Is the JCJ voting system really coercion-resistant?

Véronique Cortier
Université de Lorraine, CNRS, Inria
LORIA, Nancy, France

Pierrick Gaudry
Université de Lorraine, CNRS, Inria
LORIA, Nancy, France

Quentin Yang
Université de Lorraine, CNRS, Inria
LORIA, Nancy, France

***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.**

***Index Terms*—E-voting, coercion-resistance.**

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. However, compared to a paper-based solution, 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 seminal protocol which aims to counter coercion was proposed in 2005 by Juels, Catalano and Jakobsson [23]. 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. Hence JCJ aims at offering coercion-resistance in a context where voters:

- are not fully controlled by the coercer and may cast their true vote at some point, when they are not under coercion;
- may access to a tool (e.g. a free online website) that generates randoms fake credentials of the expected format;
- did receive their valid credential at an earlier step, outside the control of the attacker.

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 Δ 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 Δ. For instance, in [31], 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 have discovered that even in the original JCJ protocol, the cleansing step leaks more than the difference Δ between the sizes of its input and output. First, the ballots with the same credentials (*i.e.* revotes) are handled, keeping only one ballot per credential. Second, the ballots that use an invalid credential are removed. The size of the intermediate ballot box is leaked; moreover, anyone can observe the distribution of the number of revotes. Those

This work benefited from funding managed by the French National Research Agency under the France 2030 programme with the reference ANR-22-PECY-0006.

pieces of information can be analyzed by the coercer and help to determine if a coerced voter obeyed or not. In particular, we provide a few examples where this leakage allows the coercer to fully break coercion-resistance, even if dummy ballots are used. Admittedly, our examples are rather extreme scenarios, that are unlikely to happen. 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, inducing 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: using their a priori knowledge about the behavior of the honest voters, the coercer can use bayesian inference to decide whether it is more likely that the voter obeyed or not. In order to 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 Δ. All the variants and improvements on JCJ that we know of are also affected by this vulnerability. Note that our attacks assume that the coercer knows a precise description of the expected behavior of honest voters, which includes the probability of revoting. In practice, a reasonably good approximation can be obtained thanks to social media or exit polls.

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 Δ 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. One of the probable reasons why the leakage in JCJ was not noticed before is because their definition of coercion-resistance was too weak. A flaw was already noted in [18], which shows that the definition could not be realized. The fix proposed in [18] 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.

We provide the detailed specification of CHide and the full proofs in a companion report [13].

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 [31], 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 [32]. Other improvements include [3], 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 suffer from the same cleansing leakage as JCJ, as we further discuss in Section VI.

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 [28] completely relies on revoting to mitigate coercion, and examples of recent academic proposals based on revoting are VoteAgain by Lueks *et al.* [27] and the scheme of Locher *et al.* [26]. This approach assumes a weaker adversary which can no longer submit ballots past 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 [6], [12], [20], [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 [18] 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 δ-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 δ. Our definition of coercion-resistance can be seen as an instance of [25] where we compare the δ-coercion-resistance of the real protocol with the δ'-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 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 [22]. 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.

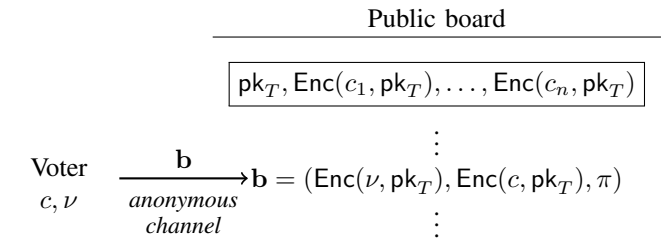

Fig. 1: Voting phase in JCJ.

Voting. To cast a ballot, a voter encrypts their vote ν with the election public key pk_T. They also encrypt their credential c and prove knowledge of ν and c. They prove that the encryptions are valid and linked, yielding a proof π. 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 [21] (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. Remark that the PETs originally used in JCJ were not verifiable when all the participants are corrupted, as stated in [29] which also proposes a fix.

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 provides her coercer with a random (and fake) credential c'. At any moment, Alice can 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.

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 [12], [20], [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.

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, and suppose that the coercer knows this. 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) The information available to the adversaries: To succeed in the above scenario, the adversary must know that the honest voters who revote always vote for B while the others always vote for A. In what follows, we consider less extreme scenarios when voters do not have such a deterministic revoting behavior but we still assume that the coercer knows the probability that the voters vote for A (or B) when they revote or when they do not. Such an information can be learned for instance thanks to a poll where the voters tell what was their final choice and whether they revoted. We leave as future work the quantitative evaluation of the leakage when the adversary only knows some approximation of the distribution.

2) 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.

3) 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, [31] uses this to mitigate a leakage during the tally. The number of fake ballots to add should however be carefully calibrated. Indeed, the more fake ballots are added, the better coercion-resistance is improved. However. adding fake ballots induces an important computation overhead at the tally phase. 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 explained 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. The generic assumptions that we make are given in Section III-A; the two scenarios that we consider are detailed in Section III-B and Section III-C.

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. We suppose that the choices follow a probability distribution $(P_0, \ldots, P_C)$, where P_0 is the probability to abstain. Let α be the voting option corresponding to the intention of the coerced voter, and β be the one that is the instruction of the coercer. The coerced voter either disobeys,

gives a fake credential and votes with option α (the evasion strategy does not imply any revote), or obeys and gives their real credential which the coercer uses to vote with option β. 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 (α, β), 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 δ^{Real} (resp. δ^{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 R^g is obtained, assuming the voter votes for α (resp. obeys the coercer and votes for β). 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

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

where $M_{\alpha,\beta}$ is the set of all possible results R^g such that $\Pr(\mathsf{R}^g|\beta) \geq \Pr(\mathsf{R}^g|\alpha)$. In [13], we explain in more details how this formula can be computed. In the remainder of this section, we compare δ^{Real} and δ^{Ideal} in two scenarios, where external events provoke many revotes. In these scenarios, we assume that there are two candidates A and B, no blank vote, but the possibility to abstain or to revote once. Also, we consider revoting to be rare. Changing a vote once casted is something that is typically not allowed in classical paper-based elections. So, 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% [15]. We sum-up the assumptions as follows.

- The honest voters behave independently from each other;
- The honest voters may revote at most once;
- The honest voters would usually not revote, unless a specific event encourages them to do so;
- The adversary knows the vote (and revote) distribution;
- The cryptography is perfect, so that we can focus on the information leaked by the protocol;
- There is a large and unpredictable number of ballots sent with a fake credential, so that the adversary does not learn any information by observing the number of ballots removed because of an invalid credential;
- We focus on a situation where there are only two candidates and where the adversary tries to coerce a single candidate.

B. The technical incident scenario

Although we consider natural revoting to be rare, many revotes can occur if an announcement is made about a technical incident, and encourages 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 the situation in which 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

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 of the voters who initially voted for A will change their mind and revote for B.

Note that 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).

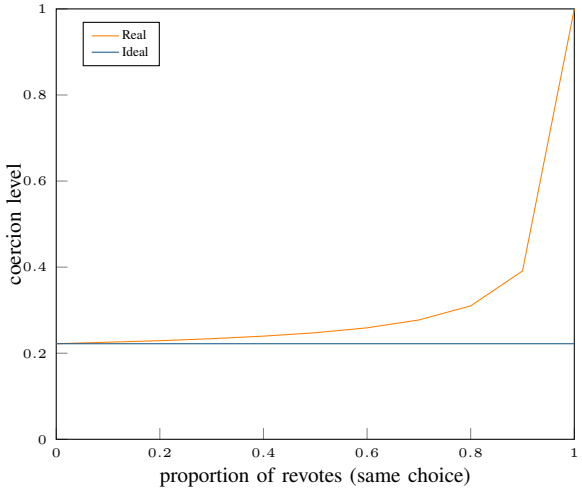

Fig. 2: Coercion levels as a function of the revote, with 20 voters, 2 candidates, 30% abstention and a 50%-50% distribution of votes between the candidates.

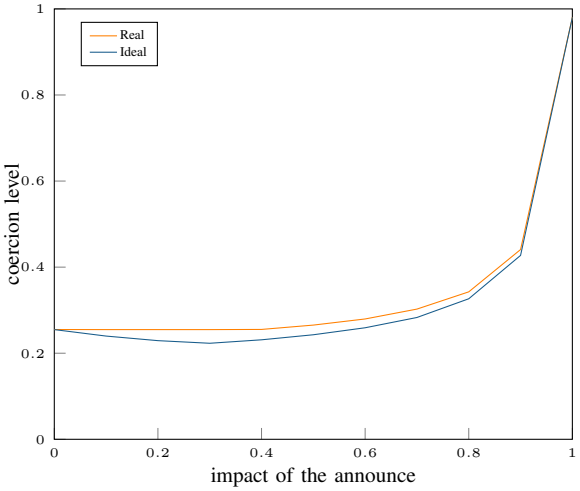

Fig. 3: Coercion levels as a function of the impact for 20 voters, 2 candidates, 30% abstention and a 70%-30% distribution between the candidates.

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.* [14], which also uses the quantitative framework of [25]) 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 they correspond to revotes or to invalid credentials. For this purpose, we propose a novel cleansing algorithm based on MPC primitives.

Interestingly, some alternatives to the JCJ protocol were already presented in the literature with the purpose of reducing the leaked information. For instance, Caveat coercitor [17] proposes a protocol in which the PETs do not reveal the total number of revotes for each credential, since the process is stopped as soon as one match is found. However, this approach is not designed in order to completely remove the leakage: even if we stop as soon as a match is found, how fast we stop still gives some information about how many matches there is.

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 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 $\mathsf{Enc}(m, \mathsf{pk}) = (g^r, g^m\, \mathsf{pk}^r)$. In general, recovering m from g^m is hard so that exponential ElGamal requires m to be in a small list of valid messages, so that decryption is feasible. An important special case is when m is a bit, because this is the kind of input used by the MPC primitives

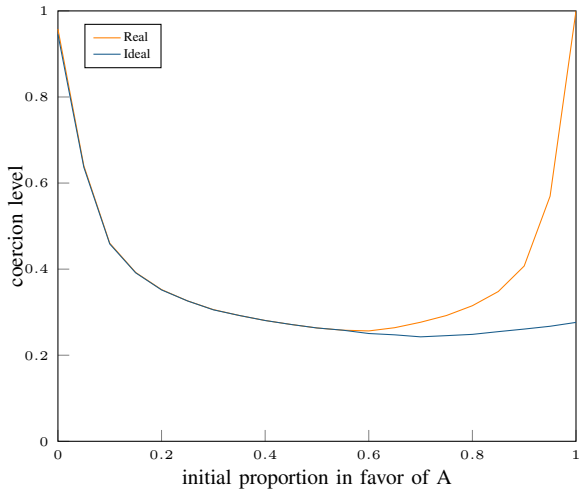

Fig. 4: Coercion levels as a function of the proportion in favor of A for 20 voters, 2 candidates and 30% abstention.

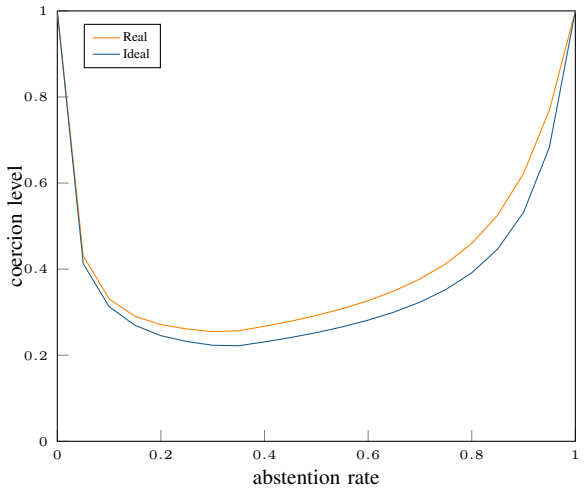

Fig. 5: Coercion levels as a function of the abstention for 20 voters, 2 candidates, 21% revotes and a distribution of 70%-30% between the candidates.

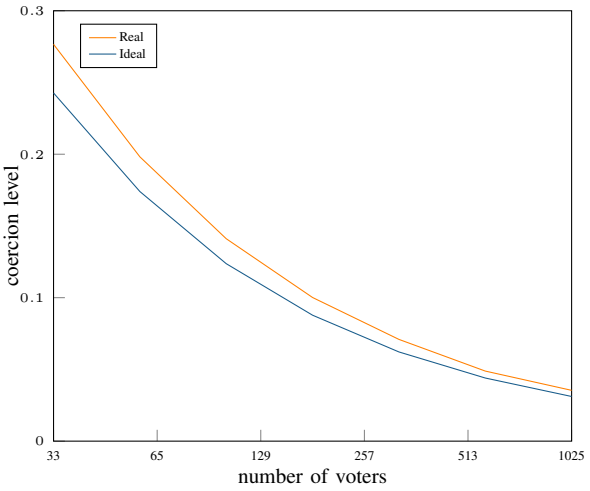

Fig. 6: Coercion levels as a function of the number of voters with 30% abstention, 21% revotes and a distribution of 70%-30% between the candidates.

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 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. An UC-secure instantiation is given in [35]; a more popular DKG protocol is described in [16].

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. An UC-secure instantiation is given in [34]; a more popular protocol is described in [36], that is the basis of the Verificatum library [33].

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 CGate protocol [30], 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 And (conjunction) and the 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 [13] for more details about

these MPC protocols.

Sorting encrypted data. Using the above logical operations, it is straightforward to design a comparison test Lt and a conditional swap 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. Since popular fast sorting algorithms, such as Quicksort, Mergesort or Heapsort, do not have this property, we use OddEvenMergeSort by Batcher [5], 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 the trust assumptions on them, and note that they 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 [19] for a possible realization of this.
- 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 λ is chosen. The election trustees jointly run the DKG protocol, yielding a public key 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 λ 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 [22] 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.* [8]).

Voting phase. In order to cast a vote for the option ν (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 π_1 that proves the knowledge of ν and c_j for all j. To ensure a strong Fiat-Shamir transformation [7], the computation of the challenge from the commitment of the Σ-protocol must include all public informations in the hash, such as g, pk, C_1 and C_2. Finally, a ZKP π_2 that ν 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 $(C_1, C_2, \pi_1, \pi_2) = \mathtt{Vote}_{\mathsf{pk}}(c, \nu)$ 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 [12], 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 [12] 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

TABLE I: Asymptotic cost of the CHide protocol in terms of computations and communications, n_B is the size of the board, n_V is the number of voters, λ is the security parameter and n_T is the number of talliers.

# exponentations	$O((n_B+n_V)\log(n_B+n_V)^2(\log n_B+\lambda)n_T)$
Data exchanged	$O(\lambda(n_B+n_V)\log(n_B+n_V)^2(\log n_B+\lambda)n_T)$

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 [13].

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 Π^{Sort}.
5. For all $1 \leq k < n_B + n_V$, compute $D_k = \mathsf{Eq}({K'}_k^\top, {K'}_{k+1}^\top)$, where ${K'}_k^\top$ refers to the λ most significant bits of K'_k. This produces the transcript $\Pi_{k,1}^{\mathsf{Eq}}$.
6. For all $1 \leq k < n_B + n_V$, compute $F_k = \mathsf{Eq}({K'}_{k+1}^\perp, U_{n_B})$, where ${K'}_{k+1}^\perp$ refers to the m least significant bits of K'_{k+1}. This produces the transcript $\Pi_{k,2}^{\mathsf{Eq}}$.
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 Π^{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.

C. Complexity and efficiency considerations

On the tallier side, CHide has a quasi-linear complexity in $n_B + n_V$, both in terms of computational cost and in term of communications. Due to the fact that the encrypted credentials require λ ciphertexts instead of a single one, a factor λ occurs multiplicatively in these complexities. We give the precise asymptotic complexities in Table I. In this table, for the computational cost, we count the number of exponentiations, which is the dominant part, and for the communications, we give the total volume of the messages exchanged (in bits, hence the additional λ factor for the communications, to take into account the bit-size of the group elements and of the exponents). Note that, in our case, this value is the same as the size of the transcript that an external verifier would need to download in order to verify the result.

TABLE II: Estimated number of exponentiations and volume of communication in JCJ and CHide, with $\lambda = 128$.

# voters	# exp.		estimated CPU time		data exchanged	
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

For the voters, forming a ballot has a complexity of $O(\lambda)$ exponentiations for the encryptions and the zero-knowledge proofs, due, again, to the credential that is encrypted bit by bit.

Let us now consider the practical cost for non-asymptotic parameters. For this purpose, we fix a number of $n_T = 3$ talliers with a threshold $t = 2$, a security parameter of $\lambda = 128$ and a number of $n_C = 2$ voting options. We can estimate the running times, based on an approximation of 5,000 exponentiations per second on the client side, and 10,000 per second on the server side. For the voters, the time for preparing the ballot remains in the realm of the second, which is satisfactory. For the talliers, as expected from the complexity, the time grows a bit faster than linearly with the number of voters, and we estimate that for an election of 1000 voters, the tally would require 2.8 days of computation on a single core, and can easily be parallelized on a few dozens of cores to turn this into a couple of hours. The amount of data to be exchanged would be around 39 GB. Assuming a 100 Mbit/s connection between the trustees, this would take about one hour. Furthermore, the CHide MPC protocol has only few strong synchronization points, so that communications and computations can be done in parallel. In Table II, we give estimates of the number of exponentiations and of the total volume exchanged for both JCJ and CHide, for various number of voters. This allows to check the quasi-linear cost of CHide.

Table II also illustrates how CHide compares to JCJ. This confirms that the bit-wise encryption of the credential induces

a heavy cost for CHide, that is balanced for large elections by the fact that JCJ has a quadratic compexity. The cross-over point is between 100,000 and 1,000,000 voters, according to our estimates. For such very large elections, 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.

Of course, CHide remains slower than the variants of JCJ that have a linear complexity, like [2]. However, these other schemes also suffer from some leakage similar to JCJ's. We discuss these in Section VI.

Finally, we mention that a recent work by Arahna et al. [1] proposes a variant of CHide that improves its efficiency.

V. Defining coercion-resistance

We guess that one of the reasons why the flaw was not discovered in the original proposition of the JCJ protocol is 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 λ and the threshold t. It computes the public key 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 pk, a credential c, and a vote ν, 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 ν. 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 pk, the threshold t and returns the result $\boldsymbol{X}$ of the election together with a proof Π.
- $\texttt{Verify}(BB, \Pi, \mathsf{res})$ takes as input a ballot box BB, a transcript Π and a result 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 ϕ 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, α), where j denotes the voter under coercion and α 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 α (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 $\nu \in [1, n_C]$;
- ϕ, which represents abstention;
- λ, which represents casting a vote with a fake credential.

We extend the `Vote` function to votes equal to λ as follows.

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

where $\tilde{c} = \texttt{Fakecred}(c)$ and ν 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 unpopular 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 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 [23])**.** *A voting system is* JCJ-coercion resistant *if for all PPT adversary $\mathbb{A}$, for all parame-*

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. λ 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.

ters 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 μ 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 noted in [18], 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 Δ 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 [23].

The authors of [18] 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 [18] for a survey). The most prominent one is the quantitative definition of [25] (KTV), where the notion of δ-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 δ, 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 δ is acceptable or not. Our definition can be seen as an instance of KTV, where δ 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, ν) 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, ν) 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, ν) 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 β (if $\beta \neq \phi$) and provides a fake credential. Hence the votes from j in $\mathcal{B}$ are replaced by a single vote for β (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 μ 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 [13]. In this theorem, we suppose secure DKG and Mixnet protocols in the SUC security framework [9]. 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 using a similar approach (full proofs in [13]).

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

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

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

$BB \longleftarrow \varnothing$
$\mathsf{pk}, (s_i, h_i)_{i=1}^{n_T} \longleftarrow P^{\mathbb{A}}_{\mathtt{Setup}}(\lambda, n_T, t)$ (* run the DKG *)
$\{c_i; i \in [1, n_V]\}, \boldsymbol{R} \longleftarrow \mathtt{register}(\lambda, \mathsf{pk}, n_V)$
$A \longleftarrow \mathbb{A}(\boldsymbol{R})$ (* corrupt voters *)
$(j, \alpha) \longleftarrow \mathbb{A}(\{c_i; i \in A\})$
(* coerce a voter j who has the intention α *)
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, ν_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**
  $c_i \longleftarrow \mathtt{Fakecred}()$
  (* 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, α) if $\alpha \neq \phi$ and removed otherwise
  $\tilde{c} \longleftarrow \mathtt{Fakecred}(c_j)$
$\mathbb{A}(\tilde{c})$ (* $\mathbb{A}$ learns $\tilde{c}$ *)
for $(i, \nu_i) \in B$ *(in this order)* **do**
  $\boldsymbol{M} \longleftarrow \mathbb{A}(BB)$ (* cast ballots *)
  $BB \longleftarrow BB \bigcup \{m \in \boldsymbol{M} \mid \mathtt{isVal}(\mathsf{pk}, m, BB) = 1\}$
  $BB \leftarrow BB \bigcup \{\mathtt{Vote}_{\mathsf{pk}}(c_i, \nu_i)\}$
$\boldsymbol{M} \longleftarrow \mathbb{A}(BB, \texttt{"last honest ballot sent"})$
$BB \longleftarrow BB \bigcup \{m \in \boldsymbol{M} \mid \mathtt{isVal}(\mathsf{pk}, m, BB) = 1\}$
$\boldsymbol{X}, \Pi \leftarrow P^{\mathbb{A}}_{\mathtt{tally}}(BB, \boldsymbol{R}, \mathsf{pk}, \{h_i, s_i\}, t)$
$b' \longleftarrow \mathbb{A}()$
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 α *)
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, ν_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, α) 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**
  $B \longleftarrow B \bigcup \{(j, \beta)\}$
$B \longleftarrow B \bigcup \{(i, \nu_i) \mid i \in A, \nu_i \in [1, n_C]\}$
$\boldsymbol{X} \longleftarrow \mathtt{result}(\mathtt{cleanse}(B))$
$b' \longleftarrow \mathbb{A}(\boldsymbol{X})$
Return 1 **if** $b' == b$ **else** 0

Fig. 8: Definition of coercion-resistance. λ 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.

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. Leakage of JCJ variants

The JCJ scheme is not the only one affected by our finding. In fact, the whole JCJ family suffers from a similar leakage. In this section, we explain how the other JCJ-like schemes are affected, and whether an MPC version of their protocol can be used to address the leakage while preserving their efficiency.

We start with the scheme presented by Araújo, Foulle and Traoré (AFT) in [3]. Its main feature is that it has a linear time complexity for the cleansing and tallying phases. While they use different cryptographic primitives from JCJ, their scheme has a similar structure: voters are given credentials to vote with, and can provide a fake credential to a coercer. Assuming that the cryptography is perfect, we can analyze their leakage and compare it with that of JCJ.

During the tally, both the number of duplicates and the number of ballots with an invalid credential are revealed. However, it is possible to deduce, by observing the board, how many revotes each ballot has. This is because each ballot is attached with a public (and deterministic) commitment of the credential, so that anyone can detect that two ballots use the same credential. In JCJ, this information is only available during the tally, when it is no longer possible for the adversary to submit a ballot (also, the ballots are shuffled beforehand).

Hence, the AFT suffers from the same leakage as in JCJ, but it might be easier to exploit since the adversary can use this information to cast their ballots in some specific way.

By contrast with the JCJ protocol, it is not easy to adapt the AFT scheme in MPC while preserving its time complexity. Indeed, one of the key feature is that anyone can detect that two ballots use the same credential, so that the duplicated credentials can be removed in linear time thanks to a hash map. In MPC, we cannot think of a way to adapt this strategy while preserving the linear time complexity: the most natural way would take a quadratic complexity, and a more advanced strategy, based on sorting, would achieve a complexity similar to that of CHide. Note that the above remarks about the leakage and the difficulty to adapt the scheme in MPC are also valid for the subsequent schemes such as [4] and [2].

A similar situation is encountered when analyzing the proposal of Spycher *et al.* [31]. First, the revotes are taken care of during a phase which also relies on detecting the duplicated values from a list of cleartexts. Second, the invalid credentials are eliminated thanks to PETs; however, this only requires one PET per ballot because each ballot indicates to which encrypted credential it must be compared with. Just as for the AFT protocol, this scheme leaks the same information as JCJ (namely the number of revotes per credential), but also leaks how many ballots pretend to be from each credential. As acknowledged by the authors, this extra information deteriorates the coercion-resistance of the scheme when compared to that of JCJ. Note that adapting this solution in MPC would also be difficult: first, we have the same difficulty as when trying to adapt the solution of AFT; second, we do not see how to eliminate the invalid credentials in linear time without leaking the number of ballots attached to each credential.

Finally, another interesting example is Civitas [11], a scheme considered as an implementation of JCJ. Among the few differences that could have an impact, we concentrate on the leakage during the cleansing and tallying phases. Interestingly, Civitas actually leaks more information than JCJ. First, it provides the same leakage as the AFT protocol: the number of revotes for each ballot can be directly deduced from the board. Furthermore, in order to reduce the quadratic number of PETs, Civitas proposes to group voters by blocks: each credential is publicly assigned to one block, and the voter indicates their block in clear when casting their ballot. Compared to JCJ, the adversary still learns how many revotes each ballot has and how many invalid ballots there is, but also has access to this information block by block. Similarly to the AFT scheme, it is difficult to adapt Civitas' approach in MPC without leaking any side-channel information. Indeed, if we group the voters by blocks, then the number of ballots that each block has is a side-channel information (if the coerced voter evades coercion, the corresponding block has one extra ballot). Hence, using this approach would either require to cast a large number of dummies for each block (which would undermine the efficiency), or to use an advanced MPC strategy to hide how many ballots each block has. Once again, we cannot think of a way to achieve this in linear time.

While JCJ does not satisfy our definition of coercion-resistance, it does provide a certain level of security. 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] D. F. Aranha, M. Battagliola, and L. Roy, "Faster coercion-resistant e-voting by encrypted sorting," Cryptology ePrint Archive, Paper 2023/837, 2023, https://eprint.iacr.org/2023/837.

[2] R. Araújo, A. Barki, S. Brunet, and J. Traoré, "Remote Electronic Voting Can Be Efficient, Verifiable and Coercion-Resistant," in *FC'16*. Springer, 2016.

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

[4] R. Araújo, N. B. Rajeb, R. Robbana, J. Traoré, and S. Yousfi, "Towards Practical and Secure Coercion-Resistant Electronic Elections," in *Cryptology and Network Security - 9th International Conference, CANS'10*. Springer, 2010.

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

[6] 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.

[7] 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.

[8] 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.

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

[13] ——, "Is the JCJ voting system really coercion-resistant? (long version)," Cryptology ePrint Archive, Paper 2022/430, 2024.

[14] 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.

[15] 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.

[16] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems," in *EUROCRYPT'99*. Springer, 1999.

[17] G. S. Grewal, M. D. Ryan, S. Bursuc, and P. Y. A. Ryan, "Caveat coercitor: Coercion-evidence in electronic voting," in *IEEE Symposium on Security and Privacy - SP'13*. IEEE, 2013.

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

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

[20] 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.
[21] M. Jakobsson and A. Juels, "Mix and Match: Secure Function Evaluation via Ciphertexts," in *ASIACRYPT'00*. Springer, 2000.
[22] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated Verifier Proofs and Their Applications," in *EUROCRYPT'96*. Springer, 1996.
[23] 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.
[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] P. Locher, R. Haenni, and R. E. Koenig, "Coercion-Resistant Internet Voting with Everlasting Privacy," in *FC'16 International Workshops*. Springer, 2016.
[27] W. Lueks, I. Querejeta-Azurmendi, and C. Troncoso, "VoteAgain: A scalable coercion-resistant voting system," in *USENIX'20*. USENIX, 2020.
[28] 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.
[29] E. McMurtry, O. Pereira, and V. Teague, "When Is a Test Not a Proof?" in *ESORICS'20*. Springer, 2020.
[30] B. Schoenmakers and P. Tuyls, "Practical Two-Party Computation Based on the Conditional Gate," in *ASIACRYPT'04)*. Springer, 2004.
[31] 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.
[32] ——, "Achieving Meaningful Efficiency in Coercion-Resistant, Verifiable Internet Voting," in *5th International Conference on Electronic Voting, EVOTE'12*. GI, 2012.
[33] D. Wikström, "Open Verificatum," https://www.verificatum.org/, version 3.1.0, retrieved Aug. 2023.
[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.
[36] ——, "A commitment-consistent proof of a shuffle," in *Information Security and Privacy, 14th Australasian Conference, ACISP'09*. Springer, 2009.