On remote electronic voting with both coercion resistance and cast-as-intended verifiability

Tamara Finogina and Javier Herranz

Abstract

In this work, we study two essential but apparently contradictory properties of electronic voting systems: coercion resistance (CR) and cast-as-intended verifiability (CAI). Informally, the CR property ensures that a voter cannot prove to anybody else the vote content, which prevents vote selling and voting under duress. The CAI property ensures that a malicious voting device cannot cheat the voter and send to the ballot box an encryption of a voting option different from the one chosen by the voter.

In this work, we formalize security definitions capturing both coercion resistance and cast-as-intended verification in settings without secure delivery channels between the election authority and voters. After that, we consider some previously proposed solutions aimed at providing these two properties. For some of them (that we call unsatisfactory solutions) we show why they fail to achieve some of the two properties. We then concentrate on one of the two generic solutions that we call satisfactory: we prove that it satisfies the two proposed definitions and we detail how it can be instantiated in both classical cryptographic (e.g., ElGamal ciphertexts) and quantum-resistant (e.g., using lattice-based cryptosystems) settings.

1. Introduction

Almost all e-voting protocols (except for code-voting and its flavors) expect to receive from a voter an encrypted vote. Usually, due to the complexity of this task, the encryption is not done by the voter but delegated to an external and untrusted voting device. However, it raises a concern: does the ballot cast by the voting device correctly reflect the voter’s intent? Schemes that allow voters to verify that the ballot contains the correct intent enjoy the property of cast-as-intended (CAI) verifiability (also known as ballot assurance).\(^1\)

There are different techniques to achieve some sort of CAI verifiability. Some voting schemes allow to generate and to verify (multiple) test ballot(s) before casting the real one [1,2]. Others enable verification of the cast ballot via zero-knowledge proofs [3], return codes [4], tracking numbers [5], QR-codes [6], etc. However, all of them require voters to keep some information secret (e.g., QR-code, proofs, a voting card with all return codes, etc.), which makes them incompatible with another crucial property - coercion-resistance.

Another desirable property of e-voting systems is coercion-resistance, which is one of the most studied but least standardized properties of such systems. While, intuitively, it is clear that the goal is to protect freedom of will, all formalized definitions [7–13] struggle to capture the broadness of the notion [14–16].

We can firmly agree that the coercer cannot control a voter all the time [9]; else, the voter has no choice but to comply [13]. Similarly, we can safely assume that the coercer would not use extreme threats such as putting a gun to the voter’s head. Otherwise, the problem perhaps is not with the voting channel per se and should be addressed by the police. Thus, coercion-resistance makes sense only when the coercer has limited observational power, and the voter can disobey without facing high risks.

We cannot evaluate individual risks, so we focus on defining the coercer’s limitations. However, this task is not as trivial as it may seem. Consider a typical scenario: the voter goes to a polling place and votes. Can the coercer stop the voter from entering the polling place? Can
the coercer force the voter to bring a photo copy of the filled ballot? For electronic voting, those question roughly translates to “are forced abstention attacks allowed?” and “is extracting randomness from the voting device permitted?”. To make the matter worse, some voters might try to intentionally sell their votes, thus turning from honest into malicious players. On top of that, electronic voting allows registration, vote casting, and vote verification to happen in different time slots. It brings an entirely new question: “does exactly the coercer can observe the voter?”.

Depending on the answers, one can come up with different definition’s flavors. Unfortunately, all existing approaches are either too weak [5,6,17] or incompatible [10,18] with cast-as-intended verification.

The situation becomes even more challenging when we remove trusted channels between voters and authorities. In other words, voters no longer can receive any vote-verification material such as pre-generated keys or pre-printed cards. As far as we know, except for two solutions in [19], no voting scheme provides CAI verification and (some sort of) coercion resistance in such settings.

1.1. Related work: a mini survey on CR and CAI properties

In this section, we provide a quite complete summary of existing work dealing with the properties of Coercion-Resistance (and the related notion of Receipt Freeness) and Cast-as-Intended verification.

1.1.1. Coercion resistance

The first verifiable election protocol that focuses on preventing participants from selling their vote to a passive coercer was designed by Benaloh and Tuinstra in 1994 [13]. In the same work, they introduced the concepts of receipt-freeness and uncoercibility. Later, in 1997, Okamoto proposed an alternative definition of receipt-freeness [20], which allowed the coercer to interact with the voter during the voting phase. Despite the name, it is essentially the first definition of what is now known as coercion-resistance.

The simplest form of coercion-resistance technique is multiple voting [6]. However, it is not always legal as it gives an unfair advantage to e-voters [14]. Moreover, it implicitly assumes that the coercer cannot see the ballot box content nor observe voting channels. Otherwise, it is trivial to see whether the voter obeyed or not, which makes multiple voting the weakest form of coercion-resistance.

The first formal definition of coercion-resistance, known as the JCJ definition, appeared more than a decade later in the work of Juels et al. [10].

The JCJ definition is the most demanding definition of coercion-resistance. It captures the idea of anonymous credentials and accounts for intentional credentials selling and forced-abstention attacks. Unfortunately, any scheme that realizes it in practice [21-25] requires anonymous voting channels and puts trust in the voting device, as CAI verification is not possible. Moreover, it requires secure pre-delivery of secret keys and a system for managing real and fake credentials, which is challenging given that voters have trouble remembering passwords they use a few times per year [14]. Recently, authors of [26] found that the JCJ definition leaks too much data in case of revoting and proposed an enhanced CR definition.

Another definition of coercion-resistance [8] challenges the adversary to distinguish between two ballots: one generated for the coercer’s preference and another for the true intent (our definition for the CR property, in Section 3.3, will follow this idea). It separates the vote-casting part from the tallying, which, despite being criticized [16], allows excluding plaintext-based leakage: e.g., Italian attacks, specific write-ins, etc. Note that plaintext leakage is almost impossible to avoid (unless homomorphic tally is used) and is present in any voting channel that permits free-form votes. Unfortunately, the definition in [8] prohibits any post-election communication between the coercer and the voters.

Delaune, Kremer, and Ryan [9] propose a formalization of coercion-resistance in the Dolev-Yao model and also formally prove that it implies receipt-freeness which, in turn, implies privacy. However, the coercer is severely restricted in its choice of strategies.

The family of UC-definition of coercion-resistance started in 1996 with a weaker notion of (post-factum) incoercibility [7]. The definition was allowing adversary to coercer voters only after voting phase was completed i.e. voter was not expected to receive any instructions or input before voting. The generalization of this property – receipt freeness given in [12] – permits coercers to corrupt voters at any time and enforce them to follow any strategy. Achieving receipt-freeness under the definition in [12], however, requires commitments with extremely strong hiding property that can only be implemented by physical means. Moreover, reactive protocols (where the input in one phase depends on the output of the other) are not modeled. Subsequent works [27,28] addressed the issue by defining Composable Incoercibility framework, which allows to model reactive protocols and gives universal composition. We stress that later work [16] has found issues in some parts of these definitions.

The only statistical/quantitative approach to measuring coercion-resistance, σ-coercion resistance, was proposed in [11]. The definition assumes the coercer cannot perform at least one step of the voting protocol without the voter, requires prior knowledge of the choices distribution and the exact election configuration, and covers a multi-coercion case. The definition permits the coercer to succeed with non-negligible probability.

1.1.2. Receipt freeness

Nowadays, coercion is typically divided into passive coercion (also known as receipt-freeness) and active coercion. For receipt-freeness, the adversary is not allowed to impose instructions on the actions of the voter during the voting phase [18,21].

The very first definition of receipt-freeness [13] aimed to adopt the idea of a private voting booth for electronic voting. While inside, the voter cannot communicate with anyone or be under observation. The definition, however, assumes the existence of a trusted randomness source (beacon) used for challenging the system.

Other definitions of receipt-freeness are similar in spirit. For example, more recent definitions [17] (which was found to be incomplete [29]) and [5] allow the coercer to demand proofs (but not voting cards or encryption randomness) from honest voters only after vote-casting and verification. Note that both definitions assume the existence of an unappable channel between voters and authority.

Contrary to traditional receipt-freeness, the strong receipt-freeness [18] property (was found to be incomplete [29]) demands voters to use coercer-selected ballot material (e.g., encryption randomness). The scheme that implements the definition works under the assumption that the adversary cannot observe the interaction between the voter and the re-randomization server, which is trusted to modify the ballot before it can be published. CAI verification is incompatible with such demands.

A quantitative approach to receipt-freeness was proposed in [30]. The σ-receipt freeness is based on [18], but similar in spirit to [11]. It does not consider the leakage from tally results nor accounts for coercers that demand voters to cast invalid votes. Moreover, the adversary is not allowed to observe the channel between the voter, the posting trustee and the voting system. However, it is the only definition that allows to measure the level of receipt-freeness.

1.1.3. Cast-as-Intended verification

The first proposal for cast-as-intended verification was published by Chaum in [31], followed by Neff’s approach [32], published the same year. Both works expect voter to cast an electronic vote in a polling place and require printing commitments for ensuring that the voter’s choice is encrypted correctly. Chaum’s proposal was based on visual cryptography and special printer, while Neff’s utilized interactive zero-knowledge proofs that can be simulated. For coercion-resistance, Neff’s
scheme assumes that the voter selects challenges from a pre-generated set, and the voting device can print a short secret without revealing it. Nevertheless, the scheme works only for honest voters; a dishonest one can attempt to link the challenge’s choice with the shown commitment (for details see $U$, solution in Section 4).

Currently, there exist numerous methods to ensure that the cast vote contains the voter’s intent and not something else. However, we can divide all those methods into several groups based on the origin of data used for the verification: pre-delivered, generated, or probabilistic check based on previous attempts.

The first group consists of schemes that require pre-printed and pre-delivered paper material. It includes all return-code (e.g. [4, 33]), all vote-code (e.g. [17]), and pre-encrypted ballots schemes (e.g. [34]). The return code-based verification means that the voter casts a vote and receives a code, which should match the expected return code for that option from the voting card. The vote code-based systems expect voters to cast a unique vote code (in clear) instead of an encrypted one; the codes are printed next to each possible voting choice. Finally, schemes based on pre-encrypted ballots expect voters to scan one of the possible ballots that contain an encryption of the desired option.

The second group relies on probabilistic checks and does not inspect the cast ballot per se. It consist of all schemes that rely on the cast-or-challenge approach (e.g. [1]), partial checks (e.g. [35]), or schemes that generate multiple ballots for the same option and allow voters to inspect all but one of them (e.g. [2]).

The schemes in the third group provide voter feedback sufficient for CAI verification. It includes outputting encryption randomness directly or as a QR-code (e.g. [6]), giving tracking numbers (e.g. [5]), generating zero-knowledge proofs of the encryption randomness, OR-proofs (e.g. [3]), etc.

Obviously, the first group mandates existence of trusted and reliable delivery channels. Moreover, coercion can be prevented if and only if only voters keep paper materials secret or destroy them. The idea that voters can fake paper material [17] is not feasible in practice. If the real printed card is indistinguishable from the fake one, voters will have no means to ensure they use the right one. The only way to guarantee the authenticity of the printed card in such a case is to physically go to authorities to obtain it, which defeats the whole purpose of remote voting.

The second group does not verify the cast ballot, thus there is always a non-negligible possibility that it was altered.

The third group is the most promising one in terms of providing coercion-resistance. However, it works only when voters can simulate the feedback provided by a voting scheme. Moreover, the majority of voting schemes rely on secure voting channels between voters and authorities for delivering keys or credentials. The only exception we consider only schemes that offer client-side encryption: a voter, who wishes to vote for an option $n$, must communicate $n$ to a voting device that encrypts $n$ using a public key encryption scheme, obtaining a ciphertext $C$ which, after the voter accepts it, is sent to the ballot box.

The actions of the voter might be dictated by the coercer or possible monetary incentive (e.g., the voter might wish to sell the vote); hence we cannot consider the voter to be trustworthy. During the vote casting protocol, the voter and the voting device interact without the presence of the coercer, who, however, can later oblige the voter to reveal the information (supposedly) exchanged between the voter and the voting device. We also assume the voter is equipped with a local and trusted device to run basic cryptographic/mathematical operations.

We recall and reformulate, in Section 5.1 and in Appendix, the two generic constructions of electronic voting given in [19] for this setting. We formally prove (in Section 5.2) that one of the constructions (the one which leads to more efficient instantiations) achieves both CR and CAI verification. To do so, we first provide new formal security definitions of these two properties, in Section 3, where we also compare our definitions with (the numerous, assorted, sometimes incomparable) existing ones.

Other than that, the paper contains: (i) necessary preliminaries in Section 2; (ii) a discussion of some unsatisfactory but inspiring ideas (some of them can be found in specific papers, some of them are folklore) to achieve both CR and CAI verification in Section 4; and (iii) a description, in Section 6, of two possible ways of instantiating the generic construction described and analyzed in Section 5: one in the classical setting of ElGamal ciphertexts and one based on lattice-based techniques that are supposed to be secure even in the presence of quantum computers.

2. Preliminaries

As usual, we say that a function $e$ is negligible in $n \in \mathbb{N}$, and write $e \in \text{negl}(n)$, if for every polynomial $p$ there exists $N$ such that for every $n \geq N$ it holds $e(n) \leq 1/p(n)$. By $x \mapsto X$ we denote taking an element $x$ from a finite set $X$ uniformly at random.

2.1. Relations, languages and zero-knowledge systems

Let $R$ be a polynomial-time decidable binary relation defined on a pair of finite spaces. That is, $R$ is a subset of $X \times W$ subject to the following conditions [37]:

- There exists a polynomial $p$ such that if $(x,w) \in R$ then $|w| \leq p(|x|)$
- There exists an efficient algorithm $\mathcal{G}$ that on input $\lambda \in \mathbb{N}$ outputs $(x,w) \in R$ with $|x| \geq \lambda$.
- There exists an efficient algorithm $\mathcal{R}$ that outputs a bit such that $R(x,w) = 1 \iff (x,w) \in R$.

Such relation $R$ is an NP-relation, and gives rise to the set of “yes”-instances defined as $\mathcal{L}_R = \{ x \in X \mid \exists w \in W \text{ s.t. } R(x,w) = 1 \} \subseteq X$, known as the language of $R$. Also, $R$ is believed to be hard if there are no known efficient algorithms that can recover $w$ from $x$ with non-negligible probability on $\lambda$.

Interactive zero-knowledge systems. Let $R$ be a binary relation parameterized with a security parameter $\lambda$. An interactive protocol $(P, V)$ between a prover $P$ and a verifier $V$, both being PPT algorithms, is said to be a zero-knowledge proof system for $R$ if the following holds [37]:

Completeness: On common input $x$, and Prover’s private input $w$, if $(x,w) \in R$ it holds $\Pr[P(x,w), V(x)] = 1$.

---

This must happen in both physical (polling stations) and Internet-based protocols. As we have mentioned in the Introduction, this assumption means we do not consider some kinds of (strong) coercion, including forced abstention or family coercion. In any case, this assumption is necessary in our setting without secure channels between voters and authorities, see first paragraph of Section 3.3.
**Soundness:** For every $y \not\in L_R$, it holds $\Pr[\langle P(y), V(y) \rangle = 1] \in \text{neg}(\lambda)$

**Perfect zero-knowledge:** For every verifier $V'$ there exists a PPT simulator $\mathcal{V}_R$ such that for every $(x, w) \in R$ the output $(P(x, w), V'(x))$ is identically distributed to the output $\mathcal{V}_R(x)$. This property can be relaxed requiring that the outputs are only statistically or computationally close.

**Sigma protocols.** A very popular and powerful special kind of interactive protocols are $\Sigma$-protocols, which produce transcripts $(a, e, z)$ in three rounds of communication: the first message $a$ is sent by $P$, then $e$ is chosen by $V$ uniformly and at random in a challenge space of size $t$ and finally $z$ is the third message, sent by $P$. The required properties for a $\Sigma$-protocol, other than completeness, are [38]:

- **Special soundness:** There exists a PPT algorithm $\mathcal{V}_R$ such that for any $(x, w) \in R$, any two accepting transcripts $(a, e, z, (a', e', z'))$ of $\Sigma$-protocol with the tuple $(x, w, V(x))$.

- **Special honest-verifier zero-knowledge:** There exists a PPT algorithm $\mathcal{V}_R$ such that for any $(x, w) \in R$, on input $x$ and a random $e$, outputs an accepting transcript $(a, e, z)$ of $\Sigma$-protocol with the same probability distribution as $(P(x, w), V(x))$.

2.2. Simulating execution of ZK proof systems for instances out of the language

The language $L_R = \{ x \in X \mid \exists w \in W \ s.t. (x, w, 1) \in X \}$ associated to a binary relation $R$ is hard-to-distinguish inside the space $X$ if, given a random element in $X$, it is hard to distinguish in polynomial time if the element belongs to $L_R$ or not. That is, for any polynomial time distinguisher $D_R$, it holds

$$\Pr[1 \leftarrow D_R(x) \mid x \not\in L_R] = \Pr[1 \leftarrow D_R'(x') \mid x' \not\in X - L_R] \in \text{neg}(\lambda)$$

where $\lambda$ is the security parameter of the system.

Let us assume that there exists a proof system for $R$ enjoying zero-knowledge: for every verifier $V'$ there exists a PPT simulator $\mathcal{V}_R$ such that for every $(x, w) \in R$ the distribution $(P(x, w), V'(x))$ is (perfectly/statistically/computationally) indistinguishable to the distribution $\mathcal{V}_R(x)$.

We claim that such a simulator $\mathcal{V}_R$ must also work when the input is an element $x' \in X - L_R$ out of the language $L_R$, if $L_R$ is a hard-to-distinguish language, and that the joint distributions $(x, \mathcal{V}_R(x))_{x \in L_R}$ and $(x', \mathcal{V}_R(x'))_{x' \in X - L_R}$ are indistinguishable.

**Lemma 1.** The two joint distributions $(x, \mathcal{V}_R(x))_{x \in L_R}$ and $(x', \mathcal{V}_R(x'))_{x' \in X - L_R}$ are indistinguishable.

**Proof.** Let us assume the existence of a distinguisher algorithm $D$ that can distinguish these two distributions. We use $D$ to build a distinguisher $D_R$ against the hard-to-distinguish property of $L_R$, as follows.

$D_R$ receives as input an element $x \in X$, and its goal is to distinguish if $x \in L_R$ or if $x \not\in X - L_R$. What $D_R$ does is to run the simulator $\mathcal{V}_R(x)$. If this simulator aborts, $D_R$ answers $x \in X - L_R$. Otherwise, $D_R$ runs the distinguisher $D$ with the tuple $(x, \mathcal{V}_R(x))$ as input, and outputs the same answer as $D$ outputs.

2.3. Homomorphic public key encryption and proofs of correct encryption

Let $E = (E, K_G, E, Enc, E, Dec)$ be a public key encryption scheme with semantic security, for a plaintext space $M_E$ and a ciphertext space $C_E$. Roughly speaking, this property says that any adversary that sees a public key $\mathcal{K}$ either chooses a pair of plaintexts $m_0, m_1$ and receives an encryption of one of the two (chosen at random) cannot distinguish which is the encrypted plaintext with better probability than $1/2$. In particular, a public key encryption scheme with such a property must be probabilistic: there exists a random space $R_S_E$ and the encryption protocol $Enc$ chooses some random value(s) in $R_S_E$ in the process of computing the ciphertext. Therefore, we can write $C = Enc(m, r)$ to denote an execution of the encryption protocol for plaintext $m \in M_E$, using randomness $r \in R_S_E$ and producing a ciphertext $C \in C_E$. For brevity, we omit the fact that the public key pk is also an input of the protocol $Enc$.

For such a probabilistic encryption scheme $E$, we can consider the binary relation $R_E$ defined on sets $X = M_E \times C_E$ and $Y = R_S_E$ as $(x, w) \in R_E \iff C = Enc(m, r)$, where $x = (m, c), w = r$. Note that, if $E$ is semantically secure, then the associated language $L_E$ is hard-to-distinguish in a quite specific way: elements $x = (m, c) \in L_E$ are indistinguishable from elements $x' = (m', c) \in X - L_E$, even if the second component (the ciphertext) is the same.

Therefore, for any zero-knowledge proof system for $R_E$, the associated simulator will also work (and in an indistinguishable way) when the input $x' = (m', c)$ is an element out of the language, and thus $Enc(C, sk) \neq m'$. This property will be crucial in our construction(s) of coercion-resistance voting schemes.

There are many semantically secure encryption schemes $E$ such that the relation $R_E$ admits an efficient zero-knowledge proof system, including $\Sigma$-protocols. This is particularly true when the encryption scheme has some homomorphic property, as we exemplify in the following subsection, for the case of ElGamal encryption.

2.3.1. A particular example: The Diffie–Hellman language and ElGamal PKE

Given a cyclic group $G = (g)$ with prime order $p$ and generator $g$, let us consider the space $X = G^3$. The witness space is $W = Z_p$ and the binary relation is $R_{DH} = \{(g_1, g_2, g_3; w) \in X \times W \mid g^w = g_1 \cdot g_2 \cdot g_3\}$. That is, an element $w = (g_1, g_2, g_3)$ belongs to the Diffie–Hellman language $L_{DH}$ if the discrete logarithm of $g_3$ with respect to basis $g$ is the same as the discrete logarithm of $g_3$ with respect to basis $g_2$. The Diffie–Hellman Assumption in $G = (g)$ states that this is a hard-to-distinguish language inside $X = G^3$.

In ElGamal encryption scheme, the secret key is an element $sk = (g, \cdot, m')$, the public key is $pk = g^m$ and an encryption of $m \in G$ is $C = (g', m', m' g^r)$, where the randomness is $r \in Z_p$. Thus, the language associated to ElGamal public key encryption scheme contains elements $(x, m) \in M \times (m, c)$, where $x = (c_1, c_2)$, satisfying the condition: “the triple $(c_1, pk, \frac{m}{c_2})$ belongs to the Diffie–Hellman language $X$.”

A $\Sigma$-protocol for the Diffie–Hellman relation (and thus, in particular, a $\Sigma$-protocol for proving that $C$ is an ElGamal encryption of $m$) works as follows: the public input for prover and verifier contains $(g, g_1, g_2, g_3)$, whereas the private input of the prover is the witness $w$. In the first step, $P$ chooses $v \in Z_p$ at random and sends the tuple $(R_1, R_2)$ to $V$, where $R_1 = g^v$ and $R_2 = g_1^v$. In the second step, $V$ chooses and sends a random challenge $e \in Z_p$. In the third step, $P$ computes and sends the last element $z = v + ew \mod p$. Finally, $V$ accepts the proof if and only if $g^z = R_1 \cdot g_2^e$ and $g_3^z = R_2 \cdot g_2^e$.

This $\Sigma$-protocol enjoys the honest-verifier zero-knowledge property: the simulator $\mathcal{S}$, when given as input an element $x = (g_1, g_2, g_3)$ in the language and a random challenge $e$, can simulate the transcript of an execution of the $\Sigma$-protocol, by taking $z \in Z_p$ at random and computing $R_1 = g_1^z \cdot g_2^e$ and $R_2 = g_3^z \cdot g_2^e$.

As we advanced in Section 2.2 this simulator also works if the input is an element $x' = (g_1', g_2', g_3')$ out of the Diffie–Hellman language; distinguishing the joint distribution $(g_1, g_2, g_3, e, x, R_1, R_2)$ resulting from


3. The voting protocol: Syntax and properties

3.1. Parties and syntax of the protocol

We are going to define cast-as-intended and coercion-resistance in a local setting, focused on a specific voter $\mathcal{V}$ who casts a vote through a voting device $\mathcal{V}D$, possibly under the coercion of an adversary $\mathcal{A}$. We will thus forget other protocols and properties of the global voting system, like the authentication of voters, the mixing or homomorphic tally step, the final decryption and publication of the result, etc.

Therefore, we will not consider a global adversarial setting where the coercer can use the final result of the election to deduce if the coerced voter followed its instructions or not. Such a global approach falls outside the scope of this work. Our local approach, however, makes perfect sense in large elections where we want to ensure that:

(i) honest voters will not be cheated by a dishonest voting device, and
(ii) (dishonest) voters cannot prove how they voted to anybody else.

The intuition is that the voter $\mathcal{V}$ of the security parameter $\lambda$, an element $C$, and an element $\text{coerc}$, which is either empty (in case of no coercion) or specifies some instructions given to $\mathcal{V}$ by the coercer $\mathcal{A}$.

We denote as $\text{rand}_\mathcal{V}$, the set of all values explicitly selected by $\mathcal{V}$ during the execution of the voting protocol (e.g., challenges), and as $\text{Trc}$ the list of messages exchanged between $\mathcal{V}$ and $\mathcal{V}D$ in such an execution. In particular, $\text{Trc}$ includes a voting option $m$ and a ciphertext $C$, which is supposed to be (if $\mathcal{V}D$ is honest) an encryption of the voting option $m$ under the public key $pk$ of the election authority and using randomness $r$, for some public-key encryption scheme $\text{Enc}$ that is $C = \text{Enc}(m, r)$, where we intentionally omit $pk$ as a well-known input of the encryption protocol $\text{Enc}$. Other than $\text{Trc}$, the voting protocol has two private output bits:

- a bit $b_\mathcal{V}$ describing if the voter $\mathcal{V}$ accepts the interaction as valid ($b_\mathcal{V} = 1$) or not ($b_\mathcal{V} = 0$),
- a bit $b_{\mathcal{V}D}$ describing if the voting device $\mathcal{V}D$ accepts the interaction as valid ($b_{\mathcal{V}D} = 1$) or not ($b_{\mathcal{V}D} = 0$).

In practice these two bits can be thought as outputs of some functions locally run by $\mathcal{V}$ and $\mathcal{V}D$, which take as inputs their internal secret values and the transcript of the interactive protocol. If both $b_\mathcal{V} = b_{\mathcal{V}D} = 1$, then $C$ is a valid vote. If $b_\mathcal{V} = 0$, the voter would complain against the voting device. Similarly, if $b_{\mathcal{V}D} = 0$ then the ballot box (as well as the voting device) would not accept the vote as valid. We denote an execution of the voting protocol as $\text{Vote}(\mathcal{V}, \mathcal{V}D)(\text{pms}, m, \text{coerc}) \rightarrow (b_\mathcal{V}, b_{\mathcal{V}D}, \text{Trc}, \text{rand}_\mathcal{V})$.

3.2. Formal definition of Cast-As-Intended (CAI) verifiability

Informally, we want to ensure that an honest not coerced voter who runs the voting protocol with a voting option $m$ and accepts the interaction as valid can be confident that $C$ actually contains encryption of $m$. The property is thus formalized by considering a dishonest voting device that tries to cheat the voter and sends to the ballot box a ciphertext $C$ that decrpyts to $m' \neq m$. We want that the probability of such a cheating behavior to happen without being detected by the voter is negligible.

Let Cheat be the event where $\text{pms} \rightarrow \text{Setup}(\lambda), (b_\mathcal{V}, b_{\mathcal{V}D}, \text{Trc}, \text{rand}_\mathcal{V}) \rightarrow \text{Vote}(\mathcal{V}, \mathcal{V}D)(\text{pms}, m, \emptyset), b_\mathcal{V} = 1$, and $C \in \text{Trc}$ satisfies $m \neq \text{Dec}(C, sk)$, where $sk$ is the decryption secret key of the global election, matching the public key $pk$, and $\text{Dec}$ is the decryption algorithm.

Definition 1. The protocol Vote enjoys Cast-As-Intended (CAI) verifiability if the probability of event Cheat is a negligible function of the security parameter $\lambda$, for any polynomial-time voting device $\mathcal{V}D$.

The probability is taken over the random choice of $m \in \mathcal{M}$ and of random values $\text{rand}_\mathcal{V}$.

3.3. Formal definition of Coercion-Resistance (CR)

In our setting, a coercer $\mathcal{A}$ (e.g., a vote buyer) may interact with the voter $\mathcal{V}$ before the execution of the voting protocol, and force him to vote for some option $m^*$ and to follow the instructions in coerc while executing Vote. During the execution of Vote, the adversary does not have access to the (secret) channel between $\mathcal{V}$ and $\mathcal{V}D$ (otherwise, since the voting option is sent by $\mathcal{V}$ to $\mathcal{V}D$ through this channel, it would be impossible to achieve any meaningful coercion-resistance property).

However, the adversary has access to the public channel through which the ciphertext $C$ is sent by $\mathcal{V}D$ to the ballot box. After the execution of the protocol, $\mathcal{A}$ expects to receive from $\mathcal{V}$ a copy of Trc along with the randomness and secret information generated by $\mathcal{V}$ during Vote.

To prevent coercion in such settings, the intuition is that the voter $\mathcal{V}$ must always have the possibility to deceive the coercer: it should be able to run a real execution of the protocol Vote with its voting option $m$, and later simulate all values (including the challenges) to make $\mathcal{A}$ believe that Vote was run with $m^*$ as input. The coercer $\mathcal{A}$ can distinguish if the voter indeed followed the instructions and voted for $m^*$ or provided simulated values with probability 1/2.

If we can prove such a property, then coercion will be avoided because $\mathcal{A}$ will never be convinced if the voter followed its instructions or not when running Vote. The formalization of this property is given in the following definition.

Definition 2. The protocol Vote enjoys coercion-resistance (CR) if for any polynomial-time coercer $\mathcal{A}$, any coercion $(m^*, \text{coerc})$ and any voting option $m$, there exists a polynomial-time simulator algorithm $\text{Sim} = (\text{Sim}_1, \text{Sim}_2)$ such that, in the experiment described below, the experiment does not abort and $\Pr[b' = b] - \frac{1}{2}$ is a negligible function of the security parameter $\lambda$:

1. $\text{Setup}(\lambda) \rightarrow \text{pms}$.
2. $\mathcal{A}(\text{pms}) \rightarrow (m^*, \text{coerc})$.
3. $\text{Vote}(\mathcal{V}, \mathcal{V}D)(\text{pms}, m^*, \text{coerc}) \rightarrow (b_\mathcal{V}, b_{\mathcal{V}D}, \text{Trc}, \text{rand}_\mathcal{V})$.
   If $b_{\mathcal{V}D} = 0$, abort the experiment.
   Else define $\text{view}_0 = (b_\mathcal{V}, b_{\mathcal{V}D}, \text{Trc}, \text{rand}_\mathcal{V})$.
4. $\text{Sim}_2(b_\mathcal{V}, b_{\mathcal{V}D}, \text{Trc}, \text{rand}_\mathcal{V}, m, m^*, \text{coerc}) \rightarrow (b_{\mathcal{V}D}', \text{Trc}, \text{rand}_\mathcal{V})$, with $C_{\mathcal{V}D} \in \text{Trc}$ satisfying $C_{\mathcal{V}D} = C$.
   Define $\text{view}_1 = (b_{\mathcal{V}D}', \text{Trc}, \text{rand}_\mathcal{V})$.
5. $b \leftarrow \{0, 1\}$ is chosen uniformly at random.
6. $\mathcal{A}(\text{pms}, m^*, \text{coerc}, \text{view}_0) \rightarrow b'$.
Since the experiment aborts when the coercion chosen by $A$ makes the (honest) voting device reject the execution of the protocol Vote, our definition does not cover some very strong types of coercion: (1) the coercer forces a voter not to participate in the election, (2) the coercer forces a voter to misbehave during the interaction with $VD$, which results in $VD$ aborting the protocol and not sending any ciphertext to the ballot box.

Under our assumption that the adversary can see the ciphertext $C$ that is sent by $VD$ to the ballot box, these stronger types of coercion could be addressed only with solutions that allow $VD$ to send a ciphertext (encryption of a dummy/special plaintext) to the ballot box even in the case when a voter abstains or when a voter misbehaves during the voting protocol. This is explicitly prohibited by some countries that run electronic elections today. Furthermore, this would open the door to possible attacks by a dishonest $VD$ which, in case of abstention, does not send an encryption of the dummy/special plaintext, but encryption of a specific valid voting option.

We also remark that the copy of $Trc$ will be sent to $A$ in an off-line way (and this will be crucial to get solutions enjoying CR). Coercers that get a copy of $Trc$ in an on-line way are known as active coercers in the incoercible multi-party computation setting [12,27,28]; enjoying CR in front of active coercers requires the use of secure channels between voters and authorities and/or the use of physical devices (such as hardware tokens).

Regarding the kind of instructions that $A$ can impose to voter $V$ inside coerc, they consist of constant values $\{K_i\}$ and functions $f_i$ that specify how $V$ must compute all the values that it is supposed to compute and send during the (coerced) execution of Vote.

### 3.4. Comparison with previous definitions of CAI and CR

Regarding the CAI definition, since CAI verifiability is a local property, it is clear that the definition must be local, focused on the interaction between the voting device and the voter in the vote-casting phase of the election. Our definition is game-based and similar in spirit to that in [36]. The main difference is that we are interested in yes/no CAI verification: the goal is that a malicious $VD$ has a negligible probability of cheating a voter. In contrast, a more general and quantified definition was provided in [36], to measure or bound the (non-necessarily negligible) probability that $VD$ cheats a voter without being detected.

The analysis of our CR definition and its comparison to other existing ones is more complex, due to the high variety of settings considered for CR in the literature. First, the fact that we consider e-voting systems without secure channels between authorities and voters is very relevant and has consequences on the CR definition. In particular, in such a setting it is clear that a voter $V$ must send its voting choice $m$ to the voting device $VD$ during the vote-casting phase; therefore, we must assume that $VD$ is trusted for privacy and for coercion-resistance. Otherwise, it would be impossible for an honest voter to deceive a coercer colluding with $VD$. This is in contrast to other definitions of CR, in particular, the JCI one [10] and variants thereof [26].

In our CR definition, the analysis is local: we assume that one single voter is under coercion (by a real coercer or by itself, in the case of vote-selling). In other words, we do not worry about the behavior of other voters or other authorities of the e-voting system. This simplification allows us to concentrate only on the vote-casting phase of the election, the only one where the voter plays an active role. This is common to the model in [8], but is different from many other CR definitions and models, which deal with more global adversaries [10,26,28].

We believe a local approach makes perfect sense, in particular for typical elections where the rest of the phases (after vote casting) are done in a verifiable way: verifiable shuffle of ciphertexts or verifiable homomorphic tally, verifiable decryption of the final results, etc., possibly executed by a threshold of servers.

A direct consequence of our local setting for CR is that we can consider a (simpler to use) game-based definition: the goal of the (real) adversary is to distinguish between two possible (real) executions of the protocol Vote. This is in contrast to CR definitions in the (conceptually more complicated, in our opinion) UC-based setting, where two different worlds (the real and an ideal world) are considered, and the goal is to show that the behavior of a real adversary (coercer, in this case) in a real world can be simulated in the ideal world, without anybody (the environment) being able to distinguish between both. Works proposing CR models and proofs in the UC-based setting include [12,26,28].

Finally, as it happened for CAI, we are interested in a yes/no notion of CR: the probability that a coercer $A$ distinguishes if $V$ is obeying coercion or not should be negligibly greater than $1/2$. This is different from some existing quantitative definitions for CR or receipt-freeness, where non-negligible success probabilities of the coercer/vote seller are allowed [11,30].

### 4. Common approaches for providing CAI verification

In this section, we discuss several approaches to achieve both CAI verification and CR properties that have been considered in previous works and show why they (may) fail, when secure delivery channels between election administrators and voters are not available.

Generally, the voting systems prefer to prove CAI rather than CR protection. The only contradicting example we are aware of is a scheme called Civitas [21], which provides CR protection at the cost of CAI as the voting device $VD$ is just assumed to be honest.

The most basic form of CAI verifiability requires no interaction with the voter, i.e., CAI that relies on transferable proofs. It includes giving voters QR-codes that store encryption randomness, a non-interactive zero-knowledge proof of knowledge of encryption randomness, etc. Such an approach fails to provide CR protection as any proof that convinces the voter would be equally convincing for the coercer.

Similarly, if we assume that a voter has a local and trusted device (e.g., mobile phone or personal computer) that can do mathematical/cryptographic operations. A natural idea would be to allow the voter to generate the ballot using that trusted device. However, letting the voter generate the ciphertext on its own immediately leads to the possibility of coercion. The coercer can require the voter to disclose the utilized randomness or force the voter to use a compromised device.

Unfortunately, even an interactive approach with some (possibly multiple) rounds of communication with the voter does not guarantee compatibility with CR. Next, we discuss the shortcomings of the most common interactive CAI verification strategies.

#### Solution $U_1$: cast-or-challenge

It is a quite popular solution [39], that allows the voter to either proceed and send the vote or do the verification. If the voter chooses to audit, the voting device $VD$ reveals the randomness $r$ and the ciphertext $C = \text{Enc}_{pk}(m;r)$. The voter then uses another device to reconstruct $C$ and check that it is correct. The voter should repeat the verification a random number of times $k$ until the voter finally decides to casts the ciphertext.

The technique is wildly used by Helios [1] and all its various derivatives. It gives some assurance that $VD$ behaves as desired as, theoretically, it cannot predict after how many tries $k$ the voter decides to stop the audit. However, we cannot audit the sent ballot; hence there is still a chance that the $VD$ guessed the number of expected verifications and altered the ballot. Moreover, in practice, voters tend to perform only one, if any, audit. The studies [40,41] show that on average 43% of users understand cast-or-challenge verification.

#### Solution $U_2$: $\Sigma$-protocols

A different possibility is to use an interactive $\Sigma$-protocol to convince the voter $V$ that the ciphertext $C$ encrypts the expected vote $m$. Unfortunately, $\Sigma$-protocol is honest-verifier zero-knowledge only; hence it is not providing CR. The coercer can force the voter to choose the challenge $e$ in a particular way, so it will be computationally infeasible to simulate a valid transcript. For example,
if the coerer demands to use $e = \text{Hash}(a)$, where $a$ is the message voter sees before introducing the challenge, then the voter has to obey. This problem has appeared previously in the literature: a solution proposed in [42] to achieve receipt-freeness was claimed to be secure because the transcript of a $\Sigma$-protocol “is not transferable”. As already noted in footnote 1 of [10], this is not true when the voter (who plays the role of the verifier in the $\Sigma$-protocol) is dishonest/coerced.

There are two ways to solve this problem. We can use Neff’s approach [32], which requires voters to have access to a trusted “random beacon” during the voting protocol for choosing the challenge uniformly and at random. The other way, which, in our opinion, is more realistic in real life, is to use a full zero-knowledge interactive system instead of a $\Sigma$-protocol (which is only honest-verifier zero-knowledge). The resulting (satisfactory) solution is described and analyzed in Section 5.

Solution $U_2$: OR-proofs. Designated verification can help to construct CAI verification mechanisms as well [3,43]. For instance, assume that a voter knows a trapdoor $x$ that generates an instance of some hard relation $y$, say $y = g^x$. Then, the voting device $VD$ can prove CAI by providing a non-interactive OR-proof of the following form: “I know $r$ such that the ciphertext $C = \text{Enc}_{g}(m; r)$ OR I know $x$ such that $y = g^x$.”

Since $VD$ does not know the trapdoor $x$, it can only generate a valid OR-proof for the $C$ that contains the expected option $m$. At the same time, the voter can use the knowledge of $x$ to simulate as many transcripts as necessary.

Unfortunately, the CR of this method heavily relies on the fact that an (honest) $V$ does not know the witness $x$, which is not always the case. The coerer can force the voter to run the protocol for a specific $y$, for which the voter $V$ does not know the trapdoor $x$, thus preventing any possibility of simulating another valid transcript.

A solution proposed in [43] to avoid this situation is that $V$ must convince $P$ at some point, in zero-knowledge, of the knowledge of the witness $x$. However, if it is done offline before the voting phase, the coerer, who controls the voter $V$, can force the voter $V$ to act as a proxy while performing all operations by himself. Therefore, the only solution is to add this modification to the online voting phase. The resulting protocol, which increases the number of rounds of the interaction of this idea $U_3$, from two to four, was first described in [19] and is recalled in Appendix.

We want to stress that designated verification protocols have been used in previous works as a tool to achieve coercion-resistance. In particular, they are used in the solution in [10], not in the voting phase (which is the focus of this work), but in the registration phase where voters obtain the credentials that they will use later in the voting phase.

5. A generic protocol achieving CAI+CR

Among the two satisfactory solutions to the CAI+CR problem described in [19], we will focus on the first one, $S_1$, because it leads to more efficient specific instantiations. The description of the generic version of $S_1$ is included in Appendix for completeness.

The solution $S_1$ is an improvement of unsatisfactory solution $U_2$. Recall that the solution $U_2$ inherits honest-verifier zero-knowledge, and not full zero-knowledge, from its underlying $\Sigma$-protocol. This, unfortunately, implies that the coerer can force a specific distribution of the challenges, which will make the transcripts not simulatable for instances out of the language.

Luckily, we can easily solve this problem by using a full zero-knowledge interactive protocol for the language $L_{u_2} = \{(C, m) | C = \text{encryption of } m\}$. To do so, we can use a well-known technique [44]: make the voter $V$ commit to the challenge $e$ before receiving anything from the voting device. In turn, the voting device $VD$ will accept the challenge $e$ only if it corresponds to the commitment previously received from $V$.

Since we will need a commitment scheme, let us recall the syntax of such a protocol and its relevant security properties. A commitment scheme $\text{Com} = (\text{Com.St.}, \text{Com.Eval})$ consists of two protocols. The setup one takes as input a security parameters and outputs some public parameters $\text{Com.param}$ along with plaintext, randomness and commitment spaces $\text{Com}.m, \text{Com}.r, \text{Com}.C$. The evaluation protocol takes as input $\text{Com.param}$, a plaintext $e \in \text{Com}.m$ and randomness $r \in \text{Com}.r$, and outputs an element $c \in \text{Com}.C$. We omit for simplicity the use of $\text{Com.param}$ as input, and denote an execution of the evaluation protocol as $c \leftarrow \text{Com.Eval}(e, r)$. A commitment scheme is hiding if a commitment $c$ does not leak any information about the committed plaintext $e$, in a semantic security sense (the adversary can choose two plaintexts and try to guess which one has been committed). A commitment scheme is binding if an adversary cannot open a commitment in two different ways, that is, it cannot output $(e, r), (e', r')$ such that $(e, r) \neq (e', r')$ and $\text{Com.Eval}(e, r) = \text{Com.Eval}(e', r')$.

5.1. Description of the protocol

The setup algorithm $\text{Setup}(\lambda) \rightarrow \text{Com.param}$ of the voting scheme runs $\text{Com.St}(\lambda)$ resulting in $\text{Com.param}$, $\text{Com}.m, \text{Com}.r, \text{Com}.C$, and runs $\text{ENK}(\lambda)$ resulting in a public key $\mathcal{p}$ for the encryption scheme $E$, along with spaces $\mathcal{E}_g, \mathcal{E}_z, R, \mathcal{S}_g, \mathcal{S}_z$. We assume that the $\Sigma$-protocol $(P, V)$ for $\mathcal{E}_g$ has challenge space $\mathcal{C}$ (with size exponential in $\lambda$) and produces transcripts $(a, e, z)$. We also assume that $\mathcal{C} \subseteq \mathcal{E}_g$. All these public parameters of $\mathcal{E}_g$ and $E$ form the public parameters $\text{Com}.param$ of the voting protocol.

An execution of the voting protocol,

$\text{Vote}^{(V, D)}(\text{Com.param}, m, \theta) \rightarrow (b_V, b_D, \text{Trc, rand}_V),$

with coer $= \theta$, works as follows:

1. The voter $V$ chooses $e \in \mathcal{C}$; $i \in \mathcal{S}_g$ and computes $cmt = \text{Com.Eval}(e, i)$. It sends $(m, cmt)$ to the voting device $VD$.

2. If $m$ is not a valid voting option, $VD$ aborts and sets $b_D = 0$. Otherwise $VD$ chooses $i \in \mathcal{S}_g$ uniformly at random and computes $C = \mathcal{E}_g(\text{Enc}(m, i))$. Then it runs the first step of the $\Sigma$-protocol for $\mathcal{E}_g$, with public input $i = (m, C)$ and witness $w = r$, to produce a value $a$. It sends $(C, a)$ to $V$.

3. $V$ sends $(e, i)$ to $VD$.

4. $VD$ checks if $e \in \mathcal{C}$, if $i \in \mathcal{S}_g$ and if $cmt = \text{Com.Eval}(e, i)$. If this is not the case, or if $VD$ received nothing in step 3, $VD$ aborts and sets $b_D = 0$. Otherwise $VD$ uses challenge $e$ to compute the final value $z$ of the $\Sigma$-protocol for $\mathcal{E}_g$. Finally $VD$ sends $z$ to $V$ and sets $b_D = 1$.

The voter $V$ accepts the interaction (and so $b_V = 1$) if and only if $(a, e, z)$ is a valid transcript for the $\Sigma$-protocol for $\mathcal{E}_g$, on public input $i = (m, C)$.

If $b_D = 1$, then $VD$ proposes $C$ to $V$ as the encrypted vote to be sent to the ballot box. If $b_V = 1$, then $V$ accepts this proposal and $C$ is sent to the ballot box.

Following the notation introduced in Section 3.1, we have $\text{rand}_V = (e, i)$ and $\text{Trc} = (m, C, a, e, z, cmt, i)$.

5.2. Security analysis of the protocol

**Theorem 1.** Assuming the special soundness of the $\Sigma$-protocol for $\mathcal{E}_g$ and the hiding property of $\text{Com}$, the protocol described in Section 5.1 achieves the Cast-as-Intended (CAI) property (Definition 1).

**Proof.** Remember that the CAI property is satisfied if the probability of Cheat is negligible in the security parameter $\lambda$, where Cheat is the event where $\text{Setup}(\lambda) \rightarrow \text{Com.param}$ and $\text{Vote}^{(V, D)}(\text{Com.param}, m, \theta) \rightarrow (b_V, b_D, \text{Trc, rand}_V)$ are executed and $b_V = 1$ and $C \in \text{Trc}$ satisfies $C$ s.t. $m \neq \text{Dec}(C, sk)$. 
Since the commitment scheme Com is hiding, the voting device \(VD\) does not gain any information on the challenge \(e\) before choosing the first message \(a\) of the \(\Sigma\)-protocol. In this setting, using also the fact (see for instance Proposition 6.2.3 in [44]) that the special soundness property of a \(\Sigma\)-protocol implies soundness of the interactive system with soundness error \(e^{\Sigma} \in \negl(\lambda)\), we directly get that the probability that event \(\text{Cheat happens}\) happens, which means that a transcript for the interactive system is accepted for an instance out of the language, is negligible in \(\lambda\) \(\Box\).

**Theorem 2.** Assuming the special honest-verifier zero-knowledge property of the \(\Sigma\)-protocol for \(R_E\), the IND-CPA security of \(E\) and the binding property of \(Com\), the protocol described in Section 5.1 achieves the Coercion-Resistance (CR) property (Definition 2).

**Proof.** Other than choosing its preferred voting option \(m^*\), the adversary \(A\) may eventually force the voter to deviate from the correct execution (of some of) the steps in the voting protocol. This part of the coercion may affect essentially the distribution of the elements \(e, \hat{r}, C?, m\), which are the only ones generated/sent by the voter. We will model this fact with three functions \(f_e, f_\hat{r}, f_m\) that describe how these three elements are to be computed by the coerced voter. These functions may depend on the voting option \(m^*\) and on values previously exchanged during the protocol, for instance \(f_e\) might be a function of \(m^*, C, a, c, m\).

Let us consider first coercions where the voter is forced to choose \(e\) or \(\hat{r}\) (in round 3 of the protocol \(Vote\)) as a function of the values \(a, C\) sent by \(VD\) in round 2. This means the value \(cmt\) sent by \(V\) in round 1 has been computed before the values \(e, \hat{r}\), unless the value \(a\) has been guessed in advance, which happens with negligible probability. In such a situation, it must be \(cmt \neq \text{Com.Eval}(e, \hat{r}; V)\); otherwise, such a coerced execution could be used to break the binding property of the commitment scheme \(Com\).

Therefore, step 3 in the CR experiment for these coercions will lead to \(VD\) rejecting the \(Vote\) protocol in Round 4, without answering the final \(z\) and setting \(b_{VD} = 0\). Thus, the CR experiment will abort.

For the rest of coercions, in step 3 of the CR experiment, the first execution of the \(Vote\) protocol results in \(C^{(0)}\) being an encryption of \(m^*\), where both \(m^*\) and \(C^{(0)}\) are part of \(Tr(c)\).

The not-coerced execution of \(Vote\) run by \(Sim_1\) in step 5, with input message \(m\), results in \(C^{(1)}\) being an encryption of \(m\), where both \(m\) and \(C^{(1)}\) are part of \(Tr(c)\).

Finally, what \(Sim_2\) does in step 6 is:

1. Since now the values of \(e, \hat{r}\) do not depend on the values \(a, C\), values \((e^{(1)}, \hat{r}^{(1)}, cmt^{(1)})\) can be chosen first by \(Sim_2\), following the instructions in \(f_m, f_c, f_s\). Since we are assuming that the CR experiment does not abort, these values satisfy \(cmt^{(1)} = \text{Com.Eval}(e^{(1)}, \hat{r}^{(1)})\).

2. \(Sim_3\) runs the simulator \(R_{R_E}\) associated to the zero-knowledge property of the \(\Sigma\)-protocol for \(R_E\) with input \(x' = (m^*, C^{(1)})\), which is an element out of the language \(L_E\), for the challenge \(e^{(1)}\). Let \((a^{(1)}, e^{(1)}, \hat{r}^{(1)})\) be the output of \(R_{R_E}\).

3. \(Sim_3\) ends by defining \(Tr^{(1)} = (m^*, C^{(1)}, a^{(1)}, e^{(1)}, \hat{r}^{(1)}, C^{(1)}, \text{cmt}^{(1)}, \hat{r}^{(1)})\).

The computational indistinguishability between the distributions of \(view^{(0)}\) and \(view^{(1)}\) is directly implied by the zero-knowledge property of the \(\Sigma\)-protocol for \(R_E\) and the IND-CPA security of \(E\), as stated in Lemma 1. \(\Box\)

**6. Specific instantiations**

**6.1. Detailed protocol for ElGamal ciphertexts**

Let us detail the protocol that we obtain if we instantiate the generic construction from previous section with ElGamal public key encryption scheme and Pedersen commitment scheme [45]. The public parameters of the election system must contain elements \(g, G, g, h\) such that \(G = \langle g \rangle = \langle h \rangle\) has prime order \(q\).

The resulting protocol, with four rounds of communication, works as follows:

1. \(V\) chooses \(e, \hat{r} \in Z_q^*\) uniformly at random, computes the commitment \(Z = g^{e \cdot h^r}\) and sends \(Z\) to \(VD\), together with the chosen voting option \(m\).
2. If \(m\) is not a valid voting option, \(VD\) aborts and sets \(b_{VD} = 0\). Otherwise \(VD\) chooses \(r, t \in Z_q^*\) uniformly at random, computes \(C = (c_1, c_2) = (g^r, m \cdot p^k)\) and computes \(a = (A_1, A_2) = (g^r, p^k)\). Both \(C\) and \(a\) are sent to \(V\).
3. \(V\) sends \((e, \hat{r})\) to \(V\).
4. \(VD\) checks if \(e, \hat{r} \in Z_q^*\) and if \(Z = g^{e \cdot h^r}\). If this is not the case, \(VD\) sets \(b_{VD} = 0\) and aborts. Otherwise, \(VD\) computes \(z = t + r \mod q\) and sends \(z\) to \(V\).

The voter \(V\) will accept if the two following equalities hold: \(g^z = A_1 \cdot c_1^z\) and \(p^z = A_2 \cdot \left(C \frac{g}{m} \right)^z\).

**6.2. An instantiation with post-quantum security**

ElGamal encryption scheme is not secure in front of adversaries equipped with a quantum computer. For those settings, the most promising cryptographic techniques seem to be lattice-based ones. For instance, a suitable public-key encryption scheme could be one based on the hardness of the Ring Learning With Errors (RLWE) problem [46]. Due to its homomorphic properties, it is possible to find a suitable \(\Sigma\)-protocol to prove, for this encryption scheme, knowledge of randomness \(r\) such that \(C = \text{Enc}_{g^r}(m; r)\).

Such a \(\Sigma\)-protocol can be obtained as a particular instance of some of the recently proposed and generic constructions of lattice-based zero-knowledge interactive systems (e.g. [47–49]).

The only missing part to instantiate the generic construction described in Section 5.1 is a quantum-safe commitment scheme. The easiest (and most efficient) possibility would be to consider as the commitment a hash function \(H\) modeled as a random oracle, so that \(\text{Com.Eval}(c, r) = H(c \parallel r)\). If one wants to get security in the standard model, not in the random oracle model, one can choose specific lattice-based commitment schemes, such as those proposed in [50,51].

**7. Conclusion**

In this paper, we have considered two desirable properties for an electronic voting system: coercion-resistance and cast-as-intended verifiability. We have focused our analysis on systems where:

- there are no secure delivery channels between the election authority and voters,
- each voter interacts with a voting device to produce an encryption of the intended vote, which will be sent to the ballot box.

We have formalized the two properties locally and individually, restricted to the behavior of one voter during the vote-casting phase of the election. This differs from other previous definitions (in particular of coercion-resistance) that consider global adversaries that can coerce many voters/authorities, which are active during all election phases. However, we emphasize that our local definitions make perfect sense in realistic e-voting settings when one is interested in avoiding individualized attacks on (or malicious behaviors of) a single voter.

We have shown that some (explicitly or implicitly) proposed solutions for e-voting in this setting cannot offer the two properties at the same time, and we then have shown how some of these solutions can be modified to get e-voting solutions that enjoy both CAI and CR in our considered setting. We have formally proved the CAI and CR properties of one of these solutions and discussed how it could be instantiated in both the pre-quantum (classical) and the post-quantum cryptographic settings.

As open lines for future work, we first stress that the type of coercion that we have considered in this work is limited to the coerced forcing the voter \(V\) to use some specific values and functions when computing and sending the elements that \(V\) sends during the execution.
of Vote. In real life, there can be other (perhaps stronger) coercion settings; for instance, a coherer \( A \) could require that the voter \( V \) sends some particular element of \( \text{Trc} \) to a time-stamped public place (like a Blockchain), to detect at what time(s) the voter is executing each part of Vote. In such settings, the satisfactory solutions analyzed in this paper would no more enjoy the CR property. A challenging open problem is to formally analyze such stronger settings for coercion and find protocols that enjoy both CAI and CR or eventually prove that such solutions cannot exist.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work has been supported by the Spanish Ministerio de Ciencia e Innovación (MICINN), under Project PID2019-109379RB-I00, and by Generalitat de Catalunya, under Project DI 2018-038.

Appendix. A second generic protocol

A second solution to the CAI+CR problem was proposed in [19], built upon the unsatisfactory solution \( U \), that we discussed in Section 4: \( \text{VD} \) computes at the end a proof for the OR language \( \{ \text{C} \} = \{ \text{OWF}(m) \} \) for some one-way function \( \text{OWF} : X \to Y \). The idea woks if \( V \) actually knows the trapdoor \( X \). But the way \( V \) convinces \( \text{VD} \) of this fact cannot be a non-interactive proof of knowledge: an active coherer could generate an instance \( (X, Y) \) and such a non-interactive proof of knowledge \( \pi \) of an instance \( (X, \pi) \) to \( V \), keeping the trapdoor \( X \) secret. In this way, the voter \( V \) could not simulate valid transcripts for voting options different from the one inside the ciphertext \( C \): the protocol would not be coercion-resistant. The solution is to impose that the proof of knowledge of \( X \) is itself interactive (and simulatable). The details follow. We assume that the \( \Sigma \)-protocol \((P, V)\) for \( \text{OWF} \) is such that \( \{ (X, Y) : X \in \Lambda \times Y \mid Y = \text{OWF}(X) \} \) is challenge space \( \chi \) (with size exponential in \( \lambda \)) and produces transcripts \((a, e, z)\). We also assume that secure (in particular, witness-indistinguishable) non-interactive zero-knowledge proofs of knowledge for the OR relation defined above exist.

An execution of the voting protocol,

\[ \text{Vote}^{(V, \text{VD})}(pms, m, \emptyset) \rightarrow (b_V, b_{\text{VD}}, \text{Trc}, rand_V) \]

with coer = \( \emptyset \), works as follows:

1. \( V \) chooses \( X \in \Lambda \) at random, computes \( Y = \text{OWF}(X) \) and runs the first step of the \( \Sigma \)-protocol for \( \text{OWF} \), with public input \( Y \) and witness \( x \) to \( X \). It sends \((Y, a, m)\) to \( \text{VD} \).
2. If \( m \) is not a valid voting option, \( \text{VD} \) aborts and sets \( b_{\text{VD}} = 0 \). Otherwise \( \text{VD} \) chooses \( r \in R_S \) uniformly at random and computes \( C = \text{Enc}(m, r) \). Finally \( \text{VD} \) checks at random \( e \) to the ciphertext \( C \) \( \text{VD} \) checks at random \( e \) to the ciphertext \( C \) and sends \((C, e)\) to \( V \).
3. \( V \) chooses challenge \( e \) to compute the final value \( z \) of the \( \Sigma \)-protocol for \( \text{OWF} \), and sends \( z \) to \( \text{VD} \).
4. \( \text{VD} \) checks if \((a, e, z)\) is a valid transcript for the \( \Sigma \)-protocol for \( \text{OWF} \), on public input \( Y \). If this is not the case, or if \( V \) received nothing, \( \text{VD} \) aborts and sets \( b_{\text{VD}} = 0 \). Otherwise \( \text{VD} \) uses witness \( r \) to compute a non-interactive zero-knowledge proof \( \pi \) of knowledge of:

\[ r \text{ s.t. } C = \text{Enc}(m, r) \text{ OR } X \text{ s.t. } Y = \text{OWF}(X) \]

The voter \( V \) accepts the interaction (and so \( b_V = 1 \)) if and only if \( \pi \) is a valid proof of knowledge for the OR relation above.

The intuition of why this protocol is secure is as follows. For the CAI property, since \( \text{OWF} \) is a secure one-way function and the \( \Sigma \)-protocol for it is zero-knowledge, \( \text{VD} \) learns nothing about the secret value \( X \). Therefore, a valid \( \pi \) coming from \( \text{VD} \) must mean that the employed witness has been a proper randomness \( r \), and thus interactions accepted by \( V \) contain a ciphertext \( C \) that encrypts the chosen option \( m \).

For the CR property, if coercion is such that \( V \) does not足够 correctly (with \( b_{\text{VD}} = 1 \)), then this means that \( V \) must be able to successfully run the \( \Sigma \)-protocol for \( RS_{\text{OWF}} \), which means \( V \) knows (or can extract) the witness \( X \) and use it to compute valid non-interactive zero-knowledge proofs for the OR relation

\[ C = E_{\text{Enc}}(m^* r) \text{ OR } Y = \text{OWF}(X) \]

even if the first clause is false, because the ciphertext that has gone to the ballot box is an encryption of \( m \). On the other hand, if coercion is such that \( \text{VD} \) aborts the protocol, then the value of \( r \) must not be simulated, and the task of \( \text{Sim} \) is even easier.

The instantiation of this generic construction for the case of ElGamal ciphertexts, and using exponentiation as \( \text{OWF} : z \mapsto g^z \) defined by \( \text{OWF}(X) = g^X \), can be found in [19].

References