coercer to detect that a single person voted once without revoting, and thus that Alice disobeyed. However, it seems very hard to instruct voters to use revoting, according to a certain distribution, when they are under coercion. As illustrated in Section II-B, the natural way to proceed does not work. This is made even harder by the fact that the strategy may evolve depending on new events that could change the revoting distribution for the honest voters.

Hence we propose another option (see Section IV) that consists in reinforcing JCJ in case of revoting, such that there is no leakage besides the total number of ballots and the number of valid ballots. For our proposed protocol, we prove coercion-resistance with the original evasion strategy of JCJ. We acknowledge that the latter is not perfect; in particular, it does not allow a voter under coercion to change their mind and revote. However, modeling a wide variety of behaviors for the coerced voter is too complex and is out of the scope of this article.

*2) More noise is needed:* A known issue of JCJ is that fake ballots should be randomly added, in order to hide to a coercer that their ballot has been removed. Indeed, if it is usual that absolutely no ballot with a fake credential is removed at Step 3 of the tally, then a coercer, who observes that exactly one ballot is removed, would suspect that the coerced voter has disobeyed.

Hence, it is necessary that an unpredictable number of ballots is removed during the tally. In JCJ, this "noise" comes from honest voters sending dummy ballots, but this source alone may not be sufficient. A natural approach is to have the authorities add a random number of dummies. For instance, [32] uses this to mitigate a leakage during the tally. This noise made of fake ballots should however be calibrated carefully since the computation overhead is important. In a context where revoting is a well spread behavior, it could be judicious to rely on revoting, at least partially, as an additional source of noise. This is not possible in JCJ where a dummy can be distinguished from a revote, but becomes a possibility if our solution from Section IV is used.

## III. The impact on coercion-resistance

In Section II, we explain the leakage of the JCJ protocol and we illustrate, in an extreme scenario, how this can be exploited to completely break coercion-resistance. In this section, we estimate the impact of the leakage in more realistic scenarios. For this purpose, we use the framework of [25] which allows to quantify the coercion level of a voting protocol.

### A. Quantifying coercion-resistance

We consider $n_V$ voters, among which one is under coercion. The others are supposed honest and independent. They choose a voting option among $C+1$ possibilities, which includes abstention and maybe blank voting. We suppose that the choices follow a probability distribution $(P_0, \dots, P_C)$, where $P_0$ is the probability to abstain. Let $\alpha$ be the voting option corresponding to the intention of the coerced voter, and $\beta$ be the one that is the instruction of the coercer. The coerced voter either disobeys, gives a fake credential and votes with option $\alpha$ (the evasion strategy does not imply any revote), or obeys and gives their real credential which the coercer uses to vote with option $\beta$. The coercer must decide whether the voter obeyed or not, given only the result.

The ideal result is $\mathsf{R}^{\mathsf{Ideal}} = \overrightarrow{res} = (\mathsf{res}_0, \cdots, \mathsf{res}_C)$, the number of voters who opted for each option. In JCJ, however, the real result $\mathsf{R}^{\mathsf{Real}}$ is $\overrightarrow{res}$, as well as, for all $k$, the number of voters who revoted $k$ times. In addition, both results should also include the number of invalid ballots. However, to focus on the leakage of JCJ, we assume that a large and unpredictable number of ballots with a fake credential are cast, so that the adversary cannot gain information by observing the number of invalid ballots. This approximation is necessary to use the framework of [25], which does not model the possibility to cast a ballot with an invalid credential in the ideal setting.

We now instantiate [25] in our scenario. To simplify the analysis, we assume that a voter revotes at most once, so that $\mathsf{R}^{\mathsf{Real}} = (\overrightarrow{res}, n_R)$, where $n_R$ is the total number of revotes. We also assume that all the parties are honest except for the coercer, and that the cryptography is perfect, so that the coercer does not learn any other information than the result. With these assumptions, we define the real and ideal games, where the behavior of the coerced voter is decided at random. The coercer wins the real (resp. ideal) game if they correctly guess the behavior of the coerced voter given the real (resp. ideal) result. For $g \in \{\mathsf{Real}, \mathsf{Ideal}\}$ and for a pair $(\alpha, \beta)$, we denote $W^g_{\alpha,\beta}$ the event when the coercer wins the game, and $\delta^g_{\alpha,\beta} = 2|\Pr(W^g_{\alpha,\beta}) - 1/2|$. Furthermore, we consider the worst case for the voter, and analyze the quantity $\delta^g = \max_{\alpha,\beta} \delta^g_{\alpha,\beta}$. We call $\delta^{\mathsf{Real}}$ (resp. $\delta^{\mathsf{Ideal}}$) the *coercion level* of the real (resp. ideal) game. Intuitively, it measures how much better the adversary's strategy is compared to a random guess, in a scale from 0 to 1.

We denote by $\Pr(\mathsf{R}^g|\alpha)$ (resp. $\Pr(\mathsf{R}^g|\beta)$), the probability that the result $\mathsf{R}^g$ is obtained, assuming the voter votes for $\alpha$ (resp. obeys the coercer and votes for $\beta$). According to [25], the best strategy for the coercer is to assume that the voter obeyed if and only if $\Pr(\mathsf{R}^g|\beta) \geq \Pr(\mathsf{R}^g|\alpha)$. This gives a close formula for the coercion level which can be written as

$$\delta^g = \max_{(\alpha,\beta)} \sum_{\mathsf{R}^g \in M_{\alpha,\beta}} \Pr(\mathsf{R}^g|\beta) - \Pr(\mathsf{R}^g|\alpha), \tag{1}$$

where $M_{\alpha,\beta}$ is the set of all possible results $\mathsf{R}^g$ such that $\Pr(\mathsf{R}^g|\beta) \geq \Pr(\mathsf{R}^g|\alpha)$. In Appendix D, we explain in more details how this formula can be computed. In the remaining of this section, we compare $\delta^{\mathsf{Real}}$ and $\delta^{\mathsf{Ideal}}$ in two scenarios, where external events provoke many revotes. In these scenar-

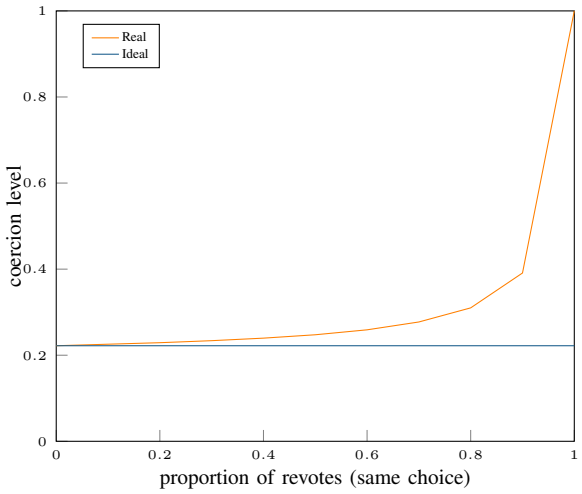

Fig. 2: Coercion levels as a function of the revote, with 20 voters, 2 candidates, 30% abstention and a 50%-50% distribution of votes between the candidates.

ios, we assume that there are two candidates $A$ and $B$, no blank vote, but the possibility to abstain or to revote once.

### B. The technical incident scenario

In general, we can expect revoting to be rare. This is something that is not allowed in classical paper-based elections, so that in a context where electronic voting is recent, voters will not be using this possibility. Even in a country such as Estonia, where revoting is available for internet voters since 2005, a recent study revealed a revoting rate of about $2\%$ [16].

However, much more revotes could occur if an announcement reveals a suspicion of a technical incident, and encourages the voters to revote to be on the safe side. In this case, many voters could be inclined to revote with the same voting option, which would seem harmless if they are not aware of the weaknesses of JCJ. Note that the coercer could be the source of such an announcement, and spread fake news about the necessity to revote.

In Fig. 2, we consider this situation where a proportion of voters (that already voted) revote for the same voting option.

We plot the coercion level in both the real and ideal settings when the proportion $x$ of revotes ranges from $0$ to $1$. When $x = 0$, both coercion levels are the same since there is no revote. However, when $x = 1$, there is no coercion-resistance in JCJ because the coerced voter would be the only one to cast a ballot without revoting. Note that the ideal coercion level remains constant since the overall probability to choose each voting option is unaffected by $x$.

### C. A discredit in the press

We still consider two candidates $A$ and $B$, and in this scenario, we assume that during the period of the voting phase, the candidate $A$ is discredited by an announcement in the press. As a consequence, some proportion of the voters who initially voted for $A$ will change their mind and revote for $B$. For simplicity, we assume that no revote occurs that is not due to this event.

Such discredits have happened in the past. For instance, Dominique Strauss-Kahn, a former IMF managing director, was highly expected to become the next French president in 2012. However, due to an accusation of sexual assault, his political party chose to support another candidate. This occurred before the time of the election, and no electronic voting was involved. But, we can mention the 2022 Tory leadership election for the succession of Boris Johnson, the members could vote by Internet, and revote was initially authorized (before a security concern forced the organizers to forbid it). The duration was more than a month, which is more than enough for a discredit event to occur (even though it did not occur, for this election).

First, we fix a small number of voters, so that the effect is more visible, and we study the influence of the other parameters.

In Fig. 3, we plot the real and ideal coercion levels as the proportion $x$ of voters who change their mind from $A$ to $B$ ranges from $0$ to $1$. When $x = 0$, there is no difference since there is no revote. When $x = 1$, there is no difference either since nobody votes for $A$ anymore, so that there is no coercion-resistance in both the real and ideal games. However, a non-negligible difference can be observed for the intermediate values of $x$.

In Fig. 4, we plot the real and ideal coercion levels with a fixed value of $x = 0.3$ (*i.e.* $30\%$ of the voters who voted for $A$ revote for $B$) and we let the initial proportion $p$ in favor of $A$ vary from $0$ to $1$. When $p = 0$, everyone votes for $B$ so that there is no coercion-resistance. When $p$ is large, we get close to the scenario presented in Section II-B, so that there is no coercion-resistance in the real game while the ideal game still offers some coercion-resistance.

In Fig. 5, we plot the real and ideal coercion levels with fixed $x = 0.3$ and $p = 0.7$ and we let the abstention rate $P_0$ range between $0$ and $1$. When $P_0 = 0$, there is no coercion-resistance because forced-abstention attacks are trivial; similarly, there is no coercion-resistance when $P_0 = 1$. However, a non-negligible difference can be observed for the intermediate values.

Finally, in Fig. 6, we plot the real and ideal coercion levels with fixed $x = 0.3$, $p = 0.7$ and $P_0 = 0.3$, for a number of uncoerced honest voters equals to $16, 32, 64, 128, 256, 512$ and $1024$. This shows that the difference between both coercion levels remains non-negligible even when the number of voters is large. An asymptotic analysis (see *e.g.* [15]) reveals that the coercion level decreases in $1/\sqrt{n_V}$, so the level of coercion is small anyway.

## IV. CHIDE: A CLEANSING-HIDING PROTOCOL

We propose a modification of JCJ that provides full coercion-resistance. During the tally phase, the trustees perform the same tasks of cleansing, mixing and decrypting than in JCJ, but in a hidden way, so that the coercer (or anyone) does not learn how many ballots were deleted because

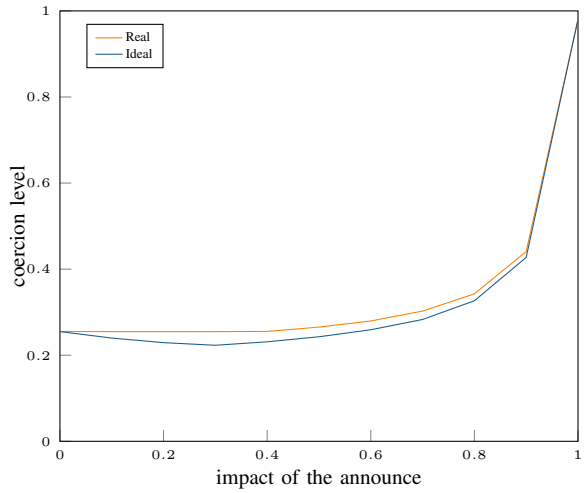


Fig. 3: Coercion levels as a function of the impact for 20 voters, 2 candidates, 30% abstention and a 70%-30% distribution between the candidates.

they correspond to revotes or to invalid credentials. For this purpose, we propose a novel cleansing algorithm based on MPC primitives.

### A. Cryptographic primitives

The security of the following primitives relies on the Decisional Diffie-Hellman (DDH) assumption and the Random Oracle Model.

*ElGamal encryption.* We use the exponential ElGamal encryption scheme on elliptic curves, which is convenient for its efficiency and its homomorphic property. Let $g$ be a public generator and $\mathsf{pk}$ the public key, *i.e.* a group element, the discrete logarithm of which is the corresponding secret key. One encrypts $m$ by choosing a random exponent $r$ and computing $\mathsf{Enc}(m, \mathsf{pk}) = (g^r, g^m\, \mathsf{pk}^r)$. We impose $m$ to be in a small list of valid messages, so that decryption is feasible (in general, recovering $m$ from $g^m$ is hard). An important special case is when $m$ is a bit, because this is the kind of input used by the MPC primitives we mention below. For a generic message $m$, we call bit-wise encryption of $m$ the list of the encryptions of the bits of $m$.

*Zero Knowledge Proofs.* We use Non-Interactive Zero Knowledge Proofs (ZKP) based on the Fiat-Shamir transformation, mostly for proving relations involving discrete logarithms, *à la* Chaum-Pedersen.

*Distributed key generation / threshold decryption.* A DKG is a protocol which allows the participants to generate an ElGamal public key $\mathsf{pk}$ in such a way that each participant gets a share of the secret key. The protocol also generates a public commitment to each participant's share. A threshold $t$ is set, such that a collusion of $t$ or less participants can not deduce any information about the secret key, or about the cleartext of any ciphertext. If $t + 1$ or more participants collaborate, they can combine their shares to recover the secret key. They can also run a threshold decryption protocol which allows to jointly decrypt any ciphertext without recovering the secret key nor revealing anything about their shares. The result comes with a ZKP of correct decryption that anyone can verify. See for instance [35] for an instantiation.

*Verifiable decryption mixnets.* A decryption mixnet is an MPC protocol where the participants take a list of ciphertexts and reveal the corresponding plaintexts, in an order that is unrelated to the initial order, so that it is not possible to tell which one comes from which ciphertext. We will assume that the output plaintexts are sorted in the lexicographic order. The result comes with a corresponding ZKP, for correctness. See for instance [34] for an instantiation.

*Logical operations on encrypted bits.* We use MPC protocols that allow the owners of the shares to jointly perform logical operations on encrypted bits, based on their threshold decryption protocol. This is done without revealing the cleartexts to anyone, and in a verifiable manner. The main building block we use is the $\mathsf{CGate}$ protocol [31], that allows to conditionally set an encrypted value $X$ to (an encryption of) 0, given an encrypted bit $Y$. In other words, if $x$ and $y$ are the corresponding plaintexts with $y \in \{0, 1\}$, $\mathsf{CGate}(X, Y)$ is a random encryption of $xy$. By combining this with the homomorphic property of the ElGamal encryption, one can derive a protocol for all the logical operations on bits (*e.g.* negation, disjunction).

More precisely, we use the $\mathsf{And}$ (conjunction) and the $\mathsf{Eq}$ (equality test) protocols. The latter is extended to bit-wise encrypted data, by computing the conjunction of all the equality tests on encrypted bits. See Appendix A for more details about these MPC protocols.

*Sorting encrypted data.* Using the above logical operations, it is straightforward to design a comparison test $\mathsf{Lt}$ and a conditional swap $\mathsf{CSwap}$. Therefore, it is possible to sort encrypted data, without revealing anything about the data (the conditional swap uses reencryptions, so that it is not possible to determine whether the data were swapped or not). For this purpose, we need a *data-oblivious* sorting algorithm, that is an algorithm whose control flow does not depend on the result of the comparisons. The popular fast sorting algorithms, such as Quicksort, Mergesort or Heapsort, do not verify this property. Consequently, we use the OddEvenMergeSort by Batcher [3], which has a quasi-linear complexity.

### B. Description of the CHide protocol

*Participants.* The CHide protocol is similar to JCJ, and the list of participants is the same. We recall them and emphasize the trust assumptions on them, which are the same as in JCJ. For simplicity, we assume that there is a single honest registrar. The literature following the JCJ approach contains techniques to have several registrars and appropriate trust assumptions on them [11]; this is orthogonal to the present discussion.

- The **public board** is an append-only list of data where all the participants can write. At any time, the content of the board can be read by anyone, and the view is the same. The board is assumed to be honest; see [18] for a possible realization of this.

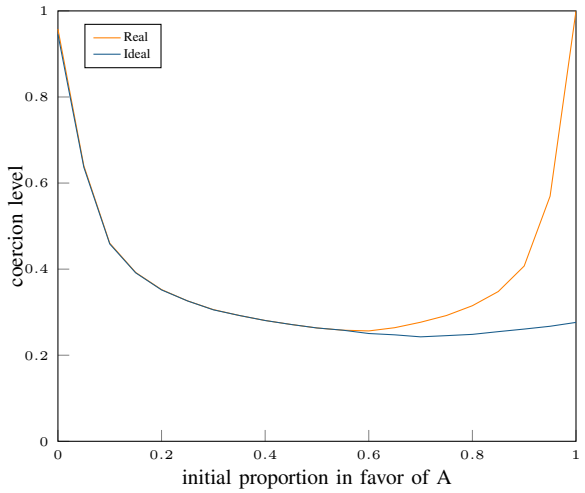


Fig. 4: Coercion levels as a function of the proportion in favor of A for 20 voters, 2 candidates and 30% abstention.

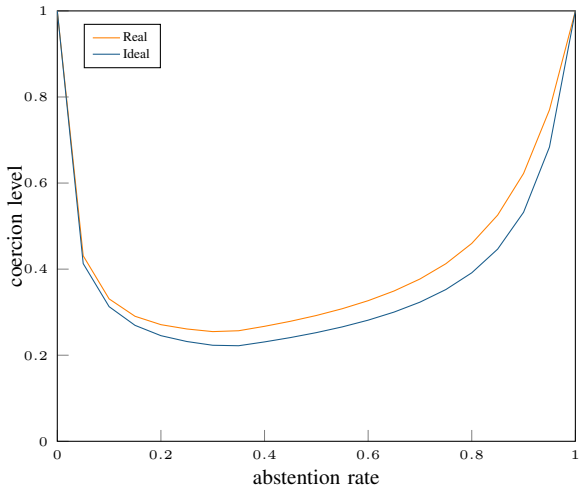


Fig. 5: Coercion levels as a function of the abstention for 20 voters, 2 candidates, 21% revotes and a distribution of 70%-30% between the candidates.

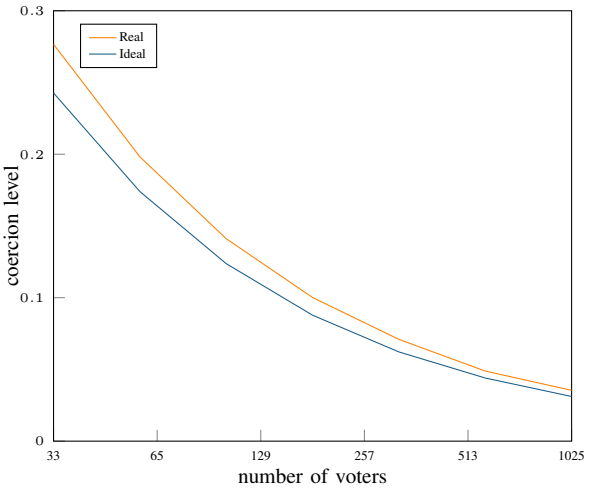


Fig. 6: Coercion levels as a function of the number of voters with 30% abstention, 21% revotes and a distribution of 70%-30% between the candidates.

- The **auditors** check the consistency of the board, including the validity of all the ZKPs. We assume that there is at least one honest auditor that reveals any detected problem.
- The **registrar** sends their voting material to the legitimate voters. It is assumed that the registrar is honest.
- The $n_T$ **election trustees** hold the key shares and perform the cleansing and the tally. It is assumed that at most $t$ dishonest trustees can collaborate, where $t < n_T$ is the threshold used in the DKG.
- There are $n_V$ **voters**. Some of them may be dishonest and collude with the attacker. Honest voters may be subject to a coercion attempt by the attacker.

*Setup phase.* During this phase, a security parameter $\lambda$ is chosen. The election trustees jointly run the DKG protocol, yielding a public key $\mathsf{pk}$ for this security level, and a secret share for each trustee. The DKG also produces public data that is sent to the public board for verifiabilty. We denote `Setup` the corresponding protocol.

*Registration phase.* For $1 \leq i \leq n_V$, the registrar generates $\lambda$ encryptions $\mathcal{R}_i = (R_{i,1}, \ldots, R_{i,\lambda})$ of random bits $(c_{i,1}, \ldots, c_{i,\lambda})$. These are called credentials. The registrar sends them to the board as a roster $\boldsymbol{R} = (\mathcal{R}_i)_{1 \leq i \leq n_V}$. The registrar shuffles the list of credentials and sends one of them to each voter. Just as in JCJ, we consider that the registration is perfect; but it can be augmented with DVZKPs [21] to allow the voter to verify the validity of their credential with respect to the public roster.

This simplified registration protocol assumes a single honest registrar. For several registrars, the (encrypted) bits of the credentials can be jointly generated, and the credentials can be sent to the voters without revealing them to the registrars, as long as one of them is honest [11]. The registrars can also jointly prove that the credentials are indeed encrypted bit by bit, and thus that the public roster is well-formed (see *e.g.* [7]).

*Voting phase.* In order to cast a vote for the option $\nu$ (encoded as a group element), a voter computes $C_1 = \mathsf{Enc}(\nu, \mathsf{pk})$ and $C_2 = (\mathsf{Enc}(c_1, \mathsf{pk}), \ldots, \mathsf{Enc}(c_\lambda, \mathsf{pk}))$, where $c = (c_1, \ldots, c_\lambda)$ is their credential. The neutral element 1 (the encoding $g^0$ of the zero bit) should not represent any voting option as it will be used to encode invalid ballots. The voter also produces a ZKP $\pi_1$ that proves the knowledge of $\nu$ and $c_j$ for all $j$. To ensure a strong Fiat-Shamir transformation [6], the computation of the challenge from the commitment of the $\Sigma$-protocol must include all public informations in the hash, such as $g$, $\mathsf{pk}$, $C_1$ and $C_2$. Finally, a ZKP $\pi_2$ that $\nu$ is a valid voting option and that $c_1, \ldots, c_\lambda$ are bits must be added to prevent forced-abstention attacks which use write-

ins. The ballot $(C_1, C_2, \pi_1, \pi_2) = \texttt{Vote}_{\mathsf{pk}}(c, \nu)$ is sent to the public board, using an anonymous channel. The voters check that their ballot is present on the board; this defines the verification step `Check`. The auditors verify that the ZKPs are valid and that there is no other ballot on the board with the same $(C_1, C_2)$; this defines the verification `isVal`.

*Cleansing and tallying phase.* Just as in JCJ, the election trustees keep only one ballot per valid credential and remove all the other ones. However, they do so in an oblivious way, by replacing the voting option of an invalid ballot by an encryption of 0, which represents an invalid voting option. For this purpose, the trustees first compute an encrypted validity boolean for each ballot. Using the toolbox from [13], we could simply propose an MPC variant of JCJ, with a quadratic number of comparisons, as in JCJ. We instead propose a quasi linear approach, which relies on sorting.

The idea is to create a list of pairs of encrypted data $(V_i, K_i)_{1 \leq i \leq n_b + n_V}$, both for ballots posted to the board and credentials from the roster. For ballots, $V_i$ contains the voting option (the $C_1$ part), and $K_i = (K_i^{\perp}, K_i^{\top})$ is formed of the encrypted order of appearance on the public board (*i.e.*, $K_i^{\perp}$ is a bitwise-encrypted integer between 0 and $n_B - 1$), and from the encrypted credential (*i.e.*, $K_i^{\top}$ is the $C_2$ part). This step can be performed by anyone using a fixed randomness for $K_i^{\perp}$ (e.g. using the random 0) and is publicly verifiable. For entries coming from the roster, $V_i$ is the encryption of an invalid option, and $K_i = (K_i^{\perp}, K_i^{\top})$ contains the encrypted integer $n_B$ in the first part, and the credential in the second part. Again, this step is performed using a fixed randomness for $V_i$ and $K_i^{\perp}$.

Then the talliers use the toolbox from [13] to sort this list in MPC, in quasi linear time. They use the following order:

- first sort according to $K_i^{\top}$ in increasing order. The effect is to group the ballots by credentials;
- for equal $K_i^{\top}$ (*i.e.* for equal credentials), sort in increasing order of $K_i^{\perp}$. The effect is to position the entry coming from the roster at the end of the group and the valid ballot (if there is one) just before it.

After this step, an entry is valid iff 1) its $K_i^{\top}$ part is the same as the one from its successor in the sorted list; and 2) the $K_i^{\perp}$ part of its successor encodes $n_B$. These tests can be efficiently implemented with the MPC toolbox and we need only a linear number of such tests, yielding an overall procedure in quasi linear time.

The $P_{\texttt{tally}}$ protocol is more precisely presented as follows. A more detailed description is available in Appendix A.

1. Discard all the ballots marked as invalid by the audit procedure. Let $(C_1^i, C_2^i)_{i=1}^{n_B}$ be the remaining ballots, without the ZKPs. We denote $m = \lceil \log n_B \rceil + 1$.
2. For all $1 \leq k \leq n_B$, set $V_k = C_1^k$ and $K_k = U_{k-1} || C_2^k$, where $U_{k-1}$ is a bit-wise encryption of $k-1$ over $m$ bits (least significant bit first), using the null randomness.
3. For all $n_B + 1 \leq k \leq n_B + n_V$, set $K_k = U_{n_B} || \mathcal{R}_{k-n_B}$ and $V_k$ is the encryption of 0, using the null randomness.
4. Sort the $(V_k, K_k)$ using the keys $K_k$, in increasing order. This produces a result $(V'_k, K'_k)_{k=1}^{n_B+n_V}$ and a transcript $\Pi^{\mathsf{Sort}}$.
5. For all $1 \leq k < n_B + n_V$, compute $D_k = \mathsf{Eq}(K'^{\top}_k, K'^{\top}_{k+1})$, where $K'^{\top}_k$ refers to the $\lambda$ most significant bits of $K'_k$. This produces the transcript $\Pi^{\mathsf{Eq}}_{k,1}$.
6. For all $1 \leq k < n_B + n_V$, compute $F_k = \mathsf{Eq}(K'^{\perp}_{k+1}, U_{n_B})$, where $K'^{\perp}_{k+1}$ refers to the $m$ least significant bits of $K'_{k+1}$. This produces the transcript $\Pi^{\mathsf{Eq}}_{k,2}$.
7. For all $1 \leq k < n_B + n_V$, replace $V'_k$ by $\mathsf{CGate}(V'_k, \mathsf{And}(D_k, F_k))$.
8. Apply the decryption mixnet protocol on the $(V'_k)_{k=1}^{n_B+n_V-1}$. This produces the result of the election as well as a verification transcript $\Pi^{\mathsf{Mixnet}}$.

Each step produces a transcript, published on the board, and verified by the auditors.

*Evading coercion.* We provide the same evasion strategy as in JCJ, which consists of giving a fake credential to the coercer and to vote (once, if the voter wants to) with the real credential.

*Assumptions on the communication channels.* As in the original JCJ protocol, the communications between the voters and the registrars must be untapable, and the communication between the voters and the public board must be anonymous. This is required only for the coercion-resistance property. Verifiability and vote secrecy rely on standard communication channels assumptions.

For verifiability, we assume that honest voters check that their ballots appear on the public board and complain as soon as something unexpected happens. For weaker voters that check their ballots only when they vote, the ballot order should be enforced, for instance using a chain of hashes (see e.g. [2]).

### C. Efficiency considerations

In terms of computational and communication costs, CHide is less efficient than JCJ, mainly because the encrypted credentials are now formed by $\lambda$ ciphertexts instead of a single one.

For the talliers, the cleansing phase is more complex but scales better with the number of submitted ballots. While JCJ is quadratic, we propose a quasi-linear tally protocol based on sorting. Another difference is that, due to the MPC toolbox, the number of communication rounds between them is no longer constant, but becomes logarithmic in the number of ballots. Nevertheless, the task is highly parallelizable and remains affordable for medium-size elections.

For the voters, the computational load increases but the total cost for realistic parameters is around a thousand exponentiations, which should be a matter of seconds with a standard implementation in Javascript running within a modern browser.

In Table I, we give estimates of the number of exponentiations and of the transcript size for both JCJ and CHide. For this purpose, we consider a number of $n_T = 3$ talliers with a threshold $t = 2$, a security parameter of $\lambda = 128$

TABLE I: Number of exponentiations and transcript size in JCJ and CHide, with $\lambda = 128$.

| # voters | # exp. | | estimated CPU time | | transcript | |
|---|---|---|---|---|---|---|
| any | JCJ | CHide | JCJ | CHide | JCJ | CHide |
| (Vote) | 27 | 1.4k | 5.4ms | 0.28s | 1.1kB | 58kB |
| 10 | JCJ | CHide | JCJ | CHide | JCJ | CHide |
| (Tally) | 4.26k | 4.1M | 0.43s | 6.8min | 170kB | 65MB |
| 100 | JCJ | CHide | JCJ | CHide | JCJ | CHide |
| (Tally) | 380k | 120M | 38s | 3.3h | 16.0MB | 1, 9GB |
| 1000 | JCJ | CHide | JCJ | CHide | JCJ | CHide |
| (Tally) | 37.5M | 2.4G | 1.0h | 2.8d | 1.59GB | 39GB |
| 10000 | JCJ | CHide | JCJ | CHide | JCJ | CHide |
| (Tally) | 3.75G | 41G | 4, 3d | 48d | 158GB | 668GB |
| 100000 | JCJ | CHide | JCJ | CHide | JCJ | CHide |
| (Tally) | $375G$ | 658G | 1.2y | 2.1y | 15.8TB | 10.6TB |
| 1000000 | JCJ | CHide | JCJ | CHide | JCJ | CHide |
| (Tally) | 37.5T | 9.4T | 120y | 30y | 1.58PB | 152TB |

and a number of $n_C = 2$ voting options. We also give the corresponding running times, based on an estimate of 5000 exponentiations per second on the client side, and 10000 per second on the server side. For 1 000 000 voters, CHide starts being more efficient than JCJ. This is due to the fact that JCJ has a quadratic complexity while CHide is quasi linear. While the cost is high, this is not out of reach since the computations are highly parallelizable and hence could be conducted within one day on a (large) cluster.

## V. Defining coercion-resistance

One of the reasons why the flaw was not discovered in the original proposition of the JCJ protocol is the fact that the definition of coercion-resistance itself was flawed, in the sense that no scheme can be proved secure w.r.t. this definition. We then introduce our own definition, that is used to analyse the security of CHide.

### A. Voting system

A *voting system* is a tuple of eight algorithms or protocols $P_{\texttt{Setup}}$, `register`, `Vote`, `Check`, `isVal`, `Fakecred`, $P_{\texttt{tally}}$, `Verify` such that:

- $P_{\texttt{Setup}}(\lambda, n_T, t)$ is a protocol run by $n_T$ authorities for the security parameter $\lambda$ and the threshold $t$. It computes the public key $\mathsf{pk}$, as well as the secret and public shares $(s_i, h_i)$ for each authority, using a DKG.
- $\texttt{register}(\lambda, \mathsf{pk}, n_V)$ generates a private credential $c_i$ for each voter $\in [1, n_V]$. It also returns some public information $\boldsymbol{R}$ that contains the public part of the credentials, and any necessary transcript for proving their validity.
- The algorithm $\texttt{Vote}_{\mathsf{pk}}(c, \nu)$ takes as input the public key of the election $\mathsf{pk}$, a credential $c$, and a vote $\nu$, and returns a ballot. The public key is often omitted for simplicity.
- The algorithm $\texttt{Check}(BB, \mathsf{pk}, \mathbf{b}, c, \nu)$ takes the bulletin board $BB$, the public key, a ballot $\mathbf{b}$, a private credential $c$ and a voting option $\nu$. It is run by the voters to check that their ballot has been added to the public board.
- The algorithm $\texttt{isVal}(\mathsf{pk}, \mathbf{b}, BB)$ takes as input the public key, a ballot and the ballot box. It outputs a bit which states whether the ballot is valid w.r.t. $BB$.
- $\texttt{Fakecred}(c)$ takes a credential and returns a fake one $\tilde{c}$.
- $P_{\texttt{tally}}(BB, \boldsymbol{R}, \mathsf{pk}, \{h_i, s_i\}, t)$ is a protocol run by the authorities that possess the secret shares $\{s_i\}$ with public commitments $\{h_i\}$. It takes as input a list of ballots $BB$, the public roster $\boldsymbol{R}$, the public key $\mathsf{pk}$, the threshold $t$ and returns the result $\boldsymbol{X}$ of the election together with a proof $\Pi$.
- $\texttt{Verify}(BB, \Pi, \mathsf{res})$ takes as input a ballot box $BB$, a transcript $\Pi$ and a result $\mathsf{res}$. It outputs a bit which states whether the result is valid with respect to $BB$.

### B. The original definition of JCJ

The intuition of the JCJ definition of coercion-resistance is that an adversary must not guess whether a coerced voter obeyed or evaded coercion. When the voter obeys ($b = 1$ in the definition), they give their real credential and abstain from doing any other action. Note that a coercer may ask the voter to cast some specific vote or to perform some specific computations, but this is not considered in the definition as the adversary might as well do it themself, with the given credential. When the voter evades ($b = 0$ in the definition), they give a fake credential and cast a single vote for the desired voting option (or abstain, depending on their personal choice).

This yields the game $\mathsf{Real}^{\mathsf{CR}}_{\mathsf{JCJ}}$ presented in Algorithm 1. Voting choices are represented as integers between 1 and $n_C$, and $\phi$ represents the choice to abstain. During this game, the adversary selects the set of corrupted voters. It is given the corresponding private credentials as well as all the public information $\boldsymbol{R}$ (*i.e.* the encrypted credentials in the JCJ protocol). It then chooses $(j, \alpha)$, where $j$ denotes the voter under coercion and $\alpha$ their desired voting option. The evasion strategy is modeled in lines 11 and 13: when the voter disobeys, they create a fake credential and cast a vote for $\alpha$ (or abstain if $\alpha = \phi$). Otherwise, they give their real credential.

Honest voters vote according to a distribution which depends on the number of options $n_C$ and returns a value that may be:

- any valid vote $\nu \in [1, n_C]$;
- $\phi$, which represents abstention;
- $\lambda$, which represents casting a vote with a fake credential.

We extend the `Vote` function to votes equal to $\lambda$ as follows.

$$\texttt{Vote}_{\mathsf{pk}}(c, \lambda) = \texttt{Vote}_{\mathsf{pk}}(\tilde{c}, \nu),$$

where $\tilde{c} = \texttt{Fakecred}(c)$ and $\nu$ is sampled from $[1, n_C]$.

It is worth noting that the advantage of an adversary in game $\mathsf{Real}^{\mathsf{CR}}_{\mathsf{JCJ}}$ will always be non negligible since one can compare the result of the tally with the expected result, given the distribution $\mathcal{D}$ of the voting intentions. For example, if the adversary wants to cast a vote for a very unlikely candidate, they may observe cases where the latter does not get a single vote in the result, which is a clear indication that the coerced voter disobeyed. Hence, the JCJ definition compares the advantage of an adversary in game $\mathsf{Real}^{\mathsf{CR}}_{\mathsf{JCJ}}$ with the one in an ideal game $\mathsf{Ideal}^{\mathsf{CR}}_{\mathsf{JCJ}}$, where there is no other information

**Algorithm 1:** $\mathsf{Real}^{\mathsf{CR}}_{\mathsf{JCJ}}$

**Require:** $\mathbb{A}, \lambda, n_T, t, n_V, n_A, n_C, \mathcal{D}$

1 $BB \longleftarrow \varnothing$
2 $\mathsf{pk}, (s_i, h_i)_{i=1}^{n_T} \longleftarrow \mathtt{Setup}^{\mathbb{A}}(\lambda, n_T, t)$
3 $A \longleftarrow \mathbb{A}()$
4 $\{c_i; i \in [1, n_V]\}, \boldsymbol{R} \longleftarrow \mathtt{register}(\lambda, \mathsf{pk}, n_V)$
5 $(j, \alpha) \longleftarrow \mathbb{A}(\{c_i; i \in V\}, \boldsymbol{R})$
6 **if** $|A| \neq n_A \vee j \notin [1, n_V] \backslash A \vee \alpha \notin [1, n_C] \bigcup \{\phi\}$ **then**
7  Return 0
8 $b \xleftarrow{\$} \{0, 1\}$
9 $\tilde{c} \longleftarrow c_j$
10 **if** $b == 0$ **then**
11  $\tilde{c} \longleftarrow \mathtt{Fakecred}(c_j)$
12  **if** $\alpha \neq \phi$ **then**
13  $BB \longleftarrow BB \bigcup \{\mathtt{Vote}_{\mathsf{pk}}(c_j, \alpha)\}$
14 **for** $i \in [1, n_V] \backslash (A \bigcup \{j\})$ **do**
15  $\nu_i \longleftarrow \mathcal{D}_{n_C}()$
16  **if** $\nu_i \neq \phi$ **then**
17  $BB \longleftarrow BB \bigcup \{\mathtt{Vote}_{\mathsf{pk}}(c_i, \nu_i)\}$
18 $BB \longleftarrow BB \bigcup \mathbb{A}(\tilde{c}, BB)$
19
20
21 $\boldsymbol{X}, \Pi \longleftarrow P^{\mathbb{A}}_{\mathtt{tally}}(BB, \boldsymbol{R}, \mathsf{pk}, \{h_i, s_i\}, t)$
22 $b' \longleftarrow \mathbb{A}()$
23 Return 1 **if** $b' == b$ **else** 0

**Algorithm 2:** $\mathsf{Ideal}^{\mathsf{CR}}_{\mathsf{JCJ}}$

**Require:** $\mathbb{A}, \lambda, n_V, n_A, n_C, \mathcal{D}$

1 $D \longleftarrow \varnothing$
2
3 $A \longleftarrow \mathbb{A}(\lambda)$
4
5 $(j, \alpha) \longleftarrow \mathbb{A}()$
6 **if** $|A| \neq n_A \vee j \notin [1, n_V] \backslash A \vee \alpha \notin [1, n_C] \bigcup \{\phi\}$ **then**
7  Return 0
8 $b \xleftarrow{\$} \{0, 1\}$
9
10 **if** $b == 0 \wedge \alpha \neq \phi$ **then**
11  $D \longleftarrow D \bigcup \{\alpha\}$
12
13
14 **for** $i \in [1, n_V] \backslash (A \bigcup \{j\})$ **do**
15  $\nu_i \longleftarrow \mathcal{D}_{n_C}()$
16  **if** $\nu_i \notin \{\phi, \lambda\}$ **then**
17  $D \longleftarrow D \bigcup \{\nu_i\}$
18 $(\nu_i)_{i \in A}, \beta \longleftarrow \mathbb{A}()$
19 **if** $b == 1 \wedge \beta \in [1, n_C]$ **then**
20  $D \longleftarrow D \bigcup \{\beta\}$
21 $\boldsymbol{X} \longleftarrow \mathtt{result}(D \bigcup \{\nu_i \mid i \in A, \nu_i \in [1, n_C]\})$
22 $b' \longleftarrow \mathbb{A}(\boldsymbol{X})$
23 Return 1 **if** $b' == b$ **else** 0

Fig. 7: JCJ definition of coercion resistance. $\lambda$ is the security parameter, $n_T$ the number of talliers, $t$ the threshold, $n_V$ the number of voters, $n_A$ the number of corrupted voters, $n_C$ the number of voting options and $\mathcal{D}$ the distribution of votes.

than what is unavoidably leaked, that is, the result. The game $\mathsf{Ideal}^{\mathsf{CR}}_{\mathsf{JCJ}}$ is presented in Algorithm 2. Compared to the original definition, we present a slightly modified version that reasons on the clear votes only. This simplifies the understanding by focusing on the information given to the adversary. All our claims and remarks hold on the original definition as well.

**Definition 1** (adapted from [22])**.** *A voting system is* JCJ-coercion resistant *if for all PPT adversary $\mathbb{A}$, for all parameters $n_T, t, n_V, n_A, n_C$, and for all distributions $\mathcal{D}$, there exists a PPT adversary $\mathbb{B}$ and a negligible function $\mu$ such that*

$$|\Pr(\mathsf{Ideal}^{\mathsf{CR}}_{\mathsf{JCJ}}(\mathbb{B}, \lambda, n_V, n_A, n_C, \mathcal{D}) = 1) - \Pr(\mathsf{Real}^{\mathsf{CR}}_{\mathsf{JCJ}}(\mathbb{A}, \lambda, n_T, t, n_V, n_A, n_C, \mathcal{D}) = 1)| \leq \mu(\lambda)\,.$$

As noticed in [17], this definition cannot be realized by a scheme which uses a public board. Indeed, in the real game, the adversary observes the length $n_B$ of the board which corresponds to the total number of ballots cast by non-corrupted voters. Then the adversary learns the result and in particular its size $n_v$, that is the number of valid ballots counted. Hence the total number $\Delta = n_B - n_v$ of ballots discarded can be deduced, which is not available in the ideal game $\mathsf{Ideal}^{\mathsf{CR}}_{\mathsf{JCJ}}$. The value of $\Delta$ can be compared with its expected number, according to the distribution $\mathcal{D}$. Since there is an additional ballot discarded (the one of the coercer) when the voter evades coercion, the adversary has a non-negligible advantage in the real game. For instance, if $\mathcal{D}$ is such that no voters cast a ballot with an invalid credential, either $n_B = n_v$, which means that the adversary's ballot has been counted, or $n_B = n_v + 1$, meaning that the adversary's ballot has been discarded and that the voter has disobeyed. Of course, the same issue applies to the JCJ definition as stated in [22].

The authors of [17] proposed a patch to the issue they discovered: the length of the board should be given to the adversary in the ideal game as well. Intuitively, this corresponds to simply rewriting line 18 of Algorithm 2 as $(\nu_i)_{i \in V}, \alpha \longleftarrow \mathbb{A}(|D|)$. However, this still does not allow to detect the leakage of the JCJ protocol during the tally. Indeed, the distribution $\mathcal{D}$ fails to model several aspects:

- First, the addition of a ballot with an invalid credential only happens when a honest voter sacrifices their own vote. This is unlikely in practice, and does not model ballots sent by non-eligible voters (for instance, by the authorities).
- Second, revoting is not considered in $\mathcal{D}$, which explains why the leakage of the JCJ protocol was not detected.

A final remark about the definition of JCJ concerns its underlying trust assumptions. For clarity, we recall them here. First of all, it is assumed that all the registrars are honest and that the adversary is inactive during the registration phase (or, alternatively, the registration is untapable). Second, the adversary can only corrupt a minority of decryption authorities. Also, ballots are cast through anonymous channels. Finally, the bulletin board is honest.

### C. The quantitative definition of KTV

Apart from the definition of JCJ, there are other definitions in the literature (see [17] for a survey). The most prominent one is the quantitative definition of [25] (KTV), where the notion of $\delta$-coercion-resistance comes with two conditions: first, the coerced voter must have a strategy to meet their objective with overwhelming probability; second, the adversary cannot decide, with an advantage greater than $\delta$, whether the voter used this strategy or forwarded all received messages (including their credential).

The KTV definition is abstract. To use it, it is necessary to model the voting protocol, its participants and the evasion strategy. In addition, it does not say much about how ballots sent with an invalid credential should be handled since the honest participants are assumed to vote following a fixed distribution of valid voting options. Finally, it does not tell if a specific $\delta$ is acceptable or not. Our definition can be seen as an instance of KTV, where $\delta$ is shown to be minimal, that is, not greater than that of an ideal protocol.

### D. Our definition of coercion-resistance

If we compare the advantage of the adversary in the real game with its advantage in the ideal one, we need to cover a large family of vote distributions. Otherwise, we may miss security flaws. In particular, we need to cover cases explicitly planned by the protocol such as revote and addition of ballots with fake credentials.

Therefore, given a set $S$ of unique identifiers and the number $n_C$ of voting options (excluding abstention), we consider a distribution $\mathcal{B}(S, n_C)$ of sequences of pairs of the form $(j, \nu)$ where $\nu \in [1, n_c]$ represents a vote and $j$ represents either a valid voter (when $j \in S$) or a fake voter, with a fake credential. Typically, if $A$ is the set of corrupted voters, $S = [1, n_V] \backslash A$. To avoid collisions with identifiers which may be in $A$, we consider that any $i \notin S$ holds a negative value. The distribution $\mathcal{B}$ captures the abstention of a voter $i$ with the absence of a couple of the form $(i, *)$. It models both revoting and the addition of fake ballots, typically by authorities:

- revoting is reflected in $\mathcal{B}$ by the fact that a voter may appear several times in the same sequence;
- fake ballots are modeled by pairs $(j, \nu)$ where $j \notin S$. They may be added by authorities or voters. Note that $\mathcal{B}$ also models the case of a revote with a fake credential.

For example, in the sequence $(1, 1)$, $(2, 1)$, $(1, 2)$, $(-1, 2)$, $(1, 1)$ with $n_V = 3$, we have three voters $V_1$, $V_2$ and $V_3$. $V_1$ first votes 1, $V_2$ votes 1, then $V_1$ revotes for 2, then a fake vote for 2 is added, then $V_1$ changes back her vote to 1. $V_3$ simply chooses to abstain.

Our $\mathsf{Real}^{\mathsf{CR}}$ game, given in Algorithm 3, is similar to $\mathsf{Real}^{\mathsf{CR}}_{\mathsf{JCJ}}$. Votes are drawn according to $\mathcal{B}([1, n_V] \backslash A, n_C)$, yielding a sequence $B$. It typically contains pairs $(i, \nu)$ with $i < 0$, which corresponds to the addition of ballots with fake credentials. For such a pair, we therefore generate a fake credential at lines 11-12. Just as in the definition of JCJ, the adversary must guess a bit $b$. If $b = 1$, the coerced voter $j$ obeys, hence any vote from $j$ is removed from $B$ and the real credential is provided to the adversary. If $b = 0$, the voter follows the evasion strategy, namely they cast one vote for $\beta$ (if $\beta \neq \phi$) and provides a fake credential. Hence the votes from $j$ in $\mathcal{B}$ are replaced by a single vote for $\beta$ (if $\beta \neq \phi$). Then ballots are added according to $\mathcal{B}$. They correspond either to real or fake votes (or revotes). Compared to the original JCJ definition, we also slightly improve the power of the adversary by letting them observe the board after each vote and add ballots if they want to, which better reflects the reality.

Again, the advantage of the adversary in the real game is compared with its advantage in an ideal game $\mathsf{Ideal}^{\mathsf{CR}}$, where the adversary can only observe the number of ballots and the result (see Algorithm 4). We assume a function `cleanse` that removes votes from invalid voters $j \notin [1, n_V]$ and that takes care of revotes according to the policy (typically, the last vote is kept). The function `result`, given a set of valid votes, returns the result of the election.

**Definition 2.** *A voting system is* coercion resistant *if for all PPT adversary $\mathbb{A}$, for all parameters $n_T, t, n_V, n_A, n_C$, and for all distribution $\mathcal{B}$, there exists a PPT adversary $\mathbb{B}$ and a negligible function $\mu$ such that*

$$\begin{aligned} &|\Pr(\mathsf{Ideal}^{\mathsf{CR}}(\mathbb{B}, \lambda, n_V, n_A, n_C, \mathcal{B}) = 1) \\ &- \Pr(\mathsf{Real}^{\mathsf{CR}}(\mathbb{A}, \lambda, n_T, t, n_V, n_A, n_C, \mathcal{B}) = 1)| \leq \mu(\lambda)\,. \end{aligned}$$

The main difference between our definition and the original one is that we consider a larger family of distributions, which allows to analyze a protocol in the context of revotes and fake ballots.

Another difference is that the adversary shall not gain any advantage for *any* distribution $\mathcal{B}$, while the JCJ definition defines coercion-resistance *with respect to* a particular distribution. This is preferable since a protocol should be as secure as the ideal one, whatever the considered distribution. It is counter-intuitive to design a cryptographic protocol that resists only for particular distributions. Of course, it makes sense to analyze the exact advantage in the ideal game for a particular distribution, and devise whether voters are reasonably protected in that case of not. But the cryptographic protocol itself should be as solid as the ideal one nevertheless.

### E. CHide is coercion-resistant

Thanks to our cleansing procedure, we show that CHide achieves coercion-resistance. This is stated in Theorem 1, proven in Appendix C-B. In this theorem, we suppose secure DKG and Mixnet protocols in the SUC security framework [9],

**Algorithm 3:** $\mathsf{Real}^{\mathsf{CR}}$

**Require:** $\mathbb{A}, \lambda, n_T, t, n_V, n_A, n_C, \mathcal{B}$

$BB \longleftarrow \varnothing$
$\mathsf{pk}, (s_i, h_i)_{i=1}^{n_T} \longleftarrow P^{\mathbb{A}}_{\texttt{Setup}}(\lambda, n_T, t)$ (* run the DKG *)
$\{c_i; i \in [1, n_V]\}, \boldsymbol{R} \longleftarrow \texttt{register}(\lambda, \mathsf{pk}, n_V)$
$A \longleftarrow \mathbb{A}(\boldsymbol{R})$ (* corrupt voters *)
$(j, \alpha) \longleftarrow \mathbb{A}(\{c_i; i \in A\})$
(* coerce a voter $j$ who has the intention $\alpha$ *)
**if** $|A| \neq n_A \vee j \notin [1, n_V] \backslash A \vee \alpha \notin [1, n_C] \bigcup \{\phi\}$ **then**
  Return 0
$B \longleftarrow \mathcal{B}([1, n_V] \backslash A, n_C)$
(* samples a sequence of pairs $(i, \nu_i)$ with $i \in ([1, n_V] \backslash A) \bigcup \{n \in \mathbb{Z} \mid n < 0\}$ and $\nu_i \in [1, n_C]$ *)
**for** $(i, *) \in B$, $i \notin [1, n_V]$ **do**
  $c_i \longleftarrow \texttt{Fakecred}()$
  (* this captures ballots sent with invalid creds *)
$b \xleftarrow{\$} \{0, 1\}$
$\tilde{c} \longleftarrow c_j$
**if** $b == 1$ **then**
  Remove all $(j, *) \in B$
**else**
  Remove all $(j, *) \in B$ but the last, which is replaced by $(j, \alpha)$ if $\alpha \neq \phi$ and removed otherwise
  $\tilde{c} \longleftarrow \texttt{Fakecred}(c_j)$
$\mathbb{A}(\tilde{c})$ (* $\mathbb{A}$ learns $\tilde{c}$ *)
**for** $(i, \nu_i) \in B$ *(in this order)* **do**
  $\boldsymbol{M} \longleftarrow \mathbb{A}(BB)$ (* cast ballots *)
  $BB \longleftarrow BB \bigcup \{m \in \boldsymbol{M} \mid \texttt{isVal}(\mathsf{pk}, m, BB) = 1\}$
  $BB \leftarrow BB \bigcup \{\texttt{Vote}_{\mathsf{pk}}(c_i, \nu_i)\}$
$\boldsymbol{M} \longleftarrow \mathbb{A}(BB, \texttt{"last honest ballot sent"})$
$BB \longleftarrow BB \bigcup \{m \in \boldsymbol{M} \mid \texttt{isVal}(\mathsf{pk}, m, BB) = 1\}$
$\boldsymbol{X}, \Pi \leftarrow P^{\mathbb{A}}_{\texttt{tally}}(BB, \boldsymbol{R}, \mathsf{pk}, \{h_i, s_i\}, t)$
$b' \longleftarrow \mathbb{A}()$
Return 1 **if** $b' == b$ **else** 0

**Algorithm 4:** $\mathsf{Ideal}^{\mathsf{CR}}$

**Require:** $\mathbb{A}, \lambda, n_V, n_A, n_C, \mathcal{B}$

$A \longleftarrow \mathbb{A}(\lambda)$ (* corrupt voters *)
$(j, \alpha) \longleftarrow \mathbb{A}()$
(* coerce a voter $j$ who has the intention $\alpha$ *)
**if** $|A| \neq n_A \vee j \notin [1, n_V] \backslash A \vee \alpha \notin [1, n_C] \bigcup \{\phi\}$ **then**
  Return 0
$B \longleftarrow \mathcal{B}([1, n_V] \backslash A, n_C)$
(* samples a sequence of pairs $(i, \nu_i)$ with $i \in ([1, n_V] \backslash A) \bigcup \{n \in \mathbb{Z} \mid n < 0\}$ and $\nu_i \in [1, n_C]$ *)
$b \xleftarrow{\$} \{0, 1\}$
**if** $b == 1$ **then**
  Remove all $(j, *) \in B$
**else**
  Remove all $(j, *) \in B$ but the last, which is replaced by $(j, \alpha)$ if $\alpha \neq \phi$ and removed otherwise
$(\nu_i)_{i \in A}, \beta \longleftarrow \mathbb{A}(|B|)$
**if** $(b == 1) \wedge (\beta \in [1, n_C])$ **then**
  $B \longleftarrow B \bigcup \{(j, \beta)\}$
$B \longleftarrow B \bigcup \{(i, \nu_i) \mid i \in A, \nu_i \in [1, n_C]\}$
$\boldsymbol{X} \longleftarrow \texttt{result}(\texttt{cleanse}(B))$
$b' \longleftarrow \mathbb{A}(\boldsymbol{X})$
Return 1 **if** $b' == b$ **else** 0

Fig. 8: Definition of coercion-resistance. $\lambda$ is the security parameter, $n_T$ the number of talliers, $t$ the threshold, $n_V$ the number of voters, $n_A$ the number of corrupted voters, $n_C$ the number of voting options and $\mathcal{B}$ the distribution.

which is introduced in Appendix B. Simply speaking, it is a Universally Composable security framework which is suitable for MPC protocols. Examples of UC-secure DKG and decryption mixnet can be found in [34], [35].

We mention that we also prove privacy and verifiability in Appendices C-C and C-D, using a similar approach.

**Theorem 1.** *Under the DDH assumption, assuming SUC-secure DKG and Mixnet protocols and in the Programmable Random Oracle Model, CHide is coercion-resistant.*

We can check that JCJ is not coercion-resistant according to our definition. This is due to the leakage during the cleansing phase, as explained in Section II-B.

**Proposition 1.** *The JCJ protocol is not coercion-resistant.*

## VI. Discussion

While JCJ does not satisfy our definition of coercion-resistance, it does provide a certain level of security. As future work, we could identify the exact nature of the leakage in JCJ and propose an alternative (weaker) definition of coercion-resistance that is achieved by JCJ. Beyond giving a concrete description of the leakage, this would show that there are several shades of coercion-resistance depending on the leakage which is considered acceptable. More generally, for any variant of JCJ in the literature, if the leakage during the cleansing is different from the one in JCJ, our definition of coercion-resistance could (in principle) be adapted to capture this. Still, CHide shows that a better notion of coercion-resistance can be achieved.

## References


[1] R. Araújo, S. Foulle, and J. Traoré, "A practical and secure coercion-resistant scheme for remote elections," in *Frontiers of Electronic Voting*. IBFI, 2007.

[2] S. Baloglu, S. Bursuc, S. Mauw, and J. Pang, "Provably Improving Election Verifiability in Belenios," in *E-Vote-ID'21*. Springer, 2021.

[3] K. E. Batcher, "Sorting Networks and Their Applications," in *American Federation of Information Processing Societies, AFIPS'68*. Thomson Book Company, Washington D.C., 1968.

[4] M. Bellare and A. Sahai, "Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization," in *CRYPTO '99*. Springer, 1999.

[5] J. Benaloh, T. Moran, L. Naish, K. Ramchen, and V. Teague, "Shufflesum: coercion-resistant verifiable tallying for STV voting," *IEEE Trans. Inf. Forensics Secur.*, vol. 4, no. 4, pp. 685–698, 2009.

[6] D. Bernhard, O. Pereira, and B. Warinschi, "How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios," in *ASIACRYPT'12*. Springer, 2012.

[7] D. Boneh, E. Boyle, H. Corrigan-Gibbs, N. Gilboa, and Y. Ishai, "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs," in *CRYPTO'19*. Springer, 2019.

[8] R. Canetti, "Universally Composable Security: A New Paradigm for Cryptographic Protocols," in *42nd Annual Symposium on Foundations of Computer Science, FOCS'01*. IEEE, 2001.

[9] R. Canetti, A. Cohen, and Y. Lindell, "A Simpler Variant of Universally Composable Security for Standard Multiparty Computation," in *CRYPTO'15*. Springer, 2015.

[10] J. Clark and U. Hengartner, "Selections: Internet Voting with Over-the-Shoulder Coercion-Resistance," in *FC'11*. Springer, 2011.

[11] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a Secure Voting System," in *S&P'08*. IEEE, 2008.

[12] V. Cortier, D. Galindo, S. Glondu, and M. Izabachène, "Election Verifiability for Helios under Weaker Trust Assumptions," in *ESORICS'14*. Springer, 2014.

[13] V. Cortier, P. Gaudry, and Q. Yang, "A Toolbox for Verifiable Tally-Hiding E-Voting Systems," in *ESORICS'22*. Springer, 2022.

[14] ——, "A toolbox for verifiable tally-hiding e-voting systems," Cryptology ePrint Archive, Paper 2021/491, 2021, https://eprint.iacr.org/2021/491.

[15] David Mestel and Johannes Müller and Pascal Reisert, "How Efficient are Replay Attacks against Vote Privacy? A Formal Quantitative Analysis," in *CSF'22*. IEEE, 2022.

[16] P. Ehin, M. Solvak, J. Willemson, and P. Vinkel, "Internet voting in Estonia 2005-2019: Evidence from eleven elections," *Gov. Inf. Q.*, vol. 39, no. 4, p. 101718, 2022.

[17] T. Haines and B. Smyth, "Surveying definitions of coercion resistance," Cryptology ePrint Archive, Report 2019/822, 2019, https://ia.cr/2019/822.

[18] L. Hirschi, L. Schmid, and D. A. Basin, "Fixing the Achilles Heel of E-Voting: The Bulletin Board," in *CSF'21*. IEEE, 2021.

[19] N. Huber, R. Küsters, T. Krips, J. Liedtke, J. Müller, D. Rausch, P. Reisert, and A. Vogt, "Kryvos: Publicly tally-hiding verifiable e-voting," in *CCS'22*. ACM, 2022.

[20] M. Jakobsson and A. Juels, "Mix and Match: Secure Function Evaluation via Ciphertexts," in *ASIACRYPT'00*. Springer, 2000.

[21] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated Verifier Proofs and Their Applications," in *EUROCRYPT'96*. Springer, 1996.

[22] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES*, 2005.

[23] D. Knuth, *The Art Of Computer Programming, vol. 3: Sorting And Searching*. Addison-Wesley, 1973.

[24] R. Küsters, J. Liedtke, J. Müller, D. Rausch, and A. Vogt, "Ordinos: A Verifiable Tally-Hiding E-Voting System," in *IEEE European Symposium on Security and Privacy (EuroS&P'20)*. IEEE, 2020.

[25] R. Küsters, T. Truderung, and A. Vogt, "A Game-Based Definition of Coercion-Resistance and Its Applications," in *CSF'10*. IEEE, 2010.

[26] ——, "Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study," in *S&P'11*. IEEE, 2011.

[27] P. Locher, R. Haenni, and R. E. Koenig, "Coercion-Resistant Internet Voting with Everlasting Privacy," in *FC'16 International Workshops*. Springer, 2016.

[28] W. Lueks, I. Querejeta-Azurmendi, and C. Troncoso, "VoteAgain: A scalable coercion-resistant voting system," in *USENIX'20*. USENIX, 2020.

[29] Ü. Madise and T. Martens, "E-voting in Estonia 2005. The first Practice of Country-wide binding Internet Voting in the World," in *2nd International Workshop in Electronic Voting (EVOTE'06)*. GI, 2006.

[30] J. B. Nielsen, "On Protocol Security in the Cryptographic Model," Ph.D. dissertation, University of Aarhus, 2003.

[31] B. Schoenmakers and P. Tuyls, "Practical Two-Party Computation Based on the Conditional Gate," in *ASIACRYPT'04)*. Springer, 2004.

[32] O. Spycher, R. E. Koenig, R. Haenni, and M. Schläpfer, "A New Approach towards Coercion-Resistant Remote E-Voting in Linear Time," in *FC'11*. Springer, 2011.

[33] ——, "Achieving Meaningful Efficiency in Coercion-Resistant, Verifiable Internet Voting," in *5th International Conference on Electronic Voting, EVOTE'12*. GI, 2012.

[34] D. Wikström, "A Universally Composable Mix-Net," in *Theory of Cryptography Conference, TCC'04*, ser. Lecture Notes in Computer Science. Springer, 2004.

[35] ——, "Universally Composable DKG with Linear Number of Exponentiations," in *Security in Communication Networks (SCN)'04*. Springer, 2004.

## APPENDIX A
## FULL SPECIFICATION OF $P_{\text{TALLY}}$

In this appendix, we give every details that are necessary to understand the tally protocol. First of all, the CGate protocol [31], built upon the exponential ElGamal encryption system and its threshold decryption protocol, is given in Algorithm 5, where we used the adapted version of [14] (which is the full version of [13]), for which there is a security proof in the SUC framework. The CGate protocol enables to perform multiple functionalities, such as computing a logical conjunction And of two encrypted bits and conditionally set an encrypted value to 0, given an encrypted boolean (we denote this functionality CSZ). Note that in addition to the output $Z$, a transcript $\Pi^{\mathsf{CGate}}$ is obtained, which allows to verify that the participants performed the correct operations on $X$ and $Y$, and that $Z$ is indeed an encryption of $xy$.

**Algorithm 5:** CGate (see [13]).
The total cost is $33n_T$ exponentiations per participants, and a transcript size of $34n_T$ elements of 256 bits.

**Require:** $G$, a group of prime order $q$ and public generator $g$
pk, an exponential ElGamal public key,
whose shares are distributed among the $n_T$ participants
$\tilde{h} \in G$, a public element independent from pk
$X$, an encryption of some $x \in \mathbb{Z}_q$
$Y$, an encryption of $y \in \{0, 1\}$
**Ensure:** $Z$, a random encryption of $xy$

$Y_0 \longleftarrow E_{-1}Y^2$; $X_0 \longleftarrow X$
**for** $i = 1$ *to* $n_T$*, for the authority* $i$*,* **do**
  $(u, v) \longleftarrow X_{i-1}$
  $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q$; $s \xleftarrow{\$} \{-1, 1\}$
  $X_i \longleftarrow (u^s g^{r_1}, v^s \mathsf{pk}^{r_1})$; $e \longleftarrow \tilde{h}^{r_1}$
  $Y_i \longleftarrow \mathsf{ReEnc}_{\mathsf{pk}}(Y_{i-1}^s, r_2)$
  Broadcast $X_i, e, Y_i$ and a ZKP $\pi_i$ that they are well formed
We denote $\Pi = (X_{n_T}, Y_{n_T}, \pi_{n_T})||\cdots||(X_1, Y_1, \pi_1)$
Each authority verifies the proof of the other authorities
They collectively rerandomize $X_{n_T}$ and $Y_{n_T}$ into $X'$ and $Y'$, which produces a transcript $\Pi^{\mathsf{ReEnc}}$
They collectively decrypt $Y'$ into $y_{n_T}$, which produces a transcript $\Pi^{\mathsf{Dec}}$.
They output $Z = (XX'^{y_{n_T}})^{\frac{1}{2}}$. The verification transcript is $\Pi^{\mathsf{CGate}} = (y_{n_T}, \Pi^{\mathsf{Dec}})||(X', Y', \Pi^{\mathsf{ReEnc}})||\Pi$

As shown in [31], due to the homomorphic property of the ElGamal cryptosystem, we can readily deduce protocols for computing a disjunction (see Algorithm 6), and an equality test (Algorithm 7) between two bits. In addition, the logical negation of an encrypted bit $B$ $\mathsf{Not}(B)$ can be obtained without interaction by computing $\mathsf{Enc}(1)/B$ (for this purpose, a fixed encryption of 1 with a trivial randomness can be used).

**Algorithm 6:** Or

**Require:** $X, Y$ such that $X$ (resp. $Y$) is an encryption of $x$ (resp. $y$), with $x, y \in \{0, 1\}$
**Ensure:** $Z = \mathsf{Enc}(z)$, with $z = 1$ if $x = 1$ or $y = 1$; 0 otherwise

Return $XY/\mathsf{CGate}(X, Y)$ and $\Pi^{\mathsf{CGate}}$

**Algorithm 7:** Eq

**Require:** $X, Y$ such that $X$ (resp. $Y$) is an encryption of $x$ (resp. $y$), with $x, y \in \{0, 1\}$
**Ensure:** $Z = \mathsf{Enc}(z)$, with $z = 1$ if $x = y$; 0 otherwise

Return $\mathsf{Enc}(1)\mathsf{CGate}(X, Y)^2/(XY)$ and $\Pi^{\mathsf{Eq}} = \Pi^{\mathsf{CGate}}$

Now, since the logical disjunction (resp. conjunction) operator is associative, we immediately deduce a protocol to extend this binary operator into a $n$-ary operator (by abuse of notation, we still denote it Or; resp. And). By using a binary tree structure, the protocol $\mathsf{Or}(B_1, \cdots, B_n)$ (resp. $\mathsf{And}(B_1, \cdots, B_n)$); which returns an encryption of 1 if one (resp. all) of the inputs is an encryption of 1, and 0 otherwise; can be performed in $O(\log n)$ rounds and a total of $n - 1$ instances of CGate.

We can also extend the Eq protocol to bitwise encrypted integers (see Algorithm 8).

By combining logical operations, it is straightforward to build a comparison protocol Lt (see Algorithm 9 from [13]) which, given two bitwise encryptions of the integers $x$ and $y$, returns an encryption of 1 if $x < y$ and an encryption of 0 otherwise. We also give the conditional swap protocol in Algorithm 10. By combining those two protocols, we can readily adapt the OddEvenMergeSort algorithm in MPC (see Algorithm 11, adapted from [23, Section 5.2.2, Algorithm M]). This sorting algorithm requires approximately $\frac{1}{4}n\log(n)^2$ comparisons and conditional swaps, where $n$ is the number of elements

**Algorithm 8:** Eq (for bitwise encrypted integers)

**Require:** $X_0, \cdots, X_m$, $Y_0, \cdots, Y_m$ bit-wise encryptions of $x$ and $y$.
**Ensure:** $Z$, an encryption of 1 if $x = y$, 0 otherwise.

**for** *all $i$ (in parallel)* **do**
  $A_i \longleftarrow \mathsf{Eq}(X_i, Y_i)$
Return $Z = \mathsf{And}(A_0, \cdots, A_{m-1})$ and the concatenation of all the verification transcript

to be sorted. In the $P_{\texttt{tally}}$ protocol, if $\lambda$ is the security parameter, $n_B$ the number of valid ballots received and $n_V$ the number of eligible voters, then $n = n_B + n_V$ and performing a comparison and a conditional swap costs about $3(\lambda + m)$ CGate in total, where $m = \lceil \log(n_B) \rceil + 1$. If $n_T$ is the number of talliers, this translates into $O((\lambda + m) n_T (n_B + n_V) \log(n_B + n_V)^2)$ exponentiations.

**Algorithm 9:** Lt

**Require:** $(X_0, \cdots, X_{m-1})$, $(Y_0, \cdots, Y_{m-1})$, bit-wise encryptions of $x$ and $y$.
**Ensure:** $R$, an encryption of 1 if $x < y$ and 0 otherwise.

$A \longleftarrow \mathsf{CGate}(X_0, Y_0)$
$Z_0 \longleftarrow X_0 Y_0 / A^2$ (* $x_0 \oplus y_0$ *)
$R \longleftarrow Y_0 / A$ (* $y_0 \wedge \neg x_0$ *)
**for** $k = 1$ *to* $m - 1$ **do**
  $A \longleftarrow \mathsf{CGate}(Y_k, R)$
  $B \longleftarrow Y_k R / A^2$ (* $y_k \oplus r$ *)
  $C \longleftarrow \mathsf{CGate}(X_k, B)$
  $Z_k \longleftarrow X_k B / C^2$ (* $x_k \oplus y_k \oplus r$ *)
  $R \longleftarrow Y_k R / (AC)$ (* $(y_k \wedge r) \vee [(y_k \vee r) \wedge \neg x_k]$ *)
Return $R$ and the concatenation of all the verification transcripts

**Algorithm 10:** CSwap

**Require:** $X, Y, B$, encryptions of $x$, $y$ and $b$ with $b \in \{0, 1\}$
**Ensure:** $X', Y'$, reencryptions of $y$ and $x$ if $b = 1$; of $x$ and $y$ otherwise

$A \longleftarrow \mathsf{CGate}(Y/X, B)$; $X' \longleftarrow XA$; $Y' \longleftarrow Y/A$
Return $X', Y'$ and $\Pi^{\mathsf{CSwap}} = \Pi^{\mathsf{CGate}}$

**Algorithm 11:** OddEvenMergeSort

**Require:** $(V_i, K_i)_{i=0}^{n-1}$, where, for all $i$, $V_i$ is an encryption while $K_i$ is a bitwise encryption of an integer $k_i$
**Ensure:** $(V'_i, K'_i)_{i=0}^{n-1}$, reencryptions of the same values, but sorted with increasing $k_i$

$t \longleftarrow \lceil \log n \rceil$; $p \longleftarrow 2^{t-1}$; $\Pi^{\mathsf{Sort}} \longleftarrow \varnothing$
**while** $p > 0$ **do**
  $q \longleftarrow 2^{t-1}$; $r \longleftarrow 0$; $d \longleftarrow p$
  **while** $d > 0$ **do**
    **for** $i = 0$ *to* $n - d - 1$ *(in parallel)* **do**
      **if** $\mathsf{BitwiseAnd}(i, p) = r$ **then**
        $B, \Pi^{\mathsf{Lt}} \longleftarrow \mathsf{Lt}(K_{i+d}, K_i)$; $\Pi^{\mathsf{Sort}} \longleftarrow \Pi^{\mathsf{Sort}} || \Pi^{\mathsf{Lt}}$
        $V_i, V_{i+d}, \Pi^{\mathsf{CSwap}} \longleftarrow \mathsf{CSwap}(V_i, V_{i+d}, B)$; $\Pi^{\mathsf{Sort}} \longleftarrow \Pi^{\mathsf{Sort}} || \Pi^{\mathsf{CSwap}}$
        $K_i, K_{i+d}, \Pi^{\mathsf{CSwap}} \longleftarrow \mathsf{CSwap}(K_i, K_{i+d}, B)$; $\Pi^{\mathsf{Sort}} \longleftarrow \Pi^{\mathsf{Sort}} || \Pi^{\mathsf{CSwap}}$ (* bit by bit, in parallel *)
    $d \longleftarrow q - p$; $q \longleftarrow \lfloor q/2 \rfloor$; $r \longleftarrow p$
  $p \longleftarrow \lfloor p/2 \rfloor$
Return $(V_i, K_i)_{i=0}^{n-1}$ and the transcript $\Pi^{\mathsf{Sort}}$

With all the above protocols, $P_{\texttt{tally}}$ can be written as Algorithm 12, where $BB$ is the ballot box; $\boldsymbol{R}$ is the public roster that contains the encrypted credentials.

**Algorithm 12:** $P_{\texttt{tally}}$

**Require:** *A $t$-threshold setup phase is assumed to have been run between the $n_T$ participants.*

**Require:** $BB, \boldsymbol{R}$

$B = \{(C_1, C_2) \mid (C_1, C_2, \pi_1, \pi_2) \in BB, \texttt{isVal}(\mathsf{pk}, \mathbf{b}, BB) = 1\}$

We denote $B = (C_1^i, C_2^i)_{i=1}^{n_B}$

We denote $\boldsymbol{R} = R_1, \cdots, R_{n_V}$

$\Pi \longleftarrow \varnothing$; $m \longleftarrow \lceil \log n_B \rceil + 1$

For $k = 0$ to $n_B$, we denote $U_k$ a bitwise encryption of $k$ with $m$ bits (least significant bits first)

(In other words, if $k = \sum_{i=0}^{m-1} b_{i,k} 2^i$, $U_k = \mathsf{Enc}(b_{0,k}), \cdots, \mathsf{Enc}(b_{m-1,k})$)

**for** $k = 1$ *to* $n_B$ **do**

  $V_k \longleftarrow C_1^k$; $K_k \longleftarrow U_{k-1} || C_2^k$

**for** $k = n_B + 1$ *to* $n_B + n_V$ **do**

  $V_k \longleftarrow \mathsf{Enc}(0)$; $K_k \longleftarrow U_{n_B} || R_{k-n_B}$

$(V'_k, K'_k)_{k=1}^{n_B+n_V}, \Pi^{\mathsf{Sort}} \longleftarrow \mathsf{OddEvenMergeSort}\left((V_k, K_k)_{k=1}^{n_B+n_V}\right)$

$\Pi \longleftarrow \Pi || \Pi^{\mathsf{Sort}}$

**for** $k = 1$ *to* $n_B + n_V - 1$ *(in parallel)* **do**

  $D_k, \Pi_1^{\mathsf{Eq}} \longleftarrow \mathsf{Eq}\left({K'}_k^{\top}, {K'}_{k+1}^{\top}\right)$ (* compare the $\lambda$ most significant bits *)

  $F_k, \Pi_2^{\mathsf{Eq}} \longleftarrow \mathsf{Eq}\left({K'}_{k+1}^{\perp}, U_{n_B}\right)$ (* compare the $m$ least significant bits *)

  $\Pi \longleftarrow \Pi || \Pi_1^{\mathsf{Eq}} || \Pi_2^{\mathsf{Eq}}$

  $G_k, \Pi^{\mathsf{And}} \longleftarrow \mathsf{And}(D_k, F_k)$; $\Pi \longleftarrow \Pi || \Pi^{\mathsf{And}}$

  $V'_k, \Pi^{\mathsf{CGate}} \longleftarrow \mathsf{CGate}(V'_k, G_k)$; $\Pi \longleftarrow \Pi || \Pi^{\mathsf{CGate}}$

$\boldsymbol{X}, \Pi^{\mathsf{DecMixnet}} \longleftarrow \mathsf{DecMixnet}({V'}_1, \cdots, {V'}_{n_B+n_V-1})$

$\Pi \longleftarrow \Pi || \Pi^{\mathsf{DecMixnet}}$

Return $\boldsymbol{X}, \Pi$

## APPENDIX B
## THE SUC SECURITY OF $P_{\texttt{TALLY}}$

In our security proofs, we use the SUC framework [9] which is a simpler variant of the Universally Composable (UC) framework [8]. In this appendix, we give a brief introduction to the SUC framework and prove that $P_{\texttt{tally}}$ is SUC-secure.

### A. Definitions, notations and composition

In the SUC framework, a protocol is a collection of participants, which are modeled as Interactive Turing Machines (ITM). As the name implies, the ITMs communicate with each others using common tapes. An execution of a protocol is called a *process*, during which the participants are executed concurrently. In turn, a process can invoke sub-processes, but with the same participants.

Some of the participants may be impersonated by an adversary $\mathbb{A}$ which also has a full control over the communication network: it can read, block and deliver the messages as it sees fit, which includes changing the order of two messages. However, the adversary cannot change (or forge) the content nor the recipient of a message, nor can it replay a message which was already delivered. In a sense, this is the same as considering strongly authenticated (but insecure) channels with enforced integrity. In addition to $\mathbb{A}$, there is a second adversarial ITM $\mathcal{Z}$ which represents the environment. It may only interact with $\mathbb{A}$, and can output either 0 or 1.

The security of a process is assessed by a comparison with an ideal process, where a trusted party computes the output of the process with the inputs of the participants. Intuitively, a protocol is SUC-secure if, for all adversary $\mathbb{A}$ in the real process, there exists a simulator $\mathcal{S}$ in the ideal process such that for all environment $\mathcal{Z}$, the latter cannot tell whether it interacts with $\mathbb{A}$ in the real process or $\mathcal{S}$ in the ideal process.

*1) The real and ideal processes:* The real process is mostly determined by the adversary and the protocol, which is the specification of every participants. Conversely, the ideal process is mostly determined by the simulator and the trusted party. If $\mathcal{F}$ is a functionality, the trusted party which computes $\mathcal{F}$ from the inputs of the participants is denoted $T_{\mathcal{F}}$. In the ideal process, it is assumed that the simulator (but not the environment) can interact with $T_{\mathcal{F}}$, which may react in various manners depending on the functionality. Typically, if the adversary corrupts so many participants that it becomes impossible for the remaining honest ones to run the protocol successfully, it is reasonable to expect that the protocol aborts and output $\perp$. In this case, in the ideal process, the simulator is given the possibility to have the trusted party abort and output $\perp$. Another quirk of the ideal process is that the simulator and the trusted parties know which participants are corrupted.

In both the ideal and real processes, the environment is given an arbitrary auxiliary input $z \in \{0,1\}^*$ and chooses the inputs of all the participants. When the process terminates, $\mathcal{Z}$ can also read the output of all the participants (which is the output of the trusted party in the ideal process).

*2) The hybrid process:* In the hybrid process, we consider an ideal functionality $\mathcal{G}$ and the corresponding trusted party $T_{\mathcal{G}}$. In addition to regular operations, we suppose that the participants can send (secure) queries to $T_{\mathcal{G}}$ which answers according to its specification. (For instance, computing the mean of several values can only be done when all values are known, so that $T_{\mathcal{G}}$ may have to wait until sufficiently many inputs are sent.) Contrary to the other communication channels, the one which connects the participants to $T_{\mathcal{G}}$ in the hybrid model is secure, meaning that the adversary cannot read the queries nor their answers (except if the participant communicating with $T_{\mathcal{G}}$ is corrupted). However, it can still block or delay the messages.

In this article, we denote $\mathsf{Real}^{\mathsf{SUC},\mathcal{G}}_{P,\mathbb{A},\mathcal{Z}}(\lambda, z)$ (resp. $\mathsf{Ideal}^{\mathsf{SUC}}_{T_{\mathcal{F}},\mathcal{S},\mathcal{Z}}(\lambda, z)$) the probability that $\mathcal{Z}$ outputs 1, given the auxiliary input $z$, when interacting with $\mathbb{A}$ in the $\mathcal{G}$-hybrid process (resp. with $\mathcal{S}$ which interacts with $T_{\mathcal{F}}$ in the ideal process), where $\lambda$ is the security parameter. We can now give the definition of SUC-security.

**Definition 3** ( [9])**.** *Let $P$ be a protocol and $T_{\mathcal{F}}$ and $T_{\mathcal{G}}$ two trusted parties, which computes the functionalities $\mathcal{F}$ and $\mathcal{G}$ respectively. We say that $P$ SUC-securely computes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model if, for all PPT real adversary $\mathbb{A}$, there exists a PPT ideal simulator $\mathcal{S}$ such that for every PPT environment $\mathcal{Z}$ and every constant $d \in \mathbb{N}$, there exists a negligible function $\mu$ such that every security parameter $\lambda \in \mathbb{N}$ and every auxiliary input $z \in \{0,1\}^{\lambda^d}$,*

$$\left|\mathsf{Real}^{\mathsf{SUC},\mathcal{G}}_{P,\mathbb{A},\mathcal{Z}}(\lambda, z) - \mathsf{Ideal}^{\mathsf{SUC}}_{T_{\mathcal{F}},\mathcal{S},\mathcal{Z}}(\lambda, z)\right| \leq \mu(\lambda).$$

*3) The composition theorem:* UC frameworks are extremely powerful due to their composition theorem, which allow to deduce the security of a protocol from the security of its sub-protocols. More precisely, suppose that $P$ is a protocol to compute $\mathcal{F}$ in the $\mathcal{G}$-hybrid model. Suppose, in addition, that there exists a protocol $Q$ to compute $\mathcal{G}$ in the $\mathcal{H}$-hybrid model. Then we can consider the protocol $P^Q$ in the $\mathcal{H}$-hybrid model, where every query to $T_{\mathcal{G}}$ is replaced by an instance of $Q$. Given this transformation, the composition theorem can be written as follows.

**Theorem 2** ( [9])**.** *Let $P$ be a protocol that SUC-securely computes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model and $Q$ a protocol that SUC-securely computes $T_{\mathcal{G}}$ in the $\mathcal{H}$-hybrid model. Then $P^Q$, as defined above, SUC-securely computes $\mathcal{F}$ in the $\mathcal{H}$-hybrid model.*

### B. Application to CHide

To assert the security of CHide, we use the SUC framework. For this purpose, we consider a static and malicious adversary which can fully impersonate the corrupted parties, but can only corrupt a fixed set of participants, which is decided at the beginning of the protocol. (We also consider that this set's cardinal may not exceed the threshold.) We consider a fixed number of participants with a fixed threshold (participants cannot leave or arrive and the threshold may not change). We use the so-called programmable random oracle model which is modeled as an ideal functionality $\mathcal{F}_{\mathcal{RO}}$, and we also suppose that the DKG and the decryption mixnet are SUC-secure. We denote $\mathcal{F}_{\mathsf{DKG}}$ and $\mathcal{F}_{\mathsf{DecMixnet}}$ the corresponding ideal functionalities. For more details, including a construction to obtain a UC-secure protocol for those, see for instance the work of Wikström [34], [35].

Hence, we place ourselves in the $\mathcal{G}$-hybrid model, where $\mathcal{G} = (\mathcal{F}_{\mathcal{RO}}, \mathcal{F}_{\mathsf{DecMixnet}})$.

Following [13], we restrict ourselves to environment which only gives as input "valid" bulletin boards; *i.e.* the ballots are of the form $(C_1, C_2, \pi_1, \pi_2)$ with valid ZKP $\pi_1, \pi_2$ as defined in Section IV, where $C_1$ is a valid ElGamal encryption (that is, any pair of group elements) and $C_2$ $\lambda$ valid ElGamal encryptions of bits. In his PhD thesis [30, Section 3.5], Nielsen proves that the composition theorem still holds if the inputs chosen by the environment are restricted to some decidable language. This allows to use [13]'s result, where the $\mathsf{CGate}$ protocol is shown to SUC-securely computing $T_{\mathsf{CSZ}}$ in the $\mathcal{G}$-hybrid model, where $\mathsf{CSZ}$ is the Conditional Set Zero functionality (*i.e.* $\mathsf{CSZ}(A, B)$, for any ciphertext $A$ and any encryption $B$ of 0 or 1, returns a random reencryption of $A$ if $B$ is an encryption of 1 and a random encryption of 0 otherwise). We can now state Theorem 3, which is a direct consequence of the composition theorem. In this theorem, we consider $T_{\texttt{tally}}$, the trusted party that computes the result of the election protocol from the public board and the inputs of all the parties, and returns the result as well as the output of all the conditional gate protocols.

**Theorem 3.** *Under the DDH assumption, $P_{\texttt{tally}}$ SUC-securely computes $T_{\texttt{tally}}$ in the $\mathcal{G}$-hybrid model.*

*Proof sketch.* As shown in [13], $\mathsf{CGate}$ SUC-securely computes $T_{\mathsf{CSZ}}$ in the $\mathcal{G}$-hybrid model. Consequently, by the composition theorem, it is sufficient to show that $Q = P_{\texttt{tally}}^{T_{\mathsf{CSZ}}, T_{\mathsf{DecMixnet}}}$ SUC-securely computes $T\texttt{tally}$ in the $(\mathcal{G}, \mathcal{F}_{\mathsf{CSZ}})$-hybrid model, where $Q$ is $P_{\texttt{tally}}$ where all the instances of $\mathsf{CGate}$ are replaced by a query to $T_{\mathsf{CSZ}}$ while the instance of the decryption mixnet protocol is replaced by a query to $T_{\mathsf{DecMixnet}}$. In other words, it is sufficient to show that, given the ideal result computed by $T_{\texttt{tally}}$, the simulator can simulate the output of all the trusted parties, which comes immediately since it can use the result for the output of the decryption mixnet, and that it has the output of the conditional gate protocols from $T_{\texttt{tally}}$. □

## APPENDIX C
## SECURITY PROOFS FOR CHIDE

### A. Preliminary: on the IND-PA0 game

To prove coercion-resistance, we use a reduction to the IND-PA0 security of the encryption scheme. This security notion is defined, for instance, in [4], where it is shown to be equivalent to NM-CPA. In this section, we recall the definition of IND-PA0 and formalize the voting protocol as an IND-PA0 encryption scheme.

First, using the notations of Bellare and Sahai, let $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. We say that it is IND-PA0 if, for all (stateful) PPT adversary $\mathbb{A}$, its advantage in the game depicted in Algorithm 13 is negligible in the security parameter $\lambda$. Note that at line 5, the adversary can send a list of several ciphertexts, and gets the element-wise decryptions of those ciphertexts at the following line.

Now, let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be the ElGamal encryption scheme, which is known to be IND-CPA secure under DDH assumption. For simplicity, we assume that any voting option can be encoded as a single group element. For any $\lambda$-bits credential $c = c_1, \cdots, c_\lambda$ and voting option $\nu$, recall that $\texttt{Vote}_{\mathsf{pk}}(c, \nu)$ is obtained by computing the encryptions $C_1 = \mathsf{Enc}_{\mathsf{pk}}(\nu)$ and $C_2 = (\mathsf{Enc}_{\mathsf{pk}}(g^{c_1}), \cdots, \mathsf{Enc}_{\mathsf{pk}}(g^{c_\lambda}))$, where $g$ is a public group element. Then, a NIZKPoK $\pi_1$ of the plaintexts $(c, \nu)$ is computed using the Fiat-Shamir heuristic. To make sure that the ballot remains non-malleable, all the public informations such as the public encryption key, $g$, $C_1$ and $C_2$ are included into the hash to compute the challenge from the commitment of the $\Sigma$-protocol. Finally, a ZKP $\pi_2$ that $(c, \nu)$ is valid is computed and the ballot $(C_1, C_2, \pi_1, \pi_2)$ is formed. Clearly, $\texttt{Vote}_{\mathsf{pk}}(c, \nu)$ can be seen as an encryption of the tuple $(c, \nu)$ by the encryption scheme $(\mathsf{Gen}, \texttt{Vote}, \mathsf{Extract})$, where the $\mathsf{Extract}$ algorithm first checks the ZKPs and outputs $\perp$ if any of the proof is invalid. Otherwise, it decrypts all the ciphertexts with the secret key and returns the tuple $(c, \nu)$. By a result of [6], this defines an NM-CPA encryption scheme, which is therefore IND-PA0 secure. Note that $\pi_2$ is not necessary for the IND-PA9 security but does not deteriorate it either. It is useful for other properties of the protocol. In the following section, whenever we use the IND-PA0 security, we refer to the security of this encryption scheme whose plaintext (resp. ciphertext) space consists of $(\lambda+1)$-tuples of plaintexts (resp. ciphertexts, along with the corresponding ZKP).

**Algorithm 13:** IND-PA0 game

1 $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathcal{K}(\lambda)$
2 $(x_0, x_1) \longleftarrow \mathbb{A}(\mathsf{pk})$
3 $b \xleftarrow{\$} \{0,1\}$
4 $y \longleftarrow \mathcal{E}_{\mathsf{pk}}(x_b)$
5 $\boldsymbol{c} \longleftarrow \mathbb{A}(y)$
6 $\boldsymbol{p} \longleftarrow \mathcal{D}_{\mathsf{sk}}(\boldsymbol{c})$
7 $b' \longleftarrow \mathbb{A}(\boldsymbol{p})$
8 Return 1 if and only if $(y \notin \boldsymbol{c}) \wedge (b' = b)$

### *B. Proof of coercion-resistance*

The definition of coercion-resistance is given in Definition 2. In this section, we prove Theorem 1, that we restate below. For this purpose, we give a succession of games such that Game 0 is the real game and Game 10 is the ideal game. We consider a PPT $\mathbb{A}_0$ for Game 0. For Game $i$, we construct a PPT adversary $\mathbb{A}_i$ for this game and we denote $S_i$ the probability that $\mathbb{A}_i$ wins this game. (To ease the notation, we drop the dependency in $\lambda$ when the context is clear.) For all $i$, we show that $|S_{i+1} - S_i|$ is negligible, which proves that $|S_0 - S_9|$ is also negligible.

**Theorem 1.** *Under the DDH assumption, assuming SUC-secure DKG and Mixnet protocols and in the Programmable Random Oracle Model, CHide is coercion-resistant.*

*Proof.* **Game 1:** In this game, the adversary no longer takes part into the tally process at line 28. Instead, it is given the output of $T_{\mathtt{tally}}$.

By Theorem 3, there exists a simulator $\mathcal{S}$ such that, for all environment $\mathcal{Z}$, $|\mathsf{Real}^{\mathsf{SUC},\mathcal{G}}_{P_{\mathtt{tally}},\mathbb{A}_0,\mathcal{Z}}(\lambda, z)(\lambda, 0) - \mathsf{Ideal}^{\mathsf{SUC}}_{T_{\mathtt{tally}},\mathcal{S},\mathcal{Z}}(\lambda, 0)|$ is negligible. In particular, we consider the environment which defines Game 0, so that $\mathsf{Real}^{\mathsf{SUC},\mathcal{G}}_{P_{\mathtt{tally}},\mathbb{A}_0,\mathcal{Z}}(\lambda, z)(\lambda, 0) = S_0$. Then, $\mathbb{A}_1$ can interact with $\mathbb{A}_0$ by simulating the real game using $\mathcal{S}$, so that $\mathsf{Ideal}^{\mathsf{SUC}}_{T_{\mathtt{tally}},\mathcal{S},\mathcal{Z}}(\lambda, 0) = S_1$. Hence, $|S_1 - S_0|$ is negligible.

**Game 2:** In this game, we remove line 2. Instead, we generate a perfectly random public encryption key $\mathsf{pk}$ and give it to the adversary at line 4. Just as in the previous transition, assuming an ideal DKG protocol, there exists a negligible function $\mu_2$ and an adversary $\mathbb{A}_2$ for Game 2 such that $|S_2 - S_1| \leq \mu_2$.

**Game 3:** In this game, the adversary is no longer given the output of the conditional gates, that we denote $\Pi^Z$.

We construct $\mathbb{A}_3$ that interacts with $\mathbb{A}_2$ by simulating $\Pi^Z$. For this purpose, $\mathbb{A}_3$ uses uniformly random ciphertexts.

To argue the validity of this transition, we construct an adversary $\mathbb{B}$ for DDH as follows. First, $\mathbb{B}$ gets the challenge tuple $(g, g_2, g_3, g_4)$ from the DDH game and sets $\mathsf{pk} = g_2$. To run the setup, $\mathbb{B}$ recovers the set $S$ of the corrupted participants from $\mathbb{A}_2$, and picks $s_i \in \mathbb{Z}_q$ at random for all $i \in S$. It the completes $S$ into $I$ by picking some additional $s_i \in \mathbb{Z}_q$ at random for all $i \in I \backslash S$, where $I \subset [1, n_T]$ is a set of size $t$ that contains $S$, and $n_T$ is the number of talliers. For $i \in I$, it computes $h_i = g_1^{s_i}$ and, for $i \in [1, n_T] \backslash I$, it deduces $h_i$ with Lagrange interpolation.

It then runs the remaining of Game 3honestly, but each time $\mathbb{A}_2$ casts a ballot, $\mathbb{B}$ extracts the corresponding voting option from $\mathbb{A}_2$'s proof of knowledge. In the ROM, this is possible in polynomial time, as a consequence of the forking lemma (see for instance Theorem [6, Theorem 1]). This way, $\mathbb{B}$ can compute the result $r$ of the tally without knowing the secret key $\mathsf{sk}$. Finally, since $\mathbb{B}$ knows the cleartexts of the ballots to tally, $\mathbb{B}$ can run the tally protocol "on the cleartexts", *i.e.* it can compute the cleartext of each of the outputs of each conditional gate, as the product of two cleartexts. To simulate the output of a conditional gate, $\mathbb{B}$ "encrypts" the corresponding cleartext $z$ by choosing two random $\rho_1, \rho_2 \in \mathbb{Z}_q$ and computing $Z = (g^{\rho_1} g_3^{\rho_2}, g^z g_2^{\rho_1} g_4^{\rho_2})$. Finally, if $\mathbb{A}_2$ wins the game, $\mathbb{B}$ states that the challenge was a DDH tuple; otherwise, it states that is was a random tuple. Remark that if $(g, g_2, g_3, g_4)$ is a DDH tuple, then $\mathbb{B}$ played a perfect simulation of Game 2to $\mathbb{A}_2$ and hence wins with probability $S_2$. On the other hand, if the challenge tuple is a random tuple, $\mathbb{B}$ played $\mathbb{A}_3$'s simulation of Game 2and wins with probability $1 - S_3$. Yet, under the DDH assumption, $\mathbb{B}$'s advantage in the DDH game must be negligible, hence $|S_2 - S_3|$ is negligible.

**Game 4:** In this game, we modify the sequence $B$ so that the honest voters no longer revote. Instead, for all honest voter $x$, we replace all but the last occurrence of the form $(x, \nu)$ in $B$ by an occurrence of the form $(\tilde{x}, \nu)$ which uses a fresh and unique $\tilde{x} < 0$. This way, the last vote remains the same but the previous votes are replaced by a vote with a fresh, random (and fake) credential. Note that this modification happens on the sequence $B$, before the voters actually cast their votes according to this sequence.

Let $n_r$ be the number of revotes. We set $\mathbb{A}_4 = \mathbb{A}_3$ and to argue that $|S_4 - S_3|$ is negligible, we give another succession of hops $H_0, \cdots, H_{n_r}$ such that in game $H_i$, we replace the last $i$ revotes as described above. This way, $H_0$ is Game 3 and $H_{n_r}$ is Game 4. For each of these games, we denote $W_i$ the probability that $\mathbb{A}_4$ wins this game.

Now, let $i$ be some index. We construct an adversary $\mathbb{B}$ for IND-PA0 (see Section C-A) as follows. First, $\mathbb{B}$ receives $\mathsf{pk}$ from the IND-PA0 game. It simulates game $H_i$ by giving this $\mathsf{pk}$ to $\mathbb{A}_4$ and generating the credentials at random. It then gets $(j, \alpha)$ from $H_i$ and chooses $b$ at random. Afterwards, $\mathbb{B}$ generates $B$ from the distribution $\mathcal{B}$, $(j, \alpha)$ and $b$. It then continues the simulation of $H_i$ by replacing the $i$ last revotes as necessary. However, for the next remaining revote, it looks up for the previous vote $(x, \nu)$ with the same $x$ and generates a random, fresh credential $c$. It plays $(c_x, \nu), (c, \nu)$ in the IND-PA0 game and gets an encrypted ballot $C$ which it plays in the simulation of $H_i$ instead of a honestly generated ballot for $(c_x, \nu)$ (therefore, $C$ is added on the board). Finally, to compute the tally, $\mathbb{B}$ simply plays the concatenation of all the valid ballots sent by $\mathbb{A}_4$ in the IND-PA0 game, which returns a decryption of these ballots. Remark that due to the nature of `isVal` which uses a form of weeding, a valid ballot is not already on the board. Therefore, none of the valid ballots sent by $\mathbb{A}_4$ can be equal to $C$, so that the IND-PA0 will indeed accept to decrypt them. $\mathbb{B}$ computes the result of the tally from the plaintexts and gives it to $\mathbb{A}_4$ which answers with some bit $b'$. This is possible because every valid ballot of the board has a valid ZKP $\pi_2$, whose soundness guarantees that the corresponding plaintext $(c, \nu)$ is such that $c$ is a $\lambda$-bit credential and $\nu$ a valid voting option. If $b = b'$, $\mathbb{B}$ states that the IND-PA0 game encrypted $(c_x, \nu)$, and $(c, \nu)$ otherwise.

Clearly, when the IND-PA0 game encrypts $(c_x, \nu)$ (resp. $(c, \nu)$), $\mathbb{B}$ plays a perfect simulation of game $H_i$ (resp. $H_{i+1}$) so that $b = b'$ with probability $W_i$ (resp. $W_{i+1}$). Hence, $\mathbb{B}$'s advantage in the IND-PA0 game is $|\frac{1}{2}(W_i + 1 - W_{i+1}) - \frac{1}{2}| \leq \varepsilon_{\mathsf{PA0}}$. By the triangular inequality,

$$|S_4 - S_3| \leq 2n_r\varepsilon_{\mathsf{PA0}}.$$

**Game 5:** In this game, the adversary no longer has access to the roster $\boldsymbol{R}$ at line 4. We construct $\mathbb{A}_5$ which interacts with $\mathbb{A}_4$ by simulating Game 4. For this purpose, it generates $n_V\lambda$ ElGamal encryptions of random bits and uses it to simulate $\boldsymbol{R}$. For the remaining of the game, it can play a perfect simulation since both games are identical. To argue that $|S_5 - S_4|$ is negligible, we construct a succession of hop $H_0, \cdots, H_{n_V}$ such that in game $H_i$, the last $i$ elements of $\boldsymbol{R}$ are replaced by a random encryption. This way, $H_0$ is Game 4 and $H_{n_V}$ is Game 5. For each of these games, we denote $W_i$ the probability that $\mathbb{A}_4$ wins this game.

Now, let $i$ be some index. We construct an adversary $\mathbb{B}$ for IND-PA0 as follows. $\mathbb{B}$ gets $\mathsf{pk}$ from the IND-PA0 game and plays this $\mathsf{pk}$ to $\mathbb{A}_4$ to simulate $H_i$. It generates the credentials and replaces the last $i$ elements of the roster by encryptions of random bits just as in $H_i$. However, for the $(n_V - i)$th element of $\boldsymbol{R}$, it generates a second random credential $c$ and plays the pair $(c_{n_V-i}, 1), (c, 1)$ in the IND-PA0 game which answers with some encryption $C$. $\mathbb{B}$ retrieves an ElGamal bitwise encryption of the credential from $C$ and uses it as the $(n_V - i)$th element of $\boldsymbol{R}$ instead of an honest bitwise encryption of $c_{n_V-i}$. Afterwards, $\mathbb{B}$ continues the simulation of $H_i$ (see the transition to Game 4 for more details) and outputs 1 if and only if $\mathbb{A}_4$ wins the game.

Clearly, when the IND-PA0 encrypts $(c_{n_V-i}, 1)$ (resp. $(c, 1)$), $\mathbb{B}$ plays a perfect simulation of game $H_i$ (resp. $H_{i+1}$) to $\mathbb{A}_4$, so that the advantage of $\mathbb{B}$ in the IND-PA0 game is $|\frac{1}{2}(W_i + 1 - W_{i+1}) - \frac{1}{2}| \leq \varepsilon_{\mathsf{PA0}}$. By the triangular inequality,

$$|S_5 - S_4| \leq 2n_V\varepsilon_{\mathsf{PA0}}.$$

**Game 6:** In this game, before computing the tally, we decrypt every valid ballot sent by the adversary at lines 23 and 26. If one of these ballots uses the same credential as a ballot sent by an honest voter (*i.e.* a ballot added to the board at line 25 for some $(i, \nu_i)$ with $i \in [1, n_V] \backslash (A \bigcup \{j\})$, we abort the game and output a random bit.

Now, we set $\mathbb{A}_6 = \mathbb{A}_5$ and, to argue that $|S_6 - S_5|$ is negligible, we remark that $|S_6 - S_5| = \varepsilon/2$, where $\varepsilon$ is the probability that we abort in Game 6. Let $E$ be the event of an abortion. We construct an adversary $\mathbb{B}$ for IND-PA0 which wins with a non-negligible advantage whenever $E$ occurs and wins with probability $1/2$ otherwise, which shows that $\varepsilon$ is negligible.

First, $\mathbb{B}$ gets $\mathsf{pk}$ from the IND-PA0 game and forwards it to $\mathbb{A}_5$. It simulate Game 5 as in the transition to Game 4. However, it chooses a random honest voter $x$ that would send a ballot (if no honest voter votes, $E$ cannot happen) and, for this voter, generates a fresh, second random credential $c$. When this voter votes (which happens at most once due to the transition to Game 4), $\mathbb{B}$ plays the pair $(c_x, \nu), (c, \nu)$ in the IND-PA0 game, where $\nu$ is the voting option chosen by $x$. It gets back an encrypted ballot $C$, which it uses in the simulation instead of the ballot from $x$. Afterwards, $\mathbb{B}$ gets the list of all valid ballots sent by $\mathbb{A}_5$ in the simulation and plays them in the IND-PA0 game to get their decryption. If there is a ballot which uses the credential $c_x$ (resp. $c$), $\mathbb{B}$ states that the IND-PA0 encrypted $(c_x, \nu)$ (resp. $(c, \nu)$). If there is no such ballot, a ballot which uses $c$ and a ballot which uses $c_x$ or if no honest voter votes, $\mathbb{B}$ guesses at random.

Now, suppose that $b = 0$ (resp. 1) in the IND-PA0 game; in other words, that $C$ is an encryption of $(c_x, \nu)$ (resp. $(c, \nu)$). Let $q$ be the number of valid ballots sent by $\mathbb{A}_5$. With probability $\varepsilon$, $\mathbb{A}_5$ managed to produce a ballot which uses the same credential as a ballot sent by some honest voter. In this case, with probability at least $1/n_H$, one of the concerned honest voter is $x$. Then, when $\mathbb{B}$ gets the decryption from the IND-PA0 game, there is a ballot of the form $(c_x, \gamma)$ (resp. $(c, \gamma)$). In addition, $\mathbb{A}_5$ has no information about $c$ (resp. $c_x$) so that with probability at least $1 - q/2^\lambda$, there is no ballot of the form $(c, \gamma)$ (resp. $(c_x, \gamma)$). Hence $\mathbb{B}$ wins with probability at least $1 - q/2^{\lambda+1}$. Otherwise, no ballot uses the credential $c_x$ (resp. $c$) and since the

adversary has no information about $c$ (resp. $c_x$), the probability that a ballot uses the credential $c$ (resp. $c_x$) is at most $q/2^\lambda$. Therefore, the probability that $\mathbb{B}$ wins the IND-PA0 game is at least $(1-q/2^\lambda)/2$. Overall, $\mathbb{B}$'s probability to win is at least

$$\frac{\varepsilon}{n_H}(1-q/2^{\lambda+1})+(1-\frac{\varepsilon}{n_H})(1-q/2^\lambda)/2=\frac{1}{2}+\frac{\varepsilon}{2n_H}-\frac{q}{2^{\lambda+1}}.$$

Therefore, we have

$$\frac{\varepsilon}{2n_H}-\frac{q}{2^{\lambda+1}}\leq \mathsf{Adv}_{\mathbb{B}}^{\mathsf{IND-PA0}}\leq \varepsilon_{\mathsf{PA0}},$$

hence $|S_6-S_5|=\varepsilon/2\leq n_H\varepsilon_{\mathsf{PA0}}+\frac{qn_H}{2^{\lambda+1}}$, where $q$ is the number of valid ballots sent by the adversary.

**Game 7:** In this game, we remove lines 23 and 24 so that the adversary can no longer insert its own ballots between two honest ballots. In other words, the adversary must send all its ballots at the end, after every honest voter has voted. We construct $\mathbb{A}_7$ which interacts with $\mathbb{A}_6$ by simulating Game 6. For this purpose, $\mathbb{A}_7$ gets $BB=(B_1,\cdots,B_n)$ at line 26 and creates a fake empty ballot box $BB'$. Then, in the $k$th iteration of the for loop, it simply appends to $BB'$ the valid ballots output by $\mathbb{A}_6$ and then $B_k$. The remaining of the simulation is similar to that of the transition to Game 4.

Clearly, $\mathbb{A}_7$ plays a perfect simulation of Game 6 if the result of the tally is the same. Besides, the latter can only differ if the credential of a ballot sent by $\mathbb{A}_6$ is the same as the credential of a ballot sent by some honest voter. In this case, both games abort with a random output and $\mathbb{A}_7$'s probability to win is the same as $\mathbb{A}_6$'s in Game 6. Consequently, $S_7=S_6$.

**Game 8:** In this game, the adversary has no longer access to the ballot box $BB$ at line 26 but instead has $|B|$ (which is equal to $|BB|$). With a similar argument as in Game 5, we construct $\mathbb{A}_8$ and we have

$$|S_8-S_7|\leq |BB|\varepsilon_{\mathsf{PA0}}.$$

**Game 9:** In this game, the lines 22 to 27 are replaced by lines 22 to 25 of the ideal game. We construct an adversary $\mathbb{A}_9$ which interacts with $\mathbb{A}_8$ by simulating Game 8. For this purpose, $\mathbb{A}_9$ creates a hashmap with keys $\{c_i; i\in A\}$ and $\tilde{c}$ and values $(\nu_i)_{i\in A}$ and $\beta$ which are initially $\phi$. For each ballot $C\in\boldsymbol{M}$, $\mathbb{A}_9$ extracts $(c,\nu)$ from the ZKPoK and, if $c$ is a key of the hashmap, it changes the corresponding value to $\nu$. It plays the obtained values in Game 9 and receives the result of the tally which it forwards to $\mathbb{A}_8$. Finally, it outputs $\mathbb{A}_8$'s output. Clearly, $S_9=S_8$.

**Game 10:** The final game is the ideal game, where the credentials and the public encryption key have been removed. Note that they did not play any role in Game 9 so that the adversary might as well generate them itself. Clearly, $S_{10}=S_9$.

**Conclusion.** With all the above transitions, we showed that for all adversary $\mathbb{A}$ for the real game, there exists an adversary $\mathbb{B}$ for the ideal game which wins with the same probability (up to a negligible difference. By definition, this shows that CHide is coercion-resistant. □

### C. Privacy

*1) Our definition of privacy:* In [26], a quantitative definition of privacy is proposed. In this definition, the adversary is an observer who can corrupt some voters and talliers. The honest voters vote according to a fixed distribution s in Section III-A and a voter under observation either chooses option $\nu_0$ or $\nu_1$, where $(\nu_0,\nu_1)$ can be chosen by the adversary. The latter must decide which was chosen, and do so with an advantage $\mathsf{Adv}(\mathbb{A})$. A Voting system is then said $\delta$-private if, for all adversary, $2\mathsf{Adv}(\mathbb{A})\leq\delta+\mu$, where $\mu$ is some negligible function.

This definition can be turned into a qualitative one when $\delta$ is shown to be minimal, in a sense that an ideal protocol achieves $\delta'$-privacy with a negligible $|\delta-\delta'|$. Hence, a natural definition of privacy is to compare the probability of success of the adversary in a real process to that in an ideal one, and to show that the difference is negligible. Just as in [26], we consider a definition where the adversary tries to guess the vote of a single voter. Note that, as opposed to the coercion-resistance case, we do not assume that ballots are sent anonymously. Consequently, the adversary can always determine whether a voter abstained or voted. Finally, we consider that the registrar is honest and that up to $t$ talliers can be corrupted by the adversary. To model this, we consider a fixed set $[1,n_C]$ of valid voting options (excluding abstention) and we give Algorithms 14 and 15.

The real process can be read as follows. First, the public encryption key and the shares are generated with $P_{\texttt{Setup}}$, in which the adversary can impersonate up to $t$ corrupted authorities. Then the voters are registered and the roster $\boldsymbol{R}$ is published. The adversary can corrupt voters and chooses a voter under observation $j$. Then the honest voters vote according to a distribution $\mathcal{B}$ and, as in Definition 2, we let the adversary freely insert any number of ballots between two ballots cast by an honest voter. To model the fact that the ballots are not sent anonymously, we give away the identity of the voter to the adversary at line 13. Also, to take into account the fact that the latter could revote a certain number of times we include $j$ as a possible identity for the honest voters at line 9. Then the adversary must choose the voting options $(\nu_0,\nu_1)$ and the voter under observation (re)votes with one of those two options. The adversary is given a last opportunity to cast ballots; after this, the tally is computed with $P_{\texttt{tally}}$, during which the adversary can impersonate the corrupted authorities (we only consider static corruption). Finally, the adversary must guess whether $\nu_0$ or $\nu_1$ was chosen.

**Algorithm 14:** $\mathsf{Real}^{\mathsf{Priv}}$

**Require:** $\mathbb{A}, \lambda, n_T, t, n_V, n_A, n_C, \mathcal{B}$

1 $BB \longleftarrow \varnothing$
2 $\mathsf{pk}, (s_i, h_i)_{i=1}^{n_T} \longleftarrow P^{\mathbb{A}}_{\texttt{Setup}}(\lambda, n_T, t)$
3 $\{c_i; i \in [1, n_V]\}, \boldsymbol{R} \longleftarrow \texttt{register}(\lambda, \mathsf{pk}, n_V)$
4 $A \longleftarrow \mathbb{A}(\boldsymbol{R})$
5 $j, \nu_0, \nu_1 \longleftarrow \mathbb{A}(\{c_i; i \in A\})$
6 (* $\mathbb{A}$ chooses the voter under observation *)
7 **if** $|A| \neq n_A \vee j \notin [1, n_V] \backslash A$ **then**
8  Return 0
9 $B \longleftarrow \mathcal{B}([1, n_V] \backslash A, n_C)$
10 **for** $(i, \nu_i) \in B$ **do**
11  $\boldsymbol{M} \longleftarrow \mathbb{A}(BB)$
12  $BB \leftarrow BB \bigcup \{m \in \boldsymbol{M} \mid \texttt{isVal}(\mathsf{pk}, m, BB) = 1\}$
13  $BB \longleftarrow BB \bigcup \{\texttt{Vote}_{\mathsf{pk}}(c_i, \nu_i)\}; \mathbb{A}(i)$
14 $\boldsymbol{M} \longleftarrow \mathbb{A}(BB, \texttt{"last honest ballot sent"})$
15 $BB \longleftarrow BB \bigcup \{m \in \boldsymbol{M} \mid \texttt{isVal}(\mathsf{pk}, m, BB) = 1\}$
16 $b \xleftarrow{\$} \{0, 1\}$
17 $BB \longleftarrow BB \bigcup \{\texttt{Vote}_{\mathsf{pk}}(c_j, \nu_b)\}$
18 $\boldsymbol{M} \longleftarrow \mathbb{A}(BB)$
19 $BB \longleftarrow BB \bigcup \{m \in \boldsymbol{M} \mid \texttt{isVal}(\mathsf{pk}, m, BB) = 1\}$
20 $\boldsymbol{X}, \Pi \longleftarrow P^{\mathbb{A}}_{\texttt{tally}}(BB, \boldsymbol{R}, \mathsf{pk}, \{h_i, s_i\}, t)$
21 $b' \longleftarrow \mathbb{A}()$
22 Return 1 **if** $\nu_0, \nu_1 \in [1, n_C] \wedge b' == b$ **else** 0

**Algorithm 15:** $\mathsf{Ideal}^{\mathsf{Priv}}$

**Require:** $\mathbb{A}, \lambda, n_T, t, n_V, n_A, n_C, \mathcal{B}$

1
2
3
4 $A \longleftarrow \mathbb{A}(\lambda)$
5 $j, \nu_0, \nu_1 \longleftarrow \mathbb{A}()$
6
7 **if** $|A| \neq n_A \vee j \notin [1, n_V] \backslash A$ **then**
8  Return 0
9 $B \longleftarrow \mathcal{B}([1, n_V] \backslash (A \bigcup \{j\}), n_C)$
10 $I \longleftarrow (i)_{(i,\nu) \in B}$
11
12
13
14 $(\alpha_i)_{i \in A} \longleftarrow \mathbb{A}(I)$
15 $B \longleftarrow B \bigcup \{(i, \alpha_i) \mid i \in A, \alpha_i \in [1, n_C]\}$
16 $b \xleftarrow{\$} \{0, 1\}$
17 $B \longleftarrow B \bigcup \{(j, \nu_b)\}$
18
19
20 $r \longleftarrow \texttt{tally}(\texttt{cleanse}(B))$
21 $b' \longleftarrow \mathbb{A}(r)$
22 Return 1 **if** $\nu_0, \nu_1 \in [1, n_C] \wedge b' == b$ **else** 0

Fig. 9: Definition of privacy, $\lambda$ is the security parameter, $n_T$ the number of talliers, $t$ the threshold, $n_V$ the number of voters, $n_A$ the number of corrupted voters, $n_C$ the number of voting options (excluding abstention) and $\mathcal{B}$ the distribution.

The ideal process is simpler. The adversary still corrupts $n_A$ voters and chooses $(\nu_0, \nu_1)$, but must guess which option was chosen given only the result $r$ of the election, and the list $I$ that indicates which voter (re)votes at which moment. The latter is given in the ideal game to model that the ballots are not sent anonymously.

**Definition 4** (Vote privacy)**.** *We say that a voting protocol ($P_{\texttt{Setup}}$,`register`,`Vote`,`Check`,`isVal`,`Fakecred`,$P_{\texttt{tally}}$,`Verify`) guarantees* vote privacy *w.r.t. a result function* `tally` *if, for all parameters $n_T, t, n_V, n_A, n_C$ with $t < a$ and $n_A \leq n_V$, for all distribution $\mathcal{B}$ and for all PPT adversary $\mathbb{A}$, there exists a PPT adversary $\mathbb{B}$ and a negligible function $\mu$ such that*

$$|Pr(\mathsf{Real}^{\mathsf{Priv}}(\mathbb{A}, \lambda, n_T, t, n_V, n_A, n_C, \mathcal{B}) = 1) - \Pr(\mathsf{Ideal}^{\mathsf{Priv}}(\mathbb{B}, \lambda, n_T, t, n_V, n_A, n_C, \mathcal{B}) = 1)| \leq \mu(\lambda).$$

*2) Proof that CHide is private:* We now prove that CHide is private, as of Definition 4. The proof is extremely similar to that of coercion-resistance in Section C-B and uses roughly the same transitions. We reproduce the transitions, with a few little tweaks to take into account the differences between $\mathsf{Real}^{\mathsf{CR}}$ and $\mathsf{Real}^{\mathsf{Priv}}$. Just as for coercion-resistance, we make several assumptions; the main difference is that we no longer assume anonymous channels for voting.

**Theorem 4.** *Under the DDH assumption and in the ROM, CHide has privacy as of Definition 4.*

*Proof.* We give a succession of games such that Game 0 is the real game and Game 9 is the ideal game. We consider a PPT $\mathbb{A}_0$ for Game 0. For Game $i$, we construct a PPT adversary $\mathbb{A}_i$ for this game and we denote $S_i$ the probability that $\mathbb{A}_i$ wins this game. (To ease the notations, we drop the dependency in $\lambda$ when the context is clear.) For all $i$, we show that $|S_{i+1} - S_i|$ is negligible, which proves that $|S_0 - S_9|$ is also negligible.

**Game 1:** In this game, the adversary no longer takes part into the tally process at line 28. Instead, it is given the output of $T_{\texttt{tally}}$.

**Game 2:** In this game, we remove line 2. Instead, we generate a perfectly random public encryption key $\mathsf{pk}$ and give it to the adversary at line 4.

**Game 3:** In this game, the adversary is no longer given the output of the conditional gates, that we denote $\Pi^Z$.

**Game 4:** In this game, we modify the sequence $B$ so that the honest voters no longer revote. Instead, for all honest voter $x$, we replace all but the last occurrence of the form $(x, \nu)$ in $B$ by an occurrence of the form $(x \to \tilde{x}, \nu)$ where $\tilde{x}$ is a fresh

and unique negative number. Then, at line 13, when $i$ is of the form $x \to \tilde{x}$, we give $x$ to the adversary but we add a ballot of the form $\mathtt{Vote}_{\mathsf{pk}}(c, \nu)$ with a fresh random (fake) credential $c$.

Although this transition is slightly different from the transition to Game 4 in the proof of coercion resistance, the reduction argument is the same.

**Game 5:** In this game, the adversary no longer has access to the roster $\boldsymbol{R}$ at line 4.

**Game 6:** In this game, before computing the tally, we decrypt every valid ballot sent by the adversary at lines 11, 14 and 18. If one of these ballots uses the same credential as a ballot sent by an honest voter (*i.e.* a ballot added to the board at line 13 for some $(i, \nu_i)$ with $i \in [1, n_V] \backslash A$, we abort the game and output a random bit.

**Game 7:** In this game, we remove lines 11, 12, 14 and 15 so that the adversary can no longer insert its own ballots between two honest ballots (including those sent by the voter under observation). In other words, the adversary must send all its ballots at the end, after every honest voter has voted.

**Game 8:** In this game, the adversary has no longer access to the ballot box $BB$ at lines 18 and 5 but instead is given $I$.

Compared to the transition to Game 8 in the proof of coercion resistance, the adversary know has access to $I$ instead of $|B|$. This extra information is used to simulate line 13.

**Game 9:** The final game is the ideal game.

We construct an adversary $\mathbb{A}_9$ which interacts with $\mathbb{A}_8$ by simulating Game 8. For this purpose, $\mathbb{A}_9$ runs the setup and the registration honestly, by generating the secret key and the credentials. This allows $\mathbb{A}_9$ to get $j, \nu_0, \nu_1$ from $\mathbb{A}_8$, that it plays in the ideal game. Then, when given $I$ in the ideal game, it forwards it to $\mathbb{A}_8$ which answers with $\boldsymbol{M}$. To deduce the corresponding voting options $(\nu_i)_{i \in A}$, $\mathbb{A}_9$ creates a hashmap with the keys $\{c_i; i \in A\}$, and values $(\nu_i)_{i \in A}$ (initially $\phi$, for abstention). For each valid ballot cast by $\mathbb{A}_8$, $\mathbb{A}_9$ decrypts the ballot using the secret key and deduces $(\nu, c)$. Since the ballot is valid, by the soundness of the ZKP, $c$ consists of $\lambda$ bits and $\nu$ is a valid voting option. If $c$ is a key of the hashmap, $\mathbb{A}_9$ changes the corresponding value to $\nu$. (Otherwise, it ignores the ballot.) It plays the obtained values in Game 9 and receives the result of the tally which it forwards to $\mathbb{A}_8$. Finally, it outputs $\mathbb{A}_8$'s output. Remark that $\mathbb{A}_9$ played a perfect simulation of $Game\ 8$, so that $S_9 = S_8$.

□

### *D. Verifiability*

*1) Our definition of verifiability:* Verifiability is often split into several sub-properties, namely *cast-as-intended*, where the voters can verify that their voting device produced a ballot which indeed contains the desired voting option; *recorded-as-cast*, where the voters can verify that the ballot produced by their voting device is indeed added to the ballot box (typically, a public bulletin board); *tally-as-recorded*, where anyone can verify that the result obtained with the tally protocol indeed corresponds to that obtained by opening every (valid) ballot of the ballot box and then applying the counting function on the cleartexts; and *eligibility*, where anyone can verify that the ballot box indeed contains ballot that comes from an eligible voter, and that each voter can have at most one ballot counted. The first two properties form the *individual verifiability* while the two last form the *universal verifiability*.

In this paper, we focus on the universal verifiability. We do not consider cast-as-intended verification and assume that the voting device is not compromised. Achieving both cast-as-intended verification and coercion-resistance is a non-trivial task that is out of the scope of this article. In addition, we assume that each time a honest voter casts a ballot, they verify that the ballot appears on the board. Since the bulletin board is considered honest, this immediately gives individual verifiability.

To formalize the notion of universal privacy, we use the approach of [12] which gives a game-based definition. Originally, their definition mixes the universal and individual verifiability and divides the honest voters into two subsets, depending on whether they perform the verification or not. For those who do not check, they require that their ballots can be dropped but that their content cannot be modified. This, however, comes with a difficulty when revoting is allowed. Indeed, if a voter's votes are, in order, $(\nu_1, \nu_2, \nu_3)$ and if the voter does not verify that the corresponding ballots appear on the board, then the adversary might drop any of those ballots and have the voter abstain, or have their vote counted as $\nu_1$, $\nu_2$ or $\nu_3$ as it sees fit. A stronger and more desirable definition would require the adversary to either have the voter abstain or have their last choice $\nu_3$ be counted. This, however, requires specific measures which are yet to appear in the literature. In this paper, we consider that those considerations are out of scope and thus consider a weaker definition where every voter systematically verifies that their ballot appears on the board.

This gives Definition 5, where the adversary first takes part in the setup and corrupts a subset $A$ of the voters. Then, given the roster and the credentials of the corrupted voters, the adversary must generate a valid bulletin board. For this purpose, it has access to the oracle $\mathcal{O}_{\mathsf{cast}}$ which takes as input a ballot $\mathbf{b}$ (see Algorithm 17). If the ballot is valid w.r.t. $BB$, $\mathcal{O}_{\mathsf{cast}}$ adds $\mathbf{b}$ to $BB$. The adversary also has access to $\mathcal{O}_{\mathsf{vote}}$, which models the ballots sent by the honest voters (see Algorithm 16). However, $\mathcal{O}_{\mathsf{vote}}$ modifies two inner tables $\mathsf{H}$ and $\mathsf{HV}$. The first one model the fact that a honest voter is supposed to check that their ballot appears on the board. If a honest voter vote but does not check or if their check fails, they become unhappy. The second tabular represent the state of a voter, and contains their ballot, their credential and their chosen voting option. The adversary

**Algorithm 19:** Ver

**Require:** $\mathbb{A}, \lambda, n_T, t, n_V, n_A, n_C$

$\mathsf{pk}, (s_i, h_i)_{i=1}^{n_T} \longleftarrow P^{\mathbb{A}}_{\texttt{Setup}}(\lambda, n_T, t)$

$\{c_i; i \in [1, n_V]\}, \boldsymbol{R} \longleftarrow \texttt{register}(\lambda, \mathsf{pk}, n_V)$

$A \longleftarrow \mathbb{A}(\boldsymbol{R})$

$BB \longleftarrow \varnothing$; **for** $i \in [1, n_V] \backslash A$ **do** $\mathsf{HV}_i \longleftarrow \perp$;$\mathsf{H}_i \longleftarrow 1$

$\mathbb{A}^{\mathcal{O}_{\mathsf{vote}}, \mathcal{O}_{\mathsf{cast}}, \mathcal{O}_{\mathsf{check}}}(\{c_i; i \in A\})$

**if** $|A| \neq n_A \vee \exists i \in [1, n_V] \backslash A \mid \mathsf{H}_i < 1$ **then** Return 0

$\boldsymbol{X}, \Pi \longleftarrow P^{\mathbb{A}}_{\texttt{tally}}(BB, \boldsymbol{R}, \mathsf{pk}, \{h_i, s_i\}, t)$

**if** $\texttt{Verify}(BB, \Pi, \boldsymbol{X}) = 0$ **then** Return 0

**if** $\exists (\mathsf{cor}_i, \alpha_i)_{i=1}^{n} \in (A \times [1, n_C]) \mid \boldsymbol{X} = \texttt{tally}(\texttt{cleanse}(\{(i, \mathsf{HV}_i[4]) \mid i \in [1, n_V] \backslash A, \mathsf{HV}_i \neq \perp\} \bigcup \{(\mathsf{cor}_i, \alpha_i) \mid i \in [1, n]\}))$ **then** Return 0 **else** Return 1

Fig. 10: Definition of verifiability, $\lambda$ is the security parameter, $n_T$ the number of talliers, $t$ the threshold, $n_V$ the number of voters, $n_A$ the number of corrupted voters and $n_C$ the number of voting options (excluding abstention)

can have a voter $i$ check their vote by calling $\mathcal{O}_{\mathsf{check}}(i)$ (see Algorithm 18) which calls the `Check` procedure with the current state $\mathsf{HV}_i$ of the voter and the current bulletin board. The adversary can call $\mathcal{O}_{\mathsf{cast}}$, $\mathcal{O}_{\mathsf{vote}}$ and $\mathcal{O}_{\mathsf{check}}$ any (polynomial) number of times, in any order. It then takes part into the tally to produce the result $\boldsymbol{X}$ as well as the transcript $\Pi$. The goal of the adversary is to produce a valid $\boldsymbol{X}, \Pi$ which is different from any result obtained with the sequence $(i, \mathsf{HV}_i)_i$ concatenated with another sequence $(\mathsf{cor}_i, \alpha_i)_i$, where the $\mathsf{cor}_i$'s are corrupted voters. In addition, all honest voter who have voted must be happy.

We acknowledge that, in our definition, we assume that the registrar is honest, and that up to $t$ decryption trustees can be corrupted. Usually, one would want verifiability even if all the talliers and the registrar are corrupted. In [12], for instance, it is assumed that either the registrar or the bulletin board is honest. However, in JCJ-like voting systems, the registrar knows the credential of the voter and can therefore break eligibility by voting instead of the abstaining voters (ballot stuffing), or even change the choice of any voter by revoting with their credential afterwards. Similarly, since an encryption of the credentials is published in the roster $\boldsymbol{R}$, if more than a threshold $t$ of talliers is corrupted, they can decrypt the credentials and perform the same attacks. Therefore, it is necessary to assume that the registrar is honest and that up to a threshold of talliers can be corrupted. Those shortcomings were already present in JCJ.

**Definition 5** (Verifiability)**.** *We say that the a voting protocol* ($P_{\texttt{Setup}}$,`register`,`Vote`,`Check`,`isVal`,`Fakecred`,$P_{\texttt{tally}}$,`Verify`) *is verifiable if, for all adversary* $\mathbb{A}$*, for all parameters* $(n_T, t, n_V, n_C)$*, there exists a negligible function* $\mu$ *such that* $\Pr(\mathsf{Ver}(\mathbb{A}, \lambda, n_T, t, n_V, n_A, n_C) = 1) \leq \mu(\lambda)$.

**Algorithm 16:** $\mathcal{O}_{\mathsf{vote}}$ (updates the tabulars HV and H)

**Require:** $i, \nu$ ($n_V$ is the number of voters, $A$ the corrupted voters, $n_C$ the number of voting options, pk the public key and $c_i$ the credential of voter $i$)

**if** $i \in [1, n_V] \backslash A \wedge \nu \in [1, n_C]$ **then**

  $\mathsf{H}_i \longleftarrow \mathsf{H}_i - 1$;

  $\mathbf{b} \longleftarrow \texttt{Vote}_{\mathsf{pk}}(c_i, \nu)$;

  $\mathsf{HV}_i \longleftarrow (\mathsf{pk}, \mathbf{b}, c_i, \nu)$;

  Return $\mathbf{b}$

**Algorithm 17:** $\mathcal{O}_{\mathsf{cast}}$ (updates the bulletin board $BB$)

**Require:** $\mathbf{b}$

**if** $\texttt{isVal}(\mathsf{pk}, BB, \mathbf{b}) = 1$ **then** $BB \longleftarrow BB \bigcup \{\mathbf{b}\}$

**Algorithm 18:** $\mathcal{O}_{\mathsf{check}}$ (updates HV and H)

**Require:** $i$

$\mathsf{pk}, \mathbf{b}, c_i, \nu \longleftarrow \mathsf{HV}_i$; $\mathsf{HV}_i[2] \longleftarrow \perp$

**if** $\texttt{Check}(BB, \mathsf{pk}, \mathbf{b}, c_i, \nu) = 1$ **then** $\mathsf{H}_i \longleftarrow \mathsf{H}_i + 1$

*2) Proof that CHide is verifiable:* We now prove that CHide is verifiable, as of Definition 5. Just as for coercion-resistance, we make several assumptions; the main difference is that the adversary has a complete control over the vote of the honest voters.

**Theorem 5.** *Under the DDH assumption and in the ROM, CHide is verifiable as of Definition 5.*

*Proof.* Despite the difference between coercion-resistance and verifiability, there are arguments in common in the two proofs and the first transitions are the same.

**Game 1:** In this game, the adversary no longer takes part into the tally process at line 28. Instead, it is given the output of $T_{\texttt{tally}}$.

**Game 2:** In this game, we remove line 2. Instead, we generate a perfectly random public encryption key $\mathsf{pk}$ and give it to the adversary at line 4.

**Game 3:** In this game, the adversary is no longer given the output of the conditional gates, that we denote $\Pi^Z$.

**Game 4:** In this game, we create a secret internal state $\texttt{Cred}_i$ for each voter. It contains a sequence of credentials which is initially $[c_i]$, where $c_i$ is $i$'s credential. In addition, we also modify $\mathcal{O}_{\mathsf{vote}}$ so that it uses the last element of $\texttt{Cred}_i$ instead of $c_i$. It also generates a fresh random credential $c$ which is added at the end of $\texttt{Cred}_i$, so that the next instance of $\mathcal{O}_{\mathsf{vote}}$ with the same voter will use this new credential instead of the old one. (In case if $c$ is a collision with another voter, we pick another random credential.) Finally, we change the way the tally $\boldsymbol{X}$ is computed. Now, we decrypt each ballot of $BB$ into a couple $(c, \nu)$. For each ballot, we look for a voter $i$ such that $c$ appears in the sequence $\texttt{Cred}_i$ and we set $i$'s vote as $\nu$ (since there is no collision, there may be up to one such voter). Finally, we define $\boldsymbol{X}$ as the voters' votes.

We take $\mathbb{A}_4 = \mathbb{A}_3$ and we show that $|S_4 - S_3|$ is negligible. For this purpose, for all honest voter $j$, we consider another succession of games $H_0^j, \cdots, H_{n_j}^j$, where $n_j$ is the number of times that $\mathbb{A}_3$ calls $\mathcal{O}_{\mathsf{vote}}$ with the voter $j$ and a valid $\nu$. (For the sake of notations, we set $n_j = 0$ when $j$ is a corrupted voter.) Game $H_k^j$ is like Game 4, except that the first $n_j - k$ times $\mathcal{O}_{\mathsf{vote}}$ is called with voter $j$, we do not add the new credential $c$ in $\texttt{Cred}_j$, so that the next instance of $\mathcal{O}_{\mathsf{vote}}$ with voter $j$ will still use the credential $c_j$. As for all voter $i > j$, we never update their $\texttt{Cred}_i$, so that they are treated as in Game 3. This way, $H_0^0$ is Game 3, $H_{n_j}^j$ is $H_0^{j+1}$ for all $j$ and $H_{n_{n_V}}^{n_V}$ is Game 4. For each of these games, we denote $W_k^j$ the probability that $\mathbb{A}_3$ wins the game.

Now, let $k$ be some index and $j$ be some voter. We construct $\mathbb{B}$ for IND-PA0 as follows. $\mathbb{B}$ gets $\mathsf{pk}$ from the IND-PA0 game and uses this value to play a simulation of $H_k^j$ to $\mathbb{A}_3$. For this purpose, it creates and updates $BB, \mathsf{HV}, \mathsf{H}$ and $\texttt{Cred}$, and simulates the oracles. It also simulates the $\texttt{register}$ procedure by generating a random credential for each voter and computing $\boldsymbol{R}$ by encrypting the bits of those credentials, using $\mathsf{pk}$. However, the $(n_j - k - 1)$th time $\mathcal{O}_{\mathsf{vote}}$ is called with voter $j$, $\mathbb{B}$ generates a new $c$ in $\texttt{Cred}_j$ as in game $H_{k+1}^j$. In addition, the next time $\mathcal{O}_{\mathsf{vote}}$ is called with voter $j$ (let $\nu$ be the corresponding voting option), instead of using the last credential of $\texttt{Cred}_j$ (which is $c$), $\mathbb{B}$ plays $(c_i, \nu), (c, \nu)$ in the IND-PA0 game and uses the result as the output $\mathbf{b}$. At some point, $\mathbb{A}_3$ terminates and $\mathbb{B}$ must decide whether the IND-PA0 game encrypted $(c_i, \nu)$ or $(c, \nu)$. For this purpose, $\mathbb{B}$ uses $BB$, the list of the ballots cast by $\mathbb{B}$. If $\mathbf{b} \in BB$, $\mathbb{B}$ removes it from $BB$ but keeps in memory its index. For the remaining ballots, $\mathbb{B}$ decrypts them using the decryption oracle of the IND-PA0 game. With all these informations, $\mathbb{B}$ can compute the tally as in Game 4. If the condition of line 9 is not verified (*i.e.* $\mathbb{A}_3$ won the simulation), $\mathbb{B}$ states that $(c_i, \nu)$ was encrypted; otherwise, it states that $(c, \nu$ was encrypted.

Clearly, when $(c_i, \nu)$ is encrypted, $\mathbb{B}$ plays a perfect simulation of $H_k^j$ while, when $(c, \nu)$ is encrypted, $\mathbb{B}$ plays a perfect simulation of $H_{k+1}^j$. Therefore, $\mathbb{B}$'s advantage in the IND-PA0 game is $|1/2(W_k^j + 1 - W_{k+1}^j) - 1/2| \leq \varepsilon_{\mathsf{PA0}}$. By triangular inequality, $|S_4 - S_3| \leq \sum_{j=1}^{n_V} 2n_j\varepsilon_{\mathsf{PA0}} = 2q\varepsilon_{\mathsf{PA0}}$, where $q$ is the number of calls to $\mathcal{O}_{\mathsf{vote}}$.

**Game 5:** In this game, the adversary no longer has access to the roster $\boldsymbol{R}$ at line 3. We construct $\mathbb{A}_5$ which interacts with $\mathbb{A}_4$ by simulating Game 4. For this purpose, it generates $n_V\lambda$ ElGamal encryptions of random bits and uses it to simulate $\boldsymbol{R}$. For the remaining of the game, it can play a perfect simulation since both games are identical. To argue that $|S_5 - S_4|$ is negligible, we construct a succession of hop $H_0, \cdots, H_{n_V}$ such that in game $H_i$, the last $i$ elements of $\boldsymbol{R}$ are replaced by a random encryption. This way, $H_0$ is Game 4 and $H_{n_V}$ is Game 5. For each of these games, we denote $W_i$ the probability that $\mathbb{A}_4$ wins this game.

Now, let $i$ be some index. We construct an adversary $\mathbb{B}$ for IND-PA0 as follows. $\mathbb{B}$ gets $\mathsf{pk}$ from the IND-PA0 game and plays this $\mathsf{pk}$ to $\mathbb{A}_4$ to simulate $H_i$. It generates the credentials and replaces the last $i$ elements of the roster by encryptions of random bits just as in $H_i$. However, for the $(n_V - i)$th element of $\boldsymbol{R}$, it generates a second random credential $c$ and plays the pair $(c_{n_V - i}, 1), (c, 1)$ in the IND-PA0 game which answers with some encryption $C$. $\mathbb{B}$ retrieves an ElGamal bitwise encryption of the credential from $C$ and uses it as the $(n_V - i)$th element of $\boldsymbol{R}$ instead of an honest bitwise encryption of $c_{n_V - i}$. Afterwards, $\mathbb{B}$ continues the simulation of $H_i$ (see the transition to Game 4 for more details) and outputs 1 if and only if $\mathbb{A}_4$ wins the game.

Clearly, when the IND-PA0 encrypts $(c_{n_V - i}, 1)$ (resp. $(c, 1)$), $\mathbb{B}$ plays a perfect simulation of game $H_i$ (resp. $H_{i+1}$) to $\mathbb{A}_4$, so that the advantage of $\mathbb{B}$ in the IND-PA0 game is $|\frac{1}{2}(W_i + 1 - W_{i+1}) - \frac{1}{2}| \leq \varepsilon_{\mathsf{PA0}}$. By the triangular inequality,

$$|S_5 - S_4| \leq 2n_V\varepsilon_{\mathsf{PA0}}.$$

**Game 6:** In this game, before computing the result, we decrypt every valid ballot of $BB$ which is not the output of some $\mathcal{O}_{\mathsf{vote}}$. If one of these ballots uses the same credential as a ballot output by $\mathcal{O}_{\mathsf{vote}}$, we abort the game and output a random bit.

Now, we set $\mathbb{A}_6 = \mathbb{A}_5$ and, to argue that $|S_6 - S_5|$ is negligible, we remark that $|S_6 - S_5| = \varepsilon/2$, where $\varepsilon$ is the probability that we abort in Game 6. Let $E$ be the event of an abortion. We construct an adversary $\mathbb{B}$ for IND-PA0 which wins with a non-negligible advantage whenever $E$ occurs and wins with probability $1/2$ otherwise, which shows that $\varepsilon$ is negligible.

First, $\mathbb{B}$ gets $\mathsf{pk}$ from the IND-PA0 game and forwards it to $\mathbb{A}_5$. It simulate Game 5 as in the transition to Game 4. However, it chooses a random instance of $\mathcal{O}_{\mathsf{vote}}$ (let $(x, \nu)$ be its inputs) and, for this instance, generates a fresh, second random credential

$\tilde{c}$. $\mathbb{B}$ plays the pair $(c,\nu),(\tilde{c},\nu)$ in the IND-PA0 game, where $c$ is the last credential in $\texttt{Cred}_x$. It gets back an encrypted ballot $\mathbf{b}$, which it uses in the simulation as the output of $\mathcal{O}_{\mathsf{vote}}$. Afterwards, $\mathbb{B}$ removes from $BB$ any ballot output by $\mathcal{O}_{\mathsf{vote}}$ (including $\mathbf{b}$) and plays $BB$ in the IND-PA0 game to get the decryption of the remaining ballots. If one uses the credential $c$ (resp. $\tilde{c}$), $\mathbb{B}$ states that the IND-PA0 encrypted $(c,\nu)$ (resp. $(\tilde{c},\nu)$). If there is no such ballot or if there is a ballot which uses $c$ and a ballot which uses $\tilde{c}$, $\mathbb{B}$ guesses at random.

Now, suppose that $b=0$ (resp. 1) in the IND-PA0 game; in other words, that $\mathbf{b}$ is an encryption of $(c,\nu)$ (resp. $(\tilde{c},\nu)$). Let $q_v$ be the number of (valid) calls to $\mathcal{O}_{\mathsf{vote}}$ and $q_c$ be the number of ballots from $BB$ which are not an output of $\mathcal{O}_{\mathsf{vote}}$. With probability $\varepsilon$, $\mathbb{A}_5$ managed to produce a ballot which uses the same credential as a ballot output by $\mathcal{O}_{\mathsf{vote}}$. In this case, with probability at least $1/q_v$, one of the concerned ballot is $\mathbf{b}$. Then, when $\mathbb{B}$ gets the decryption from the IND-PA0 game, there is a ballot of the form $(c,\gamma)$ (resp. $(\tilde{c},\gamma)$). In addition, $\mathbb{A}_5$ has no information about $\tilde{c}$ (resp. $c$) so that with probability at least $1-q_c/2^\lambda$, there is no ballot of the form $(\tilde{c},\gamma)$ (resp. $(c,\gamma)$). Hence $\mathbb{B}$ wins with probability at least $1-q_c/2^{\lambda+1}$. Otherwise, no ballot uses the credential $c$ (resp. $\tilde{c}$) and since the adversary has no information about $\tilde{c}$ (resp. $c$), the probability that a ballot uses the credential $\tilde{c}$ (resp. $c$) is at most $q_c/2^\lambda$. Therefore, the probability that $\mathbb{B}$ wins the IND-PA0 game is at least $(1-q_c/2^\lambda)/2$. Overall, $\mathbb{B}$'s probability to win is at least

$$\frac{\varepsilon}{q_v}(1-q_c/2^{\lambda+1})+(1-\frac{\varepsilon}{q_v})(1-q_c/2^\lambda)/2=\frac{1}{2}+\frac{\varepsilon}{2q_v}-\frac{q_c}{2^{\lambda+1}}.$$

Therefore, we have

$$\frac{\varepsilon}{2q_v}-\frac{q_c}{2^{\lambda+1}}\leq \mathsf{Adv}_{\mathbb{B}}^{\mathsf{IND-PA0}}\leq \varepsilon_{\mathsf{PA0}},$$

hence $|S_6-S_5|=\varepsilon/2\leq q_v\varepsilon_{\mathsf{PA0}}+\frac{q_cq_v}{2^{\lambda+1}}$, where $q_v$ is the number of calls to $\mathcal{O}_{\mathsf{vote}}$ and $q_c$ the number of ballots in $BB$ which are not an output of $\mathcal{O}_{\mathsf{vote}}$.

**Conclusion.** Now, remark that due to the nature of $\mathcal{O}_{\mathsf{vote}}$ which overwrite $\mathsf{HV}$, it is clear that the adversary must call $\mathcal{O}_{\mathsf{check}}(i)$ between each call of $\mathcal{O}_{\mathsf{vote}}$ with the same voter $i$, otherwise $\mathsf{H}_i<1$ at line 6. Since $\mathcal{O}_{\mathsf{check}}$ erases $\mathsf{HV}_i$, it is also necessary for the adversary to call $\mathcal{O}_{\mathsf{cast}}(\mathbf{b})$ before $\mathcal{O}_{\mathsf{check}}(i)$, where $\mathbf{b}$ is the output of $\mathcal{O}_{\mathsf{vote}}(i,\nu)$. Note that due to the randomness involved in $\texttt{Vote}$, except with negligible probability $\mu$, the adversary cannot call $\mathcal{O}_{\mathsf{cast}}(\mathbf{b})$ before $\mathcal{O}_{\mathsf{vote}}(i,\nu)$ (at which point it has no information about $\mathbf{b}$). Therefore, the order of the revotes of each voter is enforced in $BB$ and, because of the last transition and the nature of the tally, we readily have that the condition at line 9 always result in a 0 output, which concludes the proof. □

## APPENDIX D
## QUANTIFYING COERCION-RESISTANCE

As explained in Section III, the coercion level can be evaluated thanks to Eq. (1). This assumes that the cryptography is perfect, that a large and unpredictable number of ballots is removed during the tally phase and that the adversary is able to compare $\Pr(\mathsf{R}^g|\alpha)$ and $\Pr(\mathsf{R}^g|\beta)$, given $\mathsf{R}^g$ with $g\in\{\mathsf{Real}^{\mathsf{CR}},\mathsf{Ideal}^{\mathsf{CR}}\}$. In this appendix, we give more details about how this comparison can be done as well as some efficient ways to evaluate the formula from Eq. (1). In particular, this allows to understand how the figures from Section III were obtained.

### A. The coercion level in the ideal game

In the ideal game, $\mathsf{R}^{\mathsf{Ideal}}$ is the vector $\overrightarrow{res}$, and computing $\Pr(\overrightarrow{res}|\alpha)$ for any $\alpha$ can be done thanks to a formula given in the following result from [25].

**Theorem 6** ( [25]). *Let $(\alpha,\beta)$ be two options, $n_H$ the number of honest voters and a pure result $\overrightarrow{res}$ such that $\sum_{i=0}^{C}\mathsf{res}_i=n_H+1$. Let $\overrightarrow{P}$ be the probability distribution for the honest voters. Assuming $P_\alpha P_\beta\neq 0$, we have $\Pr(\overrightarrow{res}|\beta)\geq\Pr(\overrightarrow{res}|\alpha)$ if and only if $\mathsf{res}_\beta P_\alpha\geq\mathsf{res}_\alpha P_\beta$.*

*In addition, we have*

$$\Pr(\overrightarrow{res}|\alpha)=\frac{n_H!}{\prod_{k=0}^{C}\mathsf{res}_k!}\left(\prod_{k=0}^{C}P_k^{\mathsf{res}_k}\right)\frac{\mathsf{res}_\alpha}{P_\alpha}.$$

*Finally, with $M_{\alpha,\beta}=\{\overrightarrow{res}\mid\Pr(\overrightarrow{res}|\beta)\geq\Pr(\overrightarrow{res}|\alpha)\}$, the optimal value of $\delta_{\alpha,\beta}^{\mathsf{Ideal}}$ is*

$$\delta_{\alpha,\beta}^{\mathsf{Ideal}^{\mathsf{CR}}}=\sum_{\overrightarrow{res}\in M_{\alpha,\beta}}(\Pr(\overrightarrow{res}|\beta)-\Pr(\overrightarrow{res}|\alpha))=\sum_{\overrightarrow{res}\in M_{\alpha,\beta}}\frac{n_H!}{\prod_{k=0}^{C}\mathsf{res}_k!}\left(\prod_{k=0}^{C}P_k^{\mathsf{res}_k}\right)\left(\frac{\mathsf{res}_\beta}{P_\beta}-\frac{\mathsf{res}_\alpha}{P_\alpha}\right).$$

Thanks to this result, it appears that the adversary can compare $\Pr(\mathsf{R}^{\mathsf{Ideal}}|\alpha)$ and $\Pr(\mathsf{R}^{\mathsf{Ideal}}|\beta)$ in $O(1)$ floating operations, as long as it has access to $P_\alpha$ and $P_\beta$ (note that it does not need the whole distribution), and where the asymptotic analysis is

done for $n_H$ growing to infinity and other parameters are fixed. We can further analyze this expression to give a representation which is easier to compute. For this purpose, we give Lemma 1, which allows to compute $\delta^{\mathsf{Ideal}}$ in $O(n_H)$ floating operations.

**Lemma 1.** *Let $(\alpha, \beta)$ be two options, $n_H$ the number of honest voters and a pure result $\overrightarrow{res}$ such that $\sum_{i=0}^{C} \mathsf{res}_i = n_H + 1$. Let $\overrightarrow{P}$ be the probability distribution for the honest voters. Assuming $P_\alpha P_\beta \neq 0$, we have*

$$\delta^{\mathsf{Ideal}^{\mathsf{CR}}}_{\alpha,\beta} = n_H! \sum_{x=0}^{n_H} \frac{T_x P_\beta^{T_x-1} P_\alpha^{N_x-T_x}}{x! T_x! (N_x - T_x)!} (1 - P_\beta - P_\alpha)^x, \text{ where } N_x = n_H - x + 1 \text{ and } T_x = \left\lceil \frac{P_\beta}{P_\beta + P_\alpha} N_x \right\rceil.$$

*Proof.* First, we partition $M_{\alpha,\beta}$ into $\bigcup_{x=0}^{n_H} M^x_{\alpha,\beta}$, where $M^x_{\alpha,\beta}$ is the subset of all results in $M_{\alpha,\beta}$ where the options other than $\alpha$ and $\beta$ received exactly $x$ votes. Then, we further partition $M^x_{\alpha,\beta}$ into subsets where $\mathsf{res}_\beta = y$ (and, thus, $\mathsf{res}_\alpha = N_x - y$). Note that due to the condition from Theorem 6, $\mathsf{res}_\beta P_\alpha \geq \mathsf{res}_\alpha P_\beta$ hence $y$ ranges from $T_x$ to $N_x$. To express the formula from Theorem 6 using this partition, we assume that $\alpha$ and $\beta$ are the last two voting options (otherwise we reorder them) and denote $\tilde{M}^x = \{\overrightarrow{res} \in \mathbb{N}^{C-2} \mid \sum_{k=0}^{C-2} \mathsf{res}_k = x\}$. With these notations, we have

$$\begin{aligned}
\delta^{\mathsf{Ideal}^{\mathsf{CR}}}_{\alpha,\beta} &= \sum_{x=0}^{n_H} \sum_{\overrightarrow{res} \in \tilde{M}^x} \sum_{y=T_x}^{N_x} \frac{n_H!}{\prod_{k=0}^{C-2} \mathsf{res}_k!} \left( \prod_{k=0}^{C-2} P_k^{\mathsf{res}_k} \right) \frac{P_\beta^y P_\alpha^{N_x-y}}{y!(N_x-y)!} \left( \frac{y}{P_\beta} - \frac{N_x - y}{P_\alpha} \right) \\
&= \sum_{x=0}^{n_H} \sum_{\overrightarrow{res} \in \tilde{M}^x} \frac{n_H!}{\prod_{k=0}^{C-2} \mathsf{res}_k!} \left( \prod_{k=0}^{C-2} P_k^{\mathsf{res}_k} \right) \sum_{y=T_x}^{N_x} \frac{P_\beta^y P_\alpha^{N_x-y}}{y!(N_x-y)!} \left( \frac{y}{P_\beta} - \frac{N_x - y}{P_\alpha} \right)
\end{aligned}$$

Now, we show that, for all $0 \leq T \leq N$, $\displaystyle\sum_{y=T}^{N} \frac{P_\beta^y P_\alpha^{N-y}}{y!(N-y)!} \left( \frac{y}{P_\beta} - \frac{N-y}{P_\alpha} \right) = \frac{T P_\beta^{T-1} P_\alpha^{N-T}}{T!(N-T)!}$.

For this purpose, we first fix some $N \geq 0$ and we proceed by backward iteration over $T$. First, it is true when $T = N$. Now, suppose that it is true for some $0 < T \leq N$; we show that it is also true for $T - 1$:

$$\begin{aligned}
\sum_{y=T-1}^{N} \frac{P_\beta^y P_\alpha^{N-y}}{y!(N-y)!} \left( \frac{y}{P_\beta} - \frac{N-y}{P_\alpha} \right) &= \frac{P_\beta^{T-1} P_\alpha^{N-T+1}}{(T-1)!(N-T+1)!} \left( \frac{T-1}{P_\beta} - \frac{N-T+1}{P_\alpha} \right) + \sum_{y=T}^{N} \frac{P_\beta^y P_\alpha^{N-y}}{y!(N-y)!} \left( \frac{y}{P_\beta} - \frac{N-y}{P_\alpha} \right) \\
&= \frac{P_\beta^{T-1} P_\alpha^{N-T+1}}{(T-1)!(N-T+1)!} \left( \frac{T-1}{P_\beta} - \frac{N-T+1}{P_\alpha} \right) + \frac{T P_\beta^{T-1} P_\alpha^{N-T}}{T!(N-T)!} \\
&= \frac{(T-1) P_\beta^{T-2} P_\alpha^{N-T+1}}{(T-1)!(N-T+1)!}.
\end{aligned}$$

Hence, with $T = T_x$ and $N = N_x$, we deduce that

$$\begin{aligned}
\delta^{\mathsf{Ideal}^{\mathsf{CR}}}_{\alpha,\beta} &= \sum_{x=0}^{n_H} \sum_{\overrightarrow{res} \in \tilde{M}^x} \frac{n_H!}{\prod_{k=0}^{C-2} \mathsf{res}_k!} \left( \prod_{k=0}^{C-2} P_k^{\mathsf{res}_k} \right) \frac{T_x P_\beta^{T_x-1} P_\alpha^{N_x-T_x}}{T_x!(N_x-T_x)!} \\
&= n_H! \sum_{x=0}^{n_H} \frac{T_x P_\beta^{T_x-1} P_\alpha^{N_x-T_x}}{x! T_x! (N_x-T_x)!} (1 - P_\beta - P_\alpha)^x \sum_{\overrightarrow{res} \in \tilde{M}^x} \frac{x!}{\prod_{k=0}^{C-2} \mathsf{res}_k!} \prod_{k=0}^{C-2} \left( \frac{P_k}{1 - P_\beta - P_\alpha} \right)^{\mathsf{res}_k} \\
&= n_H! \sum_{x=0}^{n_H} \frac{T_x P_\beta^{T_x-1} P_\alpha^{N_x-T_x}}{x! T_x! (N_x-T_x)!} (1 - P_\beta - P_\alpha)^x.
\end{aligned}$$

Indeed, the last summation is the sum over all possibilities of a multinomial distribution, which is 1. □

### *B. Modeling the real game*

For the real game where revoting is possible, we propose a model in which each honest voter does the following

- Abstain with probability $P_0$,
- Otherwise, vote for option $l > 0$ with probability $p_l$,
- Revote with probability $r_l$,
- In this case, choose option $k > 0$ with probability $r_{l,k}$.

In this model, a voter may revote at most once. Also, a voter who abstains does not "revote". If a voter who initially wanted to abstain changes their mind, it will count as voting once. This approximation is made because the adversary has access to the number of revotes, and not the number of voters who changed their mind. Also, the probability to revote depends on the

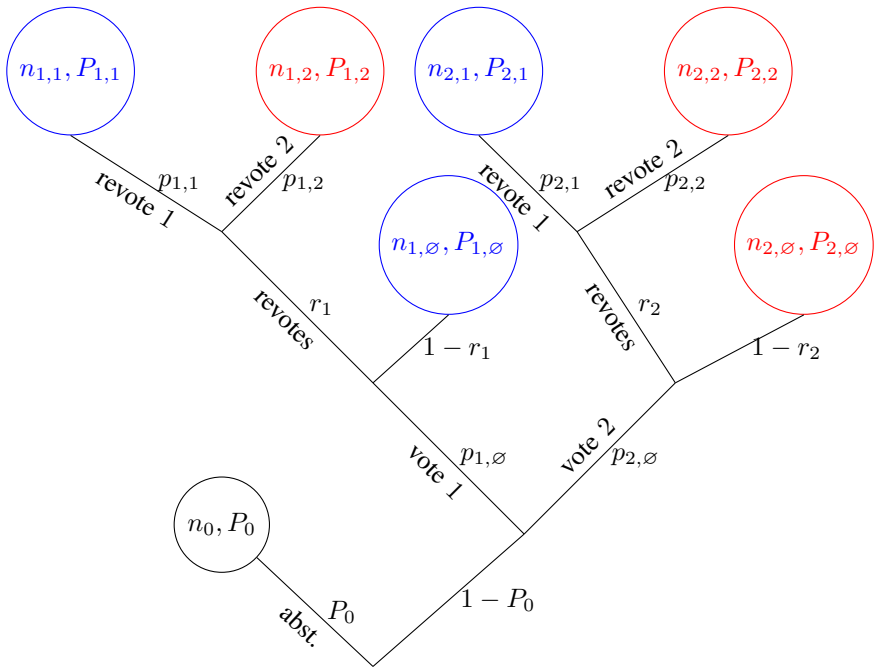


Fig. 11: Tree structure for two candidates.

initial choice. This is to capture the fact that an announcement on the press can make some voter revote if their candidate has been compromised. Similarly, the probability distribution when revoting depends on the first choice. See Fig. 11 for an illustration. In this figure, the label on the left of an edge denotes the nature of the transition and the label on the right denotes the probability of the transition. On leafs, we used labels of the form $(n, P)$ when $n$ denotes the total number of honest voters who choose this path and $P$ the probability to choose this path. For instance, $P_{1,2}$ is the probability to first vote for option 1 but to finally revote for option 2 while $P_{2,\varnothing}$ is the probability to vote for option 2 and not to revote. Since abstention plays a specific role, we denote it as choice 0, while the other choices actually imply to send a ballot.

Now, we must decide on the information the adversary has access to. The most conservative approach is that it may have access to anything which is not a secret; here, all the probability transitions. With this assumption, if $\overrightarrow{res} = (\mathsf{res}_0, \cdots, \mathsf{res}_C)$ is the number of votes for each option and $n_r$ is the number of revotes, then $\Pr((\overrightarrow{res}, n_r)|\alpha)$ is the probability that the honest voters votes $\mathsf{res}_k - \mathbb{1}_{k=\alpha}$ times for each candidate $k$ and revote $n_r$ times. Given the above tree, for any possible outcome $(\overrightarrow{res}, n_r)$ with $n_H$ honest voters, we have

$$\Pr(\overrightarrow{res}, n_r) = \sum_{H^{n_r}_{\overrightarrow{res}}} \frac{n_H!}{\mathsf{res}_0! \prod_{k=1}^{C} n_{k,\varnothing}! \prod_{l=1}^{C} n_{l,k}!} P_0^{\mathsf{res}_0} \prod_{k=1}^{C} P_{k,\varnothing}^{n_{k,\varnothing}} \prod_{l=1}^{C} P_{l,k}^{n_{l,k}}, \tag{2}$$

where $H^{n_r}_{\overrightarrow{res}} = \{(n_{k,\varnothing}), (n_{l,k}) \mid \sum_{l,k} n_{l,k} = n_r, \forall k > 0, n_{k,\varnothing} + \sum_{k=1}^{C} n_{l,k} = \mathsf{res}_k\}$ is the set of all possible atomic outcomes compatible with $\overrightarrow{res}$ and $n_r$.

### C. Quantifying the coercion level in some specific cases

Because of the complexity of Eq. (2), we only consider the specific cases where there are two candidates and the possibility to abstain (*i.e.* $C = 2$). In these simpler cases, we give Lemma 2 which allows to compute the coercion level in $O({n_H}^4)$ floating operations. Interestingly, $\delta^{\mathsf{Real}}$ does not depend on the $P_{l,k}$ directly but only on the sums $P_{r1}$ and $P_{r2}$, which correspond to the probability to revote for 1 or 2 respectively. This is because the adversary can only observe the number of revotes and has no information about the initial intention of the voter. Hence, they cannot exploit the dependency between the final choice and the first choice. In practice, we can imagine a scenario where the probability to revote and the probability distributions when revoting and when not revoting are known to the adversary, for instance thanks to exit poll or social media, where the voters tell whether they revote and what was their final choice. In this more realistic scenario when the adversary does not know all the probabilities on the tree, the latter can still compare $\Pr(\mathsf{R}^{\mathsf{Real}}|\alpha)$ and $\Pr(\mathsf{R}^{\mathsf{Real}}|\beta)$, which takes $O(n_H)$ floating operations.

**Lemma 2.** *Let $n_H$ be the number of honest voters, $(\mathsf{res}_0, \mathsf{res}_1, \mathsf{res}_2)$ such that $\mathsf{res}_0 + \mathsf{res}_1 + \mathsf{res}_2 = n_H + 1$ and $n_r \leq n_H - \mathsf{res}_0$. Let $\beta \in \{0, 1, 2\}$ be a voting option. Then the probability $\Pr((\mathsf{res}_0, \mathsf{res}_1, \mathsf{res}_2), n_r|\beta)$ that there are $\mathsf{res}_0$ abstentions, $\mathsf{res}_1$ votes for option 1, $\mathsf{res}_2$ votes for option 2 and $n_r$ revotes when the coerced voter chooses the voting option $\beta$ is given by the following algorithm.*

- *First, set $\overrightarrow{res} = (\mathsf{res}_0, \mathsf{res}_1, \mathsf{res}_2)$.*
- *If $\mathsf{res}_\beta = 0$, return 0, otherwise, do $\mathsf{res}_\beta \leftarrow \mathsf{res}_\beta - 1$.*
- *Set $N = n_H - \mathsf{res}_0 - n_r$, $P_{r1} = P_{1,1} + P_{2,1}$ and $P_{r2} = P_{1,2} + P_{2,2}$. Return*

$$\Pr(\overrightarrow{res}, n_r) = \frac{n_H! P_0^{\mathsf{res}_0}}{\mathsf{res}_0!} \sum_{n_{1,\varnothing}} \frac{P_{1,\varnothing}^{n_{1,\varnothing}} P_{2,\varnothing}^{N-n_{1,\varnothing}}}{n_{1,\varnothing}!(N-n_{1,\varnothing})!} \frac{P_{r1}^{\mathsf{res}_1 - n_{1,\varnothing}}}{(\mathsf{res}_1 - n_{1,\varnothing})!} \frac{P_{r2}^{n_r + n_{1,\varnothing} - \mathsf{res}_1}}{(n_r + n_{1,\varnothing} - \mathsf{res}_1)!},$$

*where* $\max(0, \mathsf{res}_1 - n_r) \leq n_{1,\varnothing} \leq \min(\mathsf{res}_1, n_H - \mathsf{res}_0 - n_r)$.

*Proof.* As shown in Section D-B, $\Pr((\mathsf{res}_0, \mathsf{res}_1, \mathsf{res}_2), n_r|\beta)$ is given by Eq. (2). We explicit the sum over $H_{\overrightarrow{res}}^{n_r}$. First, $0 \leq n_{2,\varnothing} \leq \mathsf{res}_2$ and $n_{1,\varnothing} + n_{2,\varnothing} = n_H - \mathsf{res}_0 - n_r$ and $\mathsf{res}_0 + \mathsf{res}_1 + \mathsf{res}_2 = n_H$, therefore

$$\begin{aligned} 0 \leq & n_{1,\varnothing} \leq \mathsf{res}_1 \\ \mathsf{res}_1 - n_r \leq & n_{1,\varnothing} \leq n_H - \mathsf{res}_0 - n_r. \end{aligned}$$

In addition, $n_{1,\varnothing} + n_{1,1} + n_{2,1} = \mathsf{res}_1$ and $0 \leq n_{2,1} \leq n_r$ so that

$$\begin{aligned} 0 \leq & n_{1,1} \leq n_r \\ \mathsf{res}_1 - n_{1,\varnothing} - n_r \leq & n_{1,1} \leq \mathsf{res}_1 - n_{1,\varnothing}. \end{aligned}$$

Finally, $n_{1,2} + n_{2,2} = n_r - n_{1,1} - n_{2,1} = n_r + n_{1,\varnothing} - \mathsf{res}_1$, thus

$$0 \leq n_{1,2} \leq \mathsf{res}_2 - n_{2,\varnothing} = n_r + n_{1,\varnothing} - \mathsf{res}_1$$

Let $N = n_H - \mathsf{res}_0 - n_r$, $N_{n_{1,\varnothing}} = \mathsf{res}_1 - n_{1,\varnothing}$ and $N'_{n_{1,\varnothing}} = n_r + n_{1,\varnothing} - \mathsf{res}_1$. Considering all these inequalities, we have

$$\begin{aligned} \Pr(\overrightarrow{res}, n_r) &= \frac{n_H! P_0^{\mathsf{res}_0}}{\mathsf{res}_0!} \sum_{n_{1,\varnothing}} \frac{P_{1,\varnothing}^{n_{1,\varnothing}} P_{2,\varnothing}^{N-n_{1,\varnothing}}}{n_{1,\varnothing}!(N-n_{1,\varnothing})!} \sum_{n_{1,1}} \frac{P_{1,1}^{n_{1,1}} P_{2,1}^{N_{n_{1,\varnothing}} - n_{1,1}}}{n_{1,1}!(N_{n_{1,\varnothing}} - n_{1,1})!} \sum_{n_{1,2}} \frac{P_{1,2}^{n_{1,2}} P_{2,2}^{N'_{n_{1,\varnothing}} - n_{1,2}}}{n_{1,2}!(N'_{n_{1,\varnothing}} - n_{1,2})!} \\ &= \frac{n_H! P_0^{\mathsf{res}_0}}{\mathsf{res}_0!} \sum_{n_{1,\varnothing}} \frac{P_{1,\varnothing}^{n_{1,\varnothing}} P_{2,\varnothing}^{N-n_{1,\varnothing}}}{n_{1,\varnothing}!(N-n_{1,\varnothing})!} \sum_{n_{1,1}} \frac{P_{1,1}^{n_{1,1}} P_{2,1}^{N_{n_{1,\varnothing}} - n_{1,1}}}{n_{1,1}!(N_{n_{1,\varnothing}} - n_{1,1})!} \frac{(P_{1,2} + P_{2,2})^{N'_{n_{1,\varnothing}}}}{N'_{n_{1,\varnothing}}!} \end{aligned}$$

Now, $\mathsf{res}_1 - n_{1,\varnothing}$ is equal to the number of revotes for 1, so that $\mathsf{res}_1 - n_{1,\varnothing} \leq n_r$. Therefore, the innermost summation is also on the full domain for $n_{1,2}$. Hence,

$$\begin{aligned} \Pr(\overrightarrow{res}, n_r) &= \frac{n_H! P_0^{\mathsf{res}_0}}{\mathsf{res}_0!} \sum_{n_{1,\varnothing}} \frac{P_{1,\varnothing}^{n_{1,\varnothing}} P_{2,\varnothing}^{N-n_{1,\varnothing}}}{n_{1,\varnothing}!(N-n_{1,\varnothing})!} \frac{(P_{1,1} + P_{2,1})^{N_{n_{1,\varnothing}}}}{N_{n_{1,\varnothing}}!} \frac{(P_{1,2} + P_{2,2})^{N'_{n_{1,\varnothing}}}}{N'_{n_{1,\varnothing}}!} \\ &= \frac{n_H! P_0^{\mathsf{res}_0}}{\mathsf{res}_0!} \sum_{n_{1,\varnothing}} \frac{P_{1,\varnothing}^{n_{1,\varnothing}} P_{2,\varnothing}^{N-n_{1,\varnothing}}}{n_{1,\varnothing}!(N-n_{1,\varnothing})!} \frac{P_{r1}^{\mathsf{res}_1 - n_{1,\varnothing}}}{(\mathsf{res}_1 - n_{1,\varnothing})!} \frac{P_{r2}^{n_r + n_{1,\varnothing} - \mathsf{res}_1}}{(n_r + n_{1,\varnothing} - \mathsf{res}_1)!}, \end{aligned}$$

where $P_{r1} = P_{1,1} + P_{2,1}$ and $P_{r2} = P_{1,2} + P_{2,2}$.

Note that this can be computed in $O(n_H)$ floating operations. Since we must range over all possibilities for $\mathsf{res}_0, \mathsf{res}_1, \mathsf{res}_2, n_r$, the overall $\delta^{\mathsf{Real}}$ can be computed in $O({n_H}^4)$ floating operations. □

In Section III, we used the formulas from Lemma 2 and 1 to evaluate the coercion level. The only exceptions are Figures 16 and 6, where the real coercion level for 513 and 1025 voters were obtained with a Monte-Carlo estimates with 1000000 iterations, which gives a sufficiently small confidence interval for our purpose. The reason why is that evaluating the formula from Lemma 2 would be too long for such values of $n_V$.

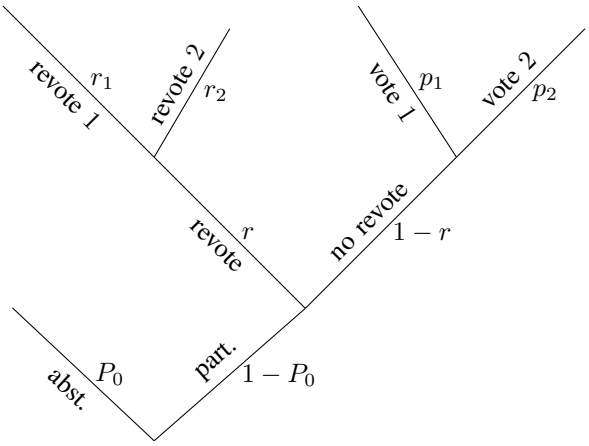


Fig. 12: Simplified tree structure for two candidates.

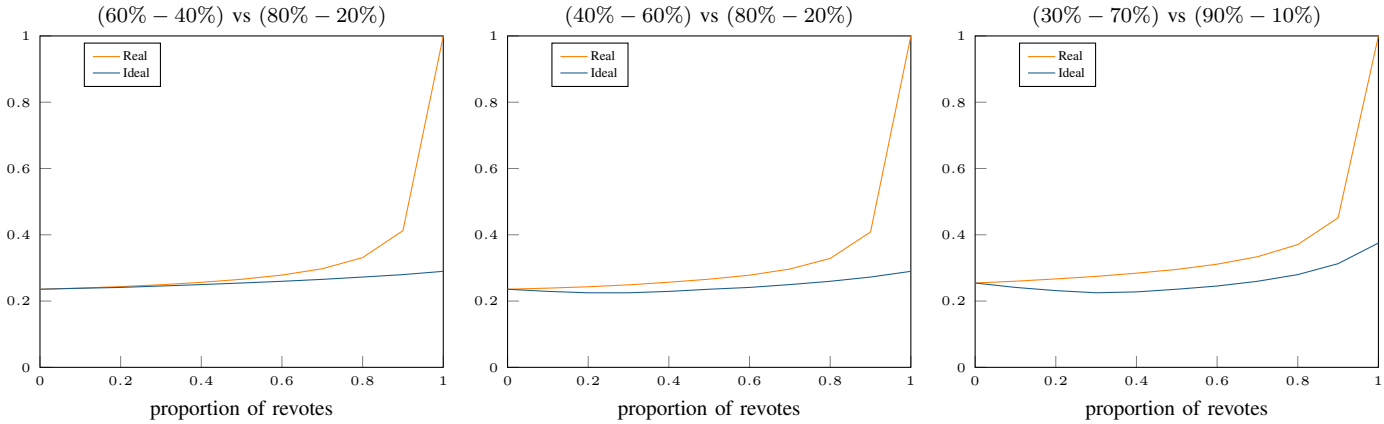


Fig. 13: Coercion levels as a function of the probability of revoting; with 20 voters, 2 candidates and 30% abstention, with three different distributions between the candidates.

### *D. The impact of the parameters*

In Lemma 2, we saw that $\delta^{\mathsf{Real}}$ only depends on $P_0$, the probability of abstention, $P_{r1}$ (resp. $P_{r2}$), the probability to revote for option 1 (resp. 2) and $P_{1,\varnothing}$ (resp. $P_{2,\varnothing}$), the probability to vote for option 1 (resp. 2) without revoting. To give a better representation of these parameters, we define

$$r = (P_{r1} + P_{r2})/(P_{r1} + P_{r2} + P_{1,\varnothing} + P_{2,\varnothing}),$$

which is the probability to revote (after voting),

$$p_1 = P_{1,\varnothing}/(P_{1,\varnothing} + P_{2,\varnothing}) \text{ and } p_2 = P_{2,\varnothing}/(P_{1,\varnothing} + P_{2,\varnothing}),$$

which are the probabilities to vote for option 1 and option 2 when voting exactly once and

$$r_1 = P_{r1}/(P_{r1} + P_{r2}) \text{ and } r_2 = P_{r2}/(P_{r1} + P_{r2}),$$

the probabilities to vote for option 1 and 2 when revoting (see Fig. 12 for an illustration).

A first important parameter is $r$, the probability of revoting. In Fig. 13, we let it vary from 0 to 1, and we plot the coercion level in JCJ and in the ideal protocol for various values of $(p_1, p_2)$ and $(r_1, r_2)$. As expected, since the leakage we detected comes from the revotes, both are the same when there is no revote. Note that when everyone revotes, there is no coercion resistance anymore. This is due to the considered evasion strategy, which instructs the voter to vote once (which is detected if they are the only one to do so).

Now, an important point to notice is that, if $(p_1, p_2)$ and $(r_1, r_2)$ are close, the difference between the ideal and the real coercion level is small, unless there are a lot of revotes. On the other hand, if both distributions are opposite, the difference is

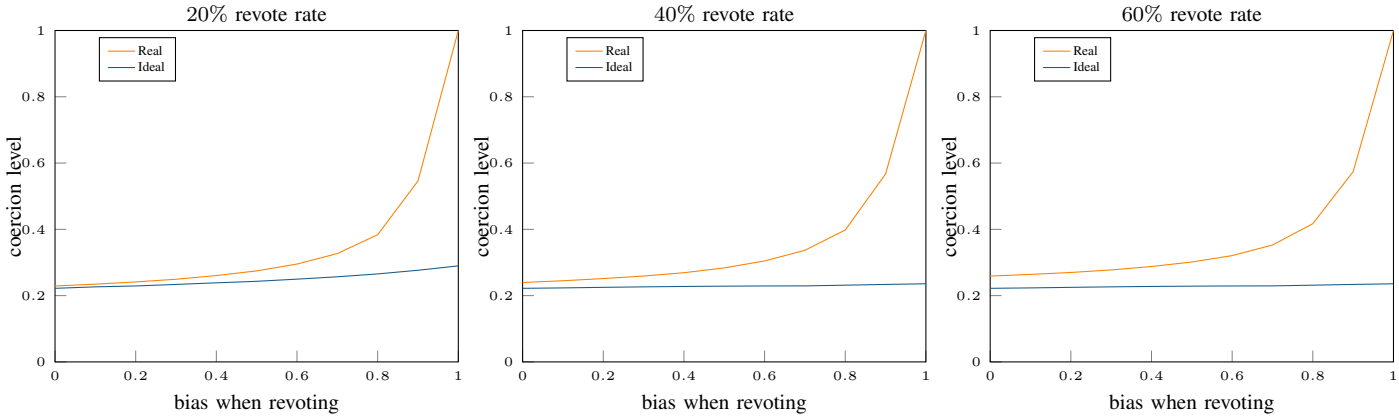


Fig. 14: Coercion levels as a function of the bias when revoting; with 20 voters, 2 candidates and 30% abstention.

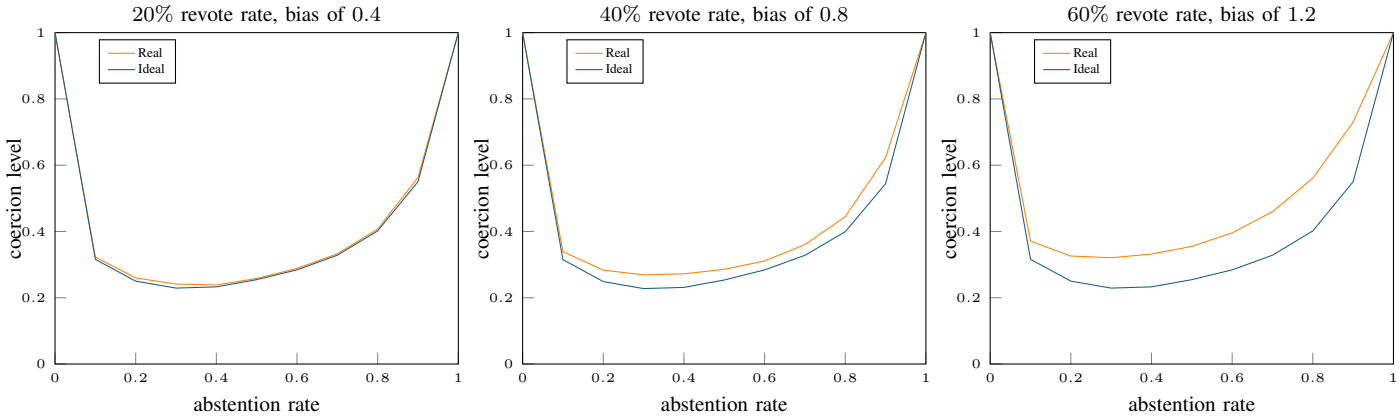


Fig. 15: Coercion level as a function of the abstention rate; with 20 voters and 2 candidates.

noticeable, even with a reasonable proportion of revotes. Therefore, another interesting parameter to consider is the distance between $(p_1, p_2)$ and $(r_1, r_2)$, defined as $2|p_1 - r_1|$, which we call the *bias*. In Fig. 14, we let the bias vary from 0 to 2 and we plot the coercion level in the real and ideal protocols, for various probability of revoting. As expected, when the bias is maximal, there is no coercion resistance at all (this corresponds to the scenario of Section II-B). Note, however, that the leakage is non-zero when there is no bias. This, once again, is due to the fact that the adversary can count the number of revotes, and therefore have a non-negligible advantage to decide whether the coerced voter voted or abstained.

Another natural parameter to consider is the abstention rate. Until now, it was fixed at 0.3 in our experimentations. However, it can variate a lot in real-life elections. In Fig. 15, we plot the real and ideal coercion levels as a function of the abstention rate, with various probabilities of revoting and bias. As expected, when there is no abstention, the attacker can trivially break coercion-resistance with a forced-abstention attack. Similarly, when everyone but the coerced voter abstains, there is no coercion-resistance (both situations are captured by the ideal model).

Finally, the last parameter of interest is the number of honest voters. In Fig. 16, we plot the real and ideal coercion levels for $16, 32, 64, 128$, 256, 512 and 1024 honest uncoerced voters, with various probabilities of revoting and bias. As expected, the coercion level decreases as $n_H$ grows for both the ideal and real game. Interestingly, the difference remains noticeable as long as there is enough revotes.

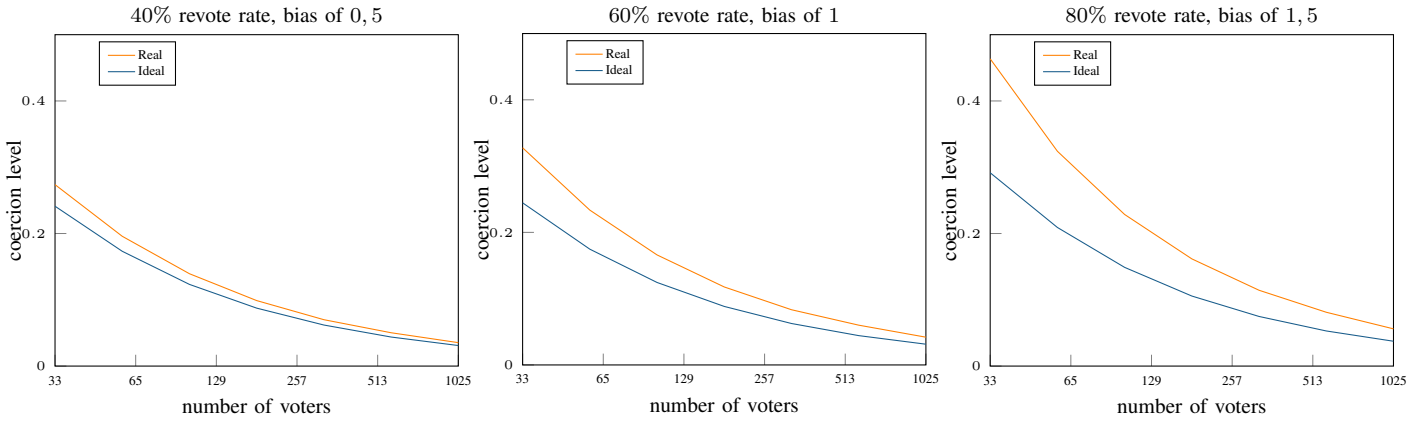


Fig. 16: Coercion level as a function of the number of voters; with 2 candidates and 30% abstention.