Coercion-Resistant Cast-as-Intended Verifiability for Computationally Limited Voters

Tamara Finogina[1,2] and Javier Herranz[1](✉)

[1] Dept. Matemàtiques, Universitat Politècnica de Catalunya, Barcelona, Spain
javier.herranz@upc.edu

[2] Scytl Election Technologies, S.L.U., Barcelona, Spain
tamara.finogina@scytl.com

Abstract. In this work, we investigate if two essential properties in electronic voting, coercion-resistance and cast-as-intended verifiability, can be jointly achieved in settings where voters are (very) limited from a computational point of view. This may be the case in elections where voters use a voting station or webpage to cast their votes but do not have specialized software or devices to perform complicated cryptographic operations (for instance, to verify zero-knowledge proofs or to generate one-way trapdoors).

We provide a solution where the only things voters have to do are: remember and compare strings of numbers, on the one hand, and press a button at the appropriate moment, on the other hand. This button activates the participation of an online entity, which is trusted to choose a random nonce for each voter and to publish it only when that voter presses the button (and not before). The most expensive part of the verification is an OR proof of knowledge, which can be done by any (powerful enough) external verifier.

1 Introduction

Coercion resistance and cast-as-intended verifiability are two crucial but contradictory properties of electronic voting schemes. The former aims not to leak any information that can prevent voters from expressing their true intent or can facilitate vote-selling. The latter tries to ensure that a cheating voting device would not be able to modify a voter's choices undetectably. The contradiction is unavoidable: one property requires outputting as little feedback regarding the vote's content as possible, while the other demands proving that the voter's intention is encrypted correctly.

In this work, we focus on this contradiction and outline the definition of coercion-resistant cast-as-intended verification in the simplest (but maybe most realistic) case for electronic elections: a voter cannot perform any complex computations on its own, for instance, because it does not have any trusted computational device capable of doing sophisticated math.

While, to our knowledge, there are no formal studies about voters' modeling or capabilities, we believe this simplest possible case is the most realistic assumption one can make about voters. It is true that, with less restrictive assumptions about voters' abilities, we perhaps can construct more elegant and secure systems. However, one may wonder - if the voter already has a trusted device or can do cryptographic operations in the head, why not use it for vote-casting directly?

As was mentioned, we consider only computationally limited voters who want to perform cast-as-intended verification while remaining safe from possible coercion threats. However, our task is not trivial: the lack of consensus regarding the coercer's abilities or voters' capabilities to withstand coercion already makes defining coercion resistance challenging. Furthermore, the matter gets more complicated by the differences in techniques for cast-as-intended verification.

1.1 On the Limitations of the Coercion and of Voters' Capabilities

Coercion is an intuitively-understandable threat. While it is clear that the goal of the coercer is to prevent free will expression, some specific details are still blurry:

- "Can a coerced prevent a voter from voting?"
- "Can an average voter extract randomness and similar secret details from the voting device (when it does not output them by default)?"
- "Are voters left without a coercer's observation during the voting?"
- "Is the coercer's goal to vote for a specific option or not?"

We can firmly agree that coercion-resistance makes sense only when the coercer has limited observational power [7], and the voter can disobey without facing high personal risks. Else, the voter has no choice but to comply [25]. Also, it was proven that resisting force-abstention attacks requires anonymous channels [13], which are incredibly hard to establish in practice. Also, we focus only on coercion that aims to change a voter's vote to a specific option or candidate. We do not deal with scenarios where the coercer wishes to delay the vote-casting process, discourage participation, or complicate voters' lives otherwise.

In this work, we assume that voters are computationally limited entities and cannot do cryptographic operations (one-way function computations, etc.). The only capability voters have is remembering and comparing strings they see - an assumption similar in spirit to [18,25]. For simplicity, we do not restrict the length of memorized data; however, we realize there is only so much information the voter can safely remember. Since we consider voters computationally limited, we exclude the possibility of force extraction of non-outputted information from an e-voting system. While it is true that, in some cases, the coercer might demand voters extract randomness used for encrypting their choice, we believe those instructions are hard to comply with for an average non-technically savvy voter.

Therefore, we consider the following scenario: a computationally limited (and potentially malicious) voter is coerced to vote for a specific option but left without supervision for (some part) of the vote-casting process. The coercer cannot:

prevent voters from voting, impersonate them, or demand data not provided by the voting protocol.

1.2 Related Work

The first proposal for cast-as-intended verification based on visual cryptography appears in Chaum's work [5], followed by an idea of simulatable zero-knowledge proofs in Neff's publication [18] the same year. In both proposals, the vote-casting happens in a polling place, and both cast-as-intended techniques require an additional trusted device (a specialized printer or randomness generator). A few years later, Benaloh [2] suggested another idea - the cast-or-challenge mechanism, which allows voters to verify multiple test ballots before finally casting the real one. It started as a pulling place testing technique based on printed commitment but quickly transformed into ballot casting assurance for remote voting [1]. Nowadays, there are many more ways to ensure a ciphertext contains the intended vote: OR proofs [10], return codes [14], tracking numbers [22], QR codes [24], etc.

The first mention of uncoercibility and receipt-freeness goes back to the election protocol designed by Benaloh and Tuinstra in 1994 [25]. A few years later, the first definition of what is now known as coercion-resistance was proposed [19]. The notion was called receipt-freeness; however, the coercer could interact with the voter during the voting phase, which implies adaptive corruption, in other words, coercion. The first widely accepted formal definition of coercion-resistance appeared only in 2010 [13]. This JCJ definition formalizes the idea of anonymous credentials and accounts for vote-selling and forced-abstention attacks. Unfortunately, achieving such a level of coercion-resistance in practice requires anonymous voting channels and effectively excludes any form of cast-as-intended verification. Interestingly, an almost unique scheme that satisfies this JCJ highly demanding notion of coercion-resistance scheme was found to leak too much information in case of re-voting, and so not to be truly coercion-resistant [6] - an attack first discovered as the "1009 attack" in [23]. Hence, the notion was enhanced by adding a cleansing-hiding procedure. The list of definitions of coercion-resistance is not limited to the JCJ derivations only. There exist many different approaches that focus on various coercion threats [4,7,9,13,16,17,25]. Unfortunately, they all struggle to capture the broadness of possible coercion strategies [11,15,21].

To our knowledge, not many articles try to combine coercion-resistance (not receipt-freeness) with any other property. We can mention: a study about relations between privacy, verifiability, accountability and coercion-resistance [20], a paper proposing to combine JCJ with Selene [12] and a discussion regarding cast-as-intended coercion resistance in settings without trusted delivery channels [8]. The solution in [12] inherits some of the (good and bad) properties of JCJ: it can resist force-abstention attacks, but the price to pay is that (i) the system must allow multiple voting, and (ii) coercers can vote on behalf of voters (which means that some part of the voting scheme is not properly authenticated).

Furthermore, all the papers mentioned in the last paragraph consider settings where the voters have the capability to run expensive/complicated (cryptographic) operations on their own. Regarding voting schemes for computationally limited (also known as human) voters, the setting we consider in this work, we can mention Neff's proposal [18], a scheme by Moran-Naor [17], Bingo voting [3] and other references therein. Authors of Bingo voting show in [3] that many of the other protocols in this setting can suffer from a specific coercion attack that they call "Babble attack". This fact leads to the idea that an external (and trusted) entity may be necessary to choose random elements on behalf of voters; this approach is taken by the Bingo voting protocol and also by the solution that we propose in this paper. We will discuss the similarities and differences between our solution and Bingo voting in Sect. 4.2.

1.3 Contributions and Organization of the Paper

In this work, we discuss voters' and coercer's limitations and propose definitions that, as we believe, capture both coercion-resistance and cast-as-intended properties without requiring voters to do complex operations (Sect. 2). The definitions are done using the standard cryptographic language for provable security (experiments, games, challengers, adversaries); furthermore, even if their goal in this paper is to analyze the security of a protocol for computationally limited voters, the definitions are written in a so general way that they could be used to analyze the cast-as-intended and coercion-resistance properties of other voting protocols.

Section 3 contains our solution of coercion-resistant cast-as-intended verification for a computationally limited voter and the proofs that it fulfills the two above-mentioned definitions. Last, we give in Sect. 4 a discussion on some practical aspects of our solution, a comparison between our solution and Bingo voting [3] and an intuitive argument on why a trusted device seems necessary in order to achieve both coercion-resistance and cast-as-intended verification in this setting of limited voters.

2 Definitions

We focus on a specific computationally limited voter $\mathcal{V}$ who wishes to cast a vote for the intent m through a voting device $\mathcal{VD}$, possibly under the coercion of an adversary $\mathcal{A}$ that prefers another option $m^\star \neq m$. Let Cpb denote the class of operations the voter $\mathcal{V}$ is supposed to be capable of doing. In the restricted setting we consider in this paper, this class contains: generating, memorizing, and comparing strings of numbers.

In such a setting, it makes perfect sense to consider the possibility that $\mathcal{V}$ gets the help of another entity or device, that we call official election device (OED for short), to execute some of the steps of the protocol. It is trusted to run the prescribed steps of the protocol correctly, and its actions are free from coercion.

We assume that the public election parameters pms include candidates, questions, election rules, mathematical descriptions of the group, a security parameter λ, hash functions, the public key pk of the encryption scheme Enc for encrypting the votes, etc. To generate pms one would run an initial protocol $\mathsf{pms} \leftarrow \mathsf{Setup}(\lambda)$.

The voting protocol itself is an interactive protocol between the voter $\mathcal{V}$ and the voting device $\mathcal{VD}$. We denote as $\mathsf{Trc} = (\mathtt{C}, \mathsf{Trc}_{\mathsf{pub}}, \mathsf{Trc}_{\mathsf{priv}})$ the result of the said interaction, where the ciphertext $\mathtt{C}$ encrypts (in principle) the voting option m chosen by $\mathcal{V}$ and the public trace $\mathsf{Trc}_{\mathsf{pub}}$ contains other messages that will be made public in the bulletin board (proofs, signatures, voter ID, etc.), along with $\mathtt{C}$. The rest of voter's (private) view of the interaction with $\mathcal{VD}$ is denoted as $\mathsf{Trc}_{\mathsf{priv}}$.

We denote an execution of the voting protocol as:

$$\mathsf{Trc} = (\mathtt{C}, \mathsf{Trc}_{\mathsf{pub}}, \mathsf{Trc}_{\mathsf{priv}}) \leftarrow \mathsf{Vote}^{\langle \mathcal{V}^{\mathsf{OED}}, \mathcal{VD} \rangle}(\mathsf{pms}, m, \mathsf{Cpb}, \mathsf{coerc})$$

Here coerc refers to a possible set of instructions forced to the voter $\mathcal{V}$ by a coercer; in case of no coercion, this variable is set to $\emptyset$. All instructions the coercer gives should be within the voter's capabilities; in other words, the actions are restricted to the class Cpb.

In our setting, where the voter has limited computational capabilities, it is clear that some part of the verification (which involves ciphertexts and thus expensive cryptographic operations) must be done by an external verifier. But of course, another part of the verification is done by the own voter $\mathcal{V}$.

The first (public) verification is done by anyone (with enough computational capabilities) by running a protocol

$$\mathsf{ValidProof}(\mathsf{pms}, \mathtt{C}, \mathsf{Trc}_{\mathsf{pub}}) \rightarrow \{0,1\}$$

The second (private) verification is done by $\mathcal{V}$ by running the protocol below, whose operations must belong to class Cpb:

$$\mathsf{ValidOption}(\mathsf{pms}, \mathtt{C}, \mathsf{Trc}, m, \mathsf{Cpb}) \rightarrow \{0,1\}$$

2.1 Cast-as-Intended Verifiability

Intuitively, this property must capture the impossibility that a dishonest voting device $\mathcal{VD}$ succeeds in cheating the voter $\mathcal{V}$ by completing an accepted execution of the protocol Vote but in which the resulting ciphertext (published in the bulletin board) encrypts something which is not the voting option chosen by $\mathcal{V}$.

The corresponding event Cheat is defined as follows:

$$\mathsf{Cheat} = \begin{cases} \mathsf{pms} \leftarrow \mathsf{Setup}(\lambda) \\ \mathsf{Vote}^{\langle \mathcal{V}^{\mathsf{OED}}, \mathcal{VD} \rangle}(\mathsf{pms}, m, \mathsf{Cpb}, \emptyset) \rightarrow (\mathtt{C}, \mathsf{Trc}_{\mathsf{pub}}, \mathsf{Trc}_{\mathsf{priv}}) \\ \mathsf{ValidProof}(\mathsf{pms}, \mathtt{C}, \mathsf{Trc}_{\mathsf{pub}}) \rightarrow 1 // \text{ public validity} \\ \mathsf{ValidOption}(\mathsf{pms}, \mathtt{C}, \mathsf{Trc}, m, \mathsf{Cpb}) \rightarrow 1 // \text{ private validity} \\ m \neq \mathsf{Dec}(\mathtt{C}, \mathsf{sk}) // \text{ but } \mathtt{C} \text{ does not contain the intent} \end{cases}$$

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* λ*, for any polynomial-time voting device* $\mathcal{VD}$*.*

2.2 Coercion-Resistance

In our setting, a coercer $\mathcal{A}$ (e.g., a vote buyer) may interact with the voter $\mathcal{V}$ before the vote-casting and force him to vote for some option $m^\star$ and to follow the instructions (belonging to the class Cpb) in coerc while executing Vote. The strategy coerc must be such that it does not affect the steps run by entity OED (which is assumed to be incoercible) and such that it leads to accepted executions of the protocol Vote. That is, we do not consider 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 $\mathcal{VD}$, which results in $\mathcal{VD}$ aborting the protocol and not sending any ciphertext to the ballot box.

During the execution of Vote, the adversary cannot see the interaction between $\mathcal{V}$ and $\mathcal{VD}$ (otherwise, since the voting option is sent by $\mathcal{V}$ to $\mathcal{VD}$ in clear, it would be impossible to achieve any meaningful coercion-resistance property). However, the adversary has access to the ballot box and can see the published values: the ciphertext C and the remaining public data $\mathsf{Trc}_{\mathsf{pub}}$. After the voting, $\mathcal{A}$ expects to receive from $\mathcal{V}$ the rest of the voter's view, $\mathsf{Trc}_{\mathsf{priv}}$, of the interaction.

To prevent coercion, the voter $\mathcal{V}$ must always be able to deceive the coercer. In other words, it should be able to run the protocol Vote with its voting option m and later simulate the view that would make $\mathcal{A}$ believe that Vote was run with $m^\star$ as input. All this must be done with the limited capabilities (in class Cpb) of the voter. We say the protocol Vote has coercion-resistance whenever the coercer $\mathcal{A}$ cannot distinguish between a result of voting for the coercer's preference and a simulated transcript hiding the disobedience with probability more than 1/2. 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* $\mathcal{A}$ *which chooses instructions* $\mathsf{coerc} \in \mathsf{Cpb}$ *that do not affect the steps run by* OED*, there exists a simulator* $\mathsf{Sim} \in \mathsf{Cpb}$ *such that, in the experiment described below,* $\left|\Pr[b' = b] - \frac{1}{2}\right|$ *is a negligible function of the security parameter* λ*:*

1. $\mathsf{pms} \leftarrow \mathsf{Setup}(\lambda)$.
2. $b \leftarrow \{0,1\}$ *is chosen uniformly at random.*
3. $\mathcal{A}(\mathsf{pms}) \rightarrow (\mathsf{coerc}, m^\star, m)$.
4. $\mathsf{Trc}^{(0)} = \left(\mathsf{C}^\star, \mathsf{Trc}^\star_{\mathsf{pub}}, \mathsf{Trc}^\star_{\mathsf{priv}}\right) \leftarrow \mathsf{Vote}^{\langle \mathcal{V}^{\mathsf{OED}}, \mathcal{VD}\rangle}(\mathsf{pms}, m^\star, \mathsf{Cpb}, \mathsf{coerc})$
   // obey the coercer and vote for $m^\star$
   If $\mathsf{ValidProof}(\mathsf{pms}, \mathsf{C}^\star, \mathsf{Trc}^\star_{\mathsf{pub}}) \rightarrow 0$*, abort // happens only if instructions* coerc *invalidate the ballot*

5. $\mathsf{Trc} = (C, \mathsf{Trc}_{\mathsf{pub}}, \mathsf{Trc}_{\mathsf{priv}}) \leftarrow \mathsf{Vote}^{\langle \mathcal{V}^{\mathsf{OED}}, \mathcal{VD} \rangle}(\mathsf{pms}, m, \mathsf{Cpb}, \emptyset)$ // *disobey the coercer and vote for m*
6. $\mathsf{Trc}_{\mathsf{priv}}^{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{Trc}, m^\star, \mathsf{Cpb})$ // *fake the voter's view*
   Set $\mathsf{Trc}^{(1)} = (C, \mathsf{Trc}_{\mathsf{pub}}, \mathsf{Trc}_{\mathsf{priv}}^{\mathsf{Sim}})$
7. $b' \leftarrow \mathcal{A}(\mathsf{pms}, \mathsf{Trc}^{(b)}, \mathsf{Cpb})$.

3 A Construction for Limited Voters

In this section, we explore possibilities for providing coercion-resistant cast-as-intended verification to voters with (very) limited capabilities, namely those who can only generate, remember and compare strings of numbers.

We propose and analyze a simple protocol where the voter needs the help of an external device OED for string generation. The OED is expected to participate in the protocol honestly and only when the voter indicates (and not a moment before).

The idea of our solution is as follows: the voter $\mathcal{V}$ chooses one of the ℓ possible voting options $m_j \in \{m_1, m_2, \dots, m_\ell\}$ and sends it to the voting device $\mathcal{VD}$, which encrypts the selected option into a ciphertext C. Along with C, the voting device $\mathcal{VD}$ will generate a non-interactive OR zero-knowledge proof of knowledge of the randomness r such that C is the encryption of *some* valid voting option in $\{m_1, m_2, \dots, m_\ell\}$.

To prevent a dishonest voting device from cheating the voter by encrypting a different voting option $m^* \neq m_j$, the zero-knowledge proof must be computed in stages: first, the voting device will show the voter some values (to be memorized), then the voter will press the button generating "Nonce", and only then the voting device will be able to finish the computation of the zero-knowledge proof, by using the corresponding nonce (selected by an official election device) as an input of the hash function.

Once the final proof is published, anybody (with enough computational resources) can verify the validity of the OR proof and ensure that a ciphertext contains a valid selection. However, only the voter can ensure that the proof is consistent with the memorized values, which implies that the ciphertext encrypts the desired choice.

The simplicity of the construction allows us to analyze its coercion-resistance and cast-as-intended properties according to our definitions. The intuition says that privacy only requires that the voting device does not leak the intent to the adversary - as all client-side encryption voting schemes. Similarly, coercion-resistance holds because voters' secret is something they saw rather than any secret key or value. Cast-as-intended property is ensured due to the soundness of OR proof and randomness of the "Nonce" provided by OED.

3.1 The Protocol (for ElGamal Ciphertexts)

Let us detail the protocol that we obtain if we use the ElGamal public key encryption scheme: public election parameters of the election system must contain elements q, G, g such that $G = \langle g \rangle = \langle h \rangle$ has prime order q and two collision-resistant hash functions $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ and $\hat{H} : \{0,1\}^* \rightarrow \mathcal{X}$, where $\mathcal{X}$ is the space of strings the voter can type and memorize. The public key of the election is $y \in \langle g \rangle$. The set $\mathcal{X}$ is a set with enough entropy to avoid brute force attacks.

1. $\mathcal{V}$ chooses a voting option m_j from the set $\{m_1, m_2, \ldots, m_\ell\}$ and sends the selected index $j \in \{1, 2, \ldots, \ell\}$ to $\mathcal{VD}$.
2. $\mathcal{VD}$ encrypts m_j using ElGamal encryption protocol:
   $\mathsf{C} = (c_1, c_2) = (g^r, m_j \cdot y^r)$, for some random and uniform $r \in \mathbb{Z}_q$.
   Now $\mathcal{VD}$ starts the computation of the OR proof as follows:
   (a) choose $t_j \in \mathbb{Z}_q$ uniformly at random;
   (b) compute commitments $A_j = g^{t_j}$, $B_j = y^{t_j}$;
   (c) for each $i \in \{1, 2, \ldots, \ell\}$, $i \neq j$:
      i. choose values $z_j, e_j \in \mathbb{Z}_q$ uniformly at random;
      ii. compute $A_i = g^{z_i} \cdot (c_1)^{-e_i}$ and $B_i = y^{z_i} \cdot \left(\frac{c_2}{m_i}\right)^{-e_i}$.

   $\mathcal{VD}$ sends to $\mathcal{V}$ the value $X_j = \hat{H}(\mathsf{C}, \{(A_s, B_s)\}_{1 \leq s \leq \ell}, \{e_i\}_{1 \leq i \leq \ell, i \neq j})$, to be memorized by $\mathcal{V}$.
3. At this point, $\mathcal{V}$ presses the "Nonce" button, which makes the official election device OED choose a random nonce $\mathsf{nc}_\mathcal{V} \in \mathcal{X}$ and publish $(\mathcal{V}, \mathsf{nc}_\mathcal{V})$ in the official public board of the election. If the publication of $(\mathcal{V}, \mathsf{nc}_\mathcal{V})$ is done in any other moment (for instance, by a coerced voter, before Step 2), the voting device $\mathcal{VD}$ aborts the execution.
4. Now $\mathcal{VD}$ can finish the OR proof:
   (a) computes $e = H(\mathsf{C}, \{(A_k, B_k)\}_{1 \leq k \leq \ell}, \mathsf{nc}_\mathcal{V})$;
   (b) sets $e_j = e - \sum\limits_{1 \leq i \leq \ell, i \neq j} e_i \mod q$;
   (c) finalizes the proof $z_j = t_j + e_j \cdot r \mod q$.

   Finally $\mathcal{VD}$ makes public the ciphertext C, the OR zero-knowledge proof $\pi = (\mathcal{V}, \mathsf{nc}_\mathcal{V}, \{(A_k, B_k, e_k, z_k)\}_{1 \leq k \leq \ell})$ and the list of couples $\{(m_k, \hat{X}_k)\}_{k \in \{1,2,\ldots,\ell\}}$, where $\hat{X}_k = \hat{H}(\mathsf{C}, \{(A_s, B_s)\}_{1 \leq s \leq \ell}, \{e_i\}_{1 \leq i \leq \ell, i \neq k})$.

Following the notation of Sect. 2, in our protocol we have:

$$\mathsf{Trc}_{\mathsf{pub}} = (\pi, \{(m_k, \hat{X}_k)\}_{k \in \{1,2,\ldots,\ell\}})$$

$$\mathsf{Trc}_{\mathsf{priv}} = (m_j, \{(m_i, X_i)\}_{i \in \{1,2,\ldots,\ell\}, i \neq j})$$

Private and Public Verifications. The voter $\mathcal{V}$ will accept the interaction if $\hat{X}_j = X_j$, for its chosen option m_j.

For the public verification, anybody with enough computational resources can first check that the initial couple $(\mathcal{V}, \mathsf{nc}_\mathcal{V})$ in π exists in the official public board of the election, and then ensure that all checks hold:

(i) $H(\mathsf{C}, \{(A_k, B_k)\}_{1 \leq k \leq \ell}, \mathsf{nc}_\mathcal{V}) = \sum\limits_{1 \leq k \leq \ell} e_k \mod q$,
(ii) for each $k \in \{1, 2, \ldots, \ell\}$, $g^{z_k} = A_k \cdot (c_1)^{e_k}$,

(iii) for each $k \in \{1, 2, \ldots, \ell\}$, $y^{z_k} = B_k \cdot \left(\frac{c_2}{m_k}\right)^{e_k}$,

(iv) for each $k \in \{1, 2, \ldots, \ell\}$, $\hat{X}_k = \hat{H}(\mathtt{C}, \{(A_s, B_s)\}_{1 \le s \le \ell}, \{e_i\}_{1 \le i \le \ell, i \ne k})$

We discuss how the outputs of these verification protocols affect the election in case of a dishonest behaviour of the voting device $\mathcal{VD}$ in Sect. 4.1.

3.2 Cast-as-Intended Verifiability of the Proposed Protocol

We prove that the protocol described in the previous section achieves cast-as-intended verifiability by using the rewinds. We assume a (dishonest) voting device $\mathcal{VD}$ can break cast-as-intended verifiability. In other words, it makes the event Cheat happen with a non-negligible probability. Then, we run Vote protocol without changes until step 3, where we make a fork and send two different nonces $\mathsf{nc}_{\mathcal{V}} \ne \mathsf{nc}'_{\mathcal{V}}$ to $\mathcal{VD}$. Since we assumed that event Cheat happens with a non-negligible probability, $\mathcal{VD}$ should be able to finish the two executions successfully and produce accepted proofs π and π'.

In the two executions, since the first two steps are identical and the hash function $\hat{H}$ is collision-resistant, we have that many values in π and π' are equal: m_j, $\mathtt{C} = (c_1, c_2)$, $\{(A_s, B_s)\}_{1 \le s \le \ell}$, $\{e_i\}_{i \in \{1,2,\ldots,\ell\}, i \ne j}$.

But since the two nonces are different, we have with overwhelming probability $e_j \ne e'_j$ as:

$$e = H(\mathtt{C}, \{(A_k, B_k)\}_{1 \le k \le \ell}, \mathsf{nc}_{\mathcal{V}}) \ne H(\mathtt{C}, \{(A_k, B_k)\}_{1 \le k \le \ell}, \mathsf{nc}'_{\mathcal{V}}) = e'$$

Now dividing the two satisfied equations $g^{z_j} = A_j \cdot (c_1)^{e_j}$ and $g^{z'_j} = A_j \cdot (c_1)^{e'_j}$ we get $c_1 = g^{\frac{z_j - z'_j}{e_j - e'_j}}$ on the one hand.

On the other hand, dividing the two satisfied equations $y^{z_j} = B_j \cdot \left(\frac{c_2}{m_j}\right)^{e_j}$ and $y^{z'_j} = B_j \cdot \left(\frac{c_2}{m_j}\right)^{e'_j}$ we get $c_2 = m_j \cdot y^{\frac{z_j - z'_j}{e_j - e'_j}}$.

Therefore, we have $\mathtt{C} = (c_1, c_2) = (g^{r_j}, m_j \cdot y^{r_j})$ for $r_j = \frac{z_j - z'_j}{e_j - e'_j} \mod q$, which means $\mathtt{C}$ is an encryption of the voting option m_j. This contradicts the fact that event Cheat was happening.

3.3 Coercion-Resistance of the Proposed Protocol

Coercion resistance is easier to argue. The voter's role in the protocol is limited - only choosing the intended voting option and pressing the "Nonce" button. Hence, the coercer $\mathcal{A}$ cannot enforce an elaborate voting strategy. The only possible instruction $\mathcal{A}$ can force voters to follow would be to vote for some option $m^\star$.

Theoretically, a coercer can force the voter to press the button at the wrong moment or not press it at all, but this would lead to a non-successful execution of the protocol, which our coercion-resistance definition does not take into account.

Similarly, our notion of coercion-resistance does not cover coercion strategies that require the voter to repeat vote-casting multiple times until some arbitrary condition regarding the output is satisfied (e.g., the ciphertext starts with 42).

Assume the coercer's goal is to make the voter vote for some option $m^\star = m_w$, for some $w \in \{1, 2, \ldots, \ell\}$, likely different to the voting option $m = m_j$ that the voter wants to choose. But in this case, all that the voter (or strictly speaking, the simulator Sim) has to do to deceive the coercer $\mathcal{A}$ is to say it received value $\hat{X}_w$ (instead of $\hat{X}_j$) from $\mathcal{VD}$ in Step 2. In terminology of Definition 2, the simulated private trace $\mathsf{Trc}_{\mathsf{priv}}^{\mathsf{Sim}}$ can be obtained from $\mathsf{Trc}_{\mathsf{priv}}$ by replacing $m = m_j$ with $m^\star = m_w$ and by replacing $\hat{X}_j$ with $\hat{X}_w$. Note that all required information is available to Sim, which has the whole trace Trc and $m^\star = m_w$ as inputs. Moreover, such replacing operations performed by Sim clearly belong to the considered class Cpb, as required.

4 Discussion and Conclusions

4.1 Practical Considerations

We assume that the final output of the protocol, computed by the voting device $\mathcal{VD}$, is published immediately in the official public board of the election, which can be checked by the voter in order to run its individual verification. If this verification fails, the voter should complain and cancel the execution of the voting phase; otherwise, the voter should press a button to confirm the vote. With respect to public verification, some users/devices selected by the election will run it: only those ciphertexts corresponding to executions of the voting phase where the voter has confirmed and where public verification is valid will be moved to the ballot box for the tally phase.

In case a voter cancels the execution, the election should specify if the voter can start the voting phase again, with the same (or another) voting device.

4.2 Comparison with Bingo Voting: On the Necessity of OED

As we have commented in the introduction, our protocol is similar in spirit to Bingo voting [3]: both schemes consider computationally limited voters, and both solutions use a trusted external entity to generate a (pseudo-)random nonce. We think our solution improves Bingo voting in two aspects. First, the pre-voting phase in the Bingo scheme is quite expensive as it requires choseing and committing to many dummy values. In our protocol, there is no pre-voting phase, only the publication of the public parameters of the election. Second, in the Bingo voting, a pseudo-random nonce must be sent to the voter and the voting device $\mathcal{VD}$ through a secure channel because the privacy of the voting phase would be lost if this value was leaked during that execution. In our solution, the random nonce $\mathsf{nc}_{\mathcal{V}} \in \mathcal{X}$ can be (and is) made public by the official election device OED right at the moment of its generation. Therefore, our solution does not need a secure channel between the external trusted entity and the voters.

At this point, one may wonder if the help of an external trusted entity (the official election device OED in our protocol) is necessary. All in all, its only task is to generate and publish a random nonce $\mathsf{nc}_{\mathcal{V}} \in \mathcal{X}$, something that even our limited voter $\mathcal{V}$ could do on its own: if we assume $\mathcal{V}$ can memorize and compare some strings of numbers, then for sure it would be able to generate a random string of numbers, as well.

But if we modify our protocol in such a way that OED disappears and Step 3 is run by the voter $\mathcal{V}$, the result is a protocol that is not coercion-resistant: a coercer who wants $\mathcal{V}$ to vote for an option $m^{\star} = m_w$ can ask the voter $\mathcal{V}$ to use as the nonce $\widehat{\mathsf{nc}}_{\mathcal{V}}$ a value which is computed deterministically from the value $X_w = \hat{H}(\mathsf{C}, \{(A_s, B_s)\}_{1 \leq s \leq \ell}, \{e_i\}_{1 \leq i \leq \ell, i \neq w})$. For instance, to use as $\widehat{\mathsf{nc}}_{\mathcal{V}}$ the first digits of X_w. The voter $\mathcal{V}$ cannot vote for another option $j \neq w$, because in such a case, $\mathcal{V}$ would get to know the value of $\hat{X}_w$ (needed to compute $\widehat{\mathsf{nc}}_{\mathcal{V}}$) only in Step 4 of the protocol, whereas the value of the nonce must be sent by $\mathcal{V}$ in Step 3.

We actually have the intuition (not formally proved) that the participation of an external entity is necessary to achieve both coercion-resistance and cast-as-intended verification in the considered setting with limited voters. The intuitive argument is as follows. At some step of the interaction, say J, between voter $\mathcal{V}$ and voting device $\mathcal{VD}$, voter has to communicate its chosen voting option m_j to $\mathcal{VD}$.

On the one hand, if the voter does not send any random values to $\mathcal{VD}$, or if all the random elements are sent to $\mathcal{VD}$ before the step J, then a malicious $\mathcal{VD}$ can break the cast-as-intended verification property, by running the fist steps $\leq J$ for another voting option $m_i \neq m_j$ and then permuting the roles of indices i and j in the rest of steps of the protocol.

On the other hand, if the voter sends some random value rnd to $\mathcal{VD}$ after the voting device $\mathcal{VD}$ has computed and sent to $\mathcal{V}$ some information info_j which depends on the voting option m_j, a coercer can force the voter to use as rnd a function f of info_j (for instance, the first bits of it). If such coercion can be avoided by simulating execution of the voting phase for another option, but following this pattern $\mathsf{rnd} = f(\mathsf{info}_j)$, it seems (and this is the part that we have not been able to prove formally) that such a simulator breaks the cast-as-intended property.

This intuitive argument supports the claims made by the authors of Bingo voting: "... which strongly suggest that the voter should not be trusted to contribute her own randomness. This gives an additional motivation for the use of trusted random number generator". A formal and complete proof of the impossibility of achieving coercion-resistance and cast-as-intended verification in the limited voters setting remains an interesting open problem.

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

References

1. Adida, B.: Helios: web-based open-audit voting. In: Proceedings of the 17th USENIX Security Symposium, pp. 335–348 (2008)
2. Benaloh, J.: Ballot casting assurance via voter-initiated poll station auditing. In: Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology, EVT'07, p. 14, USA, 2007. USENIX Association (2007)
3. Bohli, J.-M., Müller-Quade, J., Röhrich, S.: Bingo voting: secure and coercion-free voting using a trusted random number generator. In: Alkassar, A., Volkamer, M. (eds.) Vote-ID 2007. LNCS, vol. 4896, pp. 111–124. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77493-8_10
4. Canetti, R., Gennaro, R.: Incoercible multiparty computation. In: 37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14–16 October 1996, pp. 504–513. IEEE Computer Society, November 1996
5. Chaum, D.: Secret-ballot receipts: true voter-verifiable elections. IEEE Secur. Priv. Mag. **2**, 38–47 (2004)
6. Cortier, V., Gaudry, P., Yang, Q.: Is the JCJ voting system really coercion-resistant? Working paper or preprint, April 2022
7. Delaune, S., Kremer, S., Ryan, M.: Coercion-resistance and receipt-freeness in electronic voting. In: 19th IEEE Computer Security Foundations Workshop (CSFW'06), pp. 12–42 (2006)
8. Finogina, T., Herranz, J., Larraia, E.: How (not) to achieve both coercion resistance and cast as intended verifiability in remote evoting. In: Conti, M., Stevens, M., Krenn, S. (eds.) CANS 2021. LNCS, vol. 13099, pp. 483–491. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92548-2_25
9. Gardner, R.W., Garera, S., Rubin, A.D.: Coercion resistant end-to-end voting. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 344–361. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03549-4_21
10. Guasch, S., Morillo, P.: How to challenge *and* cast your e-Vote. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 130–145. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54970-4_8
11. Haines, T., Smyth, B.: Surveying definitions of coercion resistance. Cryptology ePrint Archive, Report 2019/822 (2019). https://ia.cr/2019/822
12. Iovino, V., Rial, A., Rønne, P.B., Ryan, P.Y.A.: Using Selene to verify your vote in JCJ. In: Brenner, M., et al. (eds.) FC 2017. LNCS, vol. 10323, pp. 385–403. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_24
13. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Towards Trustworthy Elections, New Directions in Electronic Voting, pp. 37–63 (2010)
14. Khazaei, S., Wikström, D.: Return code schemes for electronic voting systems. In: Krimmer, R., Volkamer, M., Braun Binder, N., Kersting, N., Pereira, O., Schürmann, C. (eds.) E-Vote-ID 2017. LNCS, vol. 10615, pp. 198–209. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68687-5_12
15. Krips, K., Willemson, J.: On practical aspects of coercion-resistant remote voting systems. In: Krimmer, R., et al. (eds.) E-Vote-ID 2019. LNCS, vol. 11759, pp. 216–232. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30625-0_14
16. Küsters, R., Truderung, T., Vogt, A.: A game-based definition of coercion-resistance and its applications. In: 2010 23rd IEEE Computer Security Foundations Symposium, pp. 122–136 (2010)

17. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_22
18. Neff, C.A.: Practical high certainty intent verification for encrypted votes (2004)
19. Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0028157
20. Pankova, A., Willemson, J.: Relations between privacy, verifiability, accountability and coercion-resistance in voting protocols. Cryptology ePrint Archive, Report 2021/1501 (2021). https://ia.cr/2021/1501
21. Riou, S.; Kulyk, O.; Marcos del Blanco, D.Y.: A formal approach to coercion resistance and its application to e-voting. Mathematics **10**, 781 (2022). https://doi.org/10.3390/math10050781
22. Ryan, P.Y.A., Rønne, P.B., Iovino, V.: Selene: voting with transparent verifiability and coercion-mitigation. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 176–192. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_12
23. Smith, W.D.: New cryptographic election protocol with best-known theoretical properties. In: Workshop on Frontiers in Electronic Election (2005). https://users.encs.concordia.ca/~clark/biblio/coercion/Smith/202005-1.pdf
24. Trechsel, A.H., Vassil, K.: Internet voting in Estonia: a comparative analysis of four elections since 2005: report for the council of Europe (2010)
25. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections. In: STOC '94 Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing, pp. 544–553. Association for Computing Machinery Inc., May 1994