Towards everlasting privacy and efficient coercion resistance in remote electronic voting

Panagiotis Grontas[1], Aris Pagourtzis[1], Alexandros Zacharakis[1], and Bingsheng Zhang[2⋆]

[1] School of Electrical and Computer Engineering
National Technical University of Athens
[2] School of Computing and Communications, Lancaster University
pgrontas@corelab.ntua.gr, pagour@cs.ntua.gr, azach@corelab.ntua.gr,
b.zhang2@lancaster.ac.uk

Abstract. In this work, we propose a first version of an e-voting scheme that achieves end-to-end verifiability, everlasting privacy and efficient coercion resistance in the JCJ setting. Everlasting privacy is achieved assuming an anonymous channel, without resorting to dedicated channels between the election authorities to exchange private data. In addition, the proposed scheme achieves coercion resistance under standard JCJ assumptions. As a core building block of our scheme, we also propose a new primitive called *publicly auditable conditional blind signature* (PACBS), where a client receives a token from the signing server after interaction; the token is a valid signature only if a certain condition holds and the validity of the signature can only be checked by a designated verifier. We utilize this primitive to blindly mark votes under coercion in an auditable manner.

Keywords: electronic voting, end-to-end verifiability, coercion resistance, everlasting privacy, publicly auditable conditional blind signatures

1 Introduction

The cryptographic research on electronic voting spans almost four decades. Despite the proliferation of proposed schemes, few have been implemented and used in actual elections. This can be attributed to the fact, that e-voting systems must reconcile conflicting properties with *integrity* and *privacy* being the most important ones. Integrity is usually achieved through verifiability, which can be individual, universal or administrative, allowing the voter, the public or some trusted authorities, respectively, to check that all participants followed the protocol. Privacy protection comes in many layers: At the most basic level the secrecy of the vote is protected from the talliers. *Everlasting privacy* aims to protect against future and more powerful adversaries modelling the fact that

⋆ This author is supported by EPSRC grants EP/P034578/1 and EP/N023234/1.

theoretical and practical advances (e.g. quantum computing) might render obsolete the cryptographic assumptions upon which privacy rests. *Receipt Freeness* protects from a dishonest voter wanting to sell her vote to a passive adversary. *Coercion Resistance* is the essential property that will enable remote electronic voting, where the lack of a controlled environment for vote casting, leaves the voters vulnerable to active adversaries that can 'look over their shoulder' and dictate their behavior.

Juels, Catalano and Jakobsson proposed in [12] a framework to defend against coercion attacks, which was implemented in Civitas [19]. Their main idea was, that in order to achieve coercion resistance, the aspiring coercer must not be able to tell whether his attempt succeeded or not. This can be done by allowing the voter, to cast many ballots accompanied by anonymous credentials. Specifically, she obtains a valid credential through a one-time use of an untappable channel. Moreover she is assumed to have the capability to generate many different but indistinguishable ones. Under coercion, she uses fake (unregistered) credentials, indistinguishable from the valid one, which is employed when the voter has a moment of privacy, a necessary condition for coercion resistance. To correctly count the votes, however, the system must filter out the ballots that correspond to false credentials. This is done by comparing (in encrypted form) the supplied credentials with the valid ones that are published in a master list - the voter roll - after registration ends. Moreover, since the JCJ scheme allows for multiple votes per voter, duplicate ballots must be removed before counting. In principle, these operations are of quadratic complexity with respect to the number of ballots cast, which is typically quite larger than the number of voters, making the scheme impractical for large scale elections.

The motivation for this work stems from the reasonable assumption, first stated in [15], that the importance of integrity peaks during and immediately after the voting process, but diminishes as more parties are convinced about the result and concede. On the other hand, privacy is important during both voting and counting but retains its importance long after the announcement of the results, since the votes serve as evidence of one's political beliefs. Verifiability implies that such evidence is available to many third parties, making it in effect undeletable. This can have dire consequences in the case of a future oppressive regime. Therefore, our focus is a voting protocol that enhances privacy, both during and after voting without sacrificing verifiability.

Our contribution We propose a first version of a voting protocol based on the architecture of FOO [4], one of the most privacy aware voting schemes in the literature, augmented with an efficient implementetation of the coercion resistance properties of JCJ [12]. In particular, we take advantage of the fact that in FOO, voting occurs in two phases, namely authorization and counting, and use it to overcome the performance bottleneck of JCJ. We achieve this by using the idea of [37], i.e. marking the fake credentials during the authorization phase where voter identification is available. By using the voter ID the correct credential can be efficiently retrieved and compared to the supplied one with no need to check all credentials. Of course, during this phase the ballot contents

must be blinded, as they can be correlated with the voter ID. The fact that the credential is invalid is conveyed to the counting phase by applying a publicly auditable variation of a novel cryptographic primitive, *Conditional Blind Signatures* (CBS) [40]. The counter receives the ballot and an authorization in the form of a blind signature, that contains a bit that specifies if the vote is valid or under coercion. The perfect blindness property of the CBS scheme combined with an anonymous channel enable us to achieve the everlasting privacy property, without residing to dedicated channels between the authorities. Our protocol achieves verifiability, coercion resistance and everlasting privacy with minimal assumptions.

Related work Various efforts in the literature have tried to overcome the performance bottleneck of the JCJ scheme. In [13,17] linear complexity is achieved, by blinding the credentials and then stripping off the encryption randomization. As a result, they could be efficiently compared through a hashtable. Both schemes, however, were later found by [16] not to be coercion resistant by using a classic tagging attack. In the same paper, a new approach improves the quadratic complexity of identifying invalid credentials, by representing them as tuples with an underlying mathematical structure and not as mere random group elements. Hashing could then be used on part of the credential, without affecting the coercion resistance property. This approach was improved in [20,28,33] by utilizing different forms of credential structure and renewal methods, in order to enable use in multiple elections and minimize reliance on the untappable channel. In a different line of work, in [25] it is pointed out that the tagging attack is irrelevant in the duplicate removal subphase. As a result, the blind hashtables can be used there, thus achieving the goal of linearity in the number of votes. For linear fake removal the voter retrieves through the untappable channel, during registration, the index where her real credential is stored in the voter roll. Through this index, coerced credentials are identified. In an alternate approach, [22] and [26] employ the idea of anonymity sets. During vote casting the voter presents the real credential mixed with reencryptions of some other credentials from the voter roll. Finally, [36] with the Selene system and [31] with coercion evidence less strict definitions of coercion resistance are offered which might prove easier to reconcile with the other conflicting properties of voting systems. Our work utilizes ideas from all these works and integrates them in an efficient manner via a new cryptographic primitive, conditional blind signatures.

The term everlasting privacy was introduced by [15] where a protocol that uses perfectly hiding commitments and homomorphic encryption was proposed. Their main idea was that the public votes are protected perfectly using the commitment scheme, while the openings are computationally protected but are exchanged through private channels. As a result they are not publicly available. This idea is presented as a primitive that can be integrated in any homomorphic tallying scheme in [30], while in [29] it is applied to mixnets providing everlasting privacy towards the public. This is further expanded and formalized as practical everlasting privacy in [27] by noting that such a future adversary might be more powerful in terms of computing power, but will have less information to

operate on, since ephemeral data generated by the protocol will be unavailable in the long run. More recently, in [39], the authors implement the commitments with verifiable secret sharing and present a scheme that provides privacy and integrity against unbounded adversaries. However they assume untappable channels between the authorities and deny voters the ability to individually verify their votes. Our scheme differs from this line of work, since we do not assume or use specific channels between the authorities. All information is exchanged through the public Bulletin Board. This mode of communication is more realistic since a future regime with advanced cryptographic capabilities will have access to information exchanged by former governmental agencies in a private manner. Moreover our scheme also provides coercion resistance.

Coercion resistance combined with everlasting privacy seems to be an important desideratum in recent works. However this has not yet been possible in the JCJ framework, which is what our scheme accomplishes. In [38], a version of Selene enhanced for JCJ coercion resistance is equipped with everlasting privacy towards the public with the use of pseudonyms. However the creation process of pseudonyms and their relationship to real voter IDs and credentials requires trust assumptions and private channels between the members of the registration authority. Our work requires only the use of an anonymous channel and provides the same guarantees to both insiders and outsiders. In [35], everlasting privacy is achieved by using perfectly hiding commitments to registered identity credentials along with an anonymous channel. To achieve coercion resistance, votes can be overwritten and only the last one counts. As a result a voter under coercion can save her real vote for the end. This is a much stronger assumption than a simple moment of privacy required by our scheme and the JCJ framework; for example, an adversary who is able to cast a last minute vote achieves coercion.

2 Preliminaries

We begin by describing the agents, the functional components and the cryptographic primitives that make up our scheme.

- The main participants are naturally the *n voters*. In our protocol, like in [22], we assume that there exist pro democratic organizations that cast extra votes for registered voters in an effort to increase the size of the anonymity set.
- The *registration authority RA* registers the identities of the voters and provides credentials. We assume this occurs offline using an untappable channel.
- The *tallying authority TA* authorizes which ballots are accepted for counting by using a blind signature with an implicit validity bit. Later in the tallying phase it counts the valid ballots and announces the result.

In our scheme the registration authority and the tallying authority can be the same physical election authority *EA*. In reality they both consist of many members with conflicting interests for the election outcome. For clarity, however, we shall refer to them henceforth as if they consist only of a single member.

Bulletin Board (BB) A standard component of most electronic voting schemes. It is an authenticated broadcast channel with memory. It is meant to be implemented with Byzantine agreement algorithms. We do not provide implementation details here, following the vast majority of the electronic voting literature. We assume that whenever the voters use the BB, they are doing so through an *anonymous* channel that reveals no information about the identity of the sender of a message and that all the messages (requests, votes, proofs etc.) produced by our protocol can be found on the BB.

Homomorphic Encryption Scheme We assume all cryptographic operations are performed by a JCJ compatible cryptosystem, i.e. one that supports reencryption and verifiable threshold decryption. In order to prove coercion resistance for our scheme we will use the Modified El Gamal (M-El Gamal) cryptosystem as presented in JCJ. It operates in a group $\mathbb{G}$ of prime order q where the DDH Problem is hard. Two generators g_1, g_2 are chosen randomly. The secret key is an element $x \in \mathbb{Z}_q$ and the public key is $h = g_1^x$. Encryption[3] is performed as: $\mathtt{E}_h(m, r) = (g_1^r, g_2^r, mh^r)$ while decryption as: $\mathtt{D}_x(a, b, c) = c \cdot a^{-x}$.

Proofs of Knowledge We make extensive use of non-interactive zero knowledge (NIZK) proofs of knowledge. For instance a (M-El Gamal) ciphertext (a, b, c) is accompanied by proof that a, b have the same secret discrete logarithm relative to g_1, g_2. This can be implemented with the Chaum - Pedersen protocol [6] and made non interactive with the Fiat - Shamir heuristic [2]. We denote this by the functionality NIZK in the following way: $\mathsf{NIZK}\{(g_1, g_2, a, b), (x) : a = g_1^x \wedge b = g_2^x\}$. We also use proofs that a value is a member of a set. We achieve this by using OR compositions of Chaum - Pedersen proofs as described in [7]. We also use proofs of knowledge of a discrete logarithm [3]. Finally designated verifier proofs [8], denoted as DVP, convince the voter but not the coercer that his credential was correctly encrypted, as in JCJ.

Verifiable Shuffles We assume a functionality Shuffle, like the one proposed in [24], that takes as input a list of encrypted values and outputs a random permutation and reencryption of these values along with NIZK proofs that these operations were correctly performed. We use shuffles in tallying as in JCJ.

Blind Signatures They allow a signer to sign messages without having access to their contents [1]. To this end the user *blinds* the message and the signer signs it in this blinded form. The user subsequently *unblinds* the signature, and retrieves a valid signature for the plain message. Their security properties [23,9] are *blindness* or *unlinkability* which states that the signer cannot retrieve the signed message or associate signatures with protocol executions. *Unforgeability* states that the user cannot generate more message-signature pairs than those obtained by the signer. In the proposed protocol we use a variation of blind

[3] For compactness we omit the encryption randomness, except when it is absolutely necessary for the operation of our scheme. We also use the plain ElGamal to describe the protocol and refer to M-El Gamal only in the coercion resistance analysis.

signatures, Conditional Blind Signatures, to enable everlasting privacy and move the marking of coerced votes from the tallying to the authorization phase.

Plaintext Equivalence Test The functionality PET is a primitive introduced in [11] to convince a distributed set of entities, who share a decryption key that two ciphertexts indeed encrypt the same plaintext. It works by first having the participants blind the ciphertexts and then employing the homomorphic properties of the underlying cryptosystem to compute a function on them, such that a joint decryption of the result indicates if the two initial ciphertexts encrypt the same message or not. We use PET to mark duplicate votes and also embed them in the signatures that mark coerced votes in the authorisation phase.

3 Publicly Auditable Conditional Blind Signatures

Our voting scheme is built on a variation of Conditional Blind Signatures (CBS) [40]. This primitive allows a signer $\mathcal{S}$ to blindly generate signatures on messages submitted by the user $\mathcal{U}$. These signatures however are verifiable only by a designated verifier $\mathcal{V}$, like in [8]. Furthermore, their validity depends on a secret information bit 'injected' into the signature along with the possession of a secret key by the designated verifier. In this way the signer can 'instruct' the verifier to accept the signature or not. The secret bit cannot be learned by the user however, since both cases are indistinguishable to her. Note that the roles of $\mathcal{S}$ and $\mathcal{V}$ can be played by the same entity, thus allowing the signer to send information regarding attributes of blinded messages to herself in the future.

The security of CBS extends the standard security properties of blind signatures such as blindness and protection against One More Forgery to account for the secret bit b. Additionally, to formally express the idea that b controls the validity of the signature an extra property, *Conditional Verifiability*, is defined in [40]. We must observe here that the user cannot validate the signature she receives, since she does not have knowledge of b. Although this seems counter-intuitive with respect to traditional signatures, in our setting it is essentially the exact property we need to achieve coercion resistance.

In [40] an instantiation of CBS is given by extending the well known three-round Okamoto-Schnorr blind signatures [5] (appendix B). This instantiation is proved to have perfect blindness, computational resistance to *Strong* One More Forgery under the Computational Diffie Hellman assumption and Conditional Verifiability under the Decisional Diffie Hellman assumption.

In practice, the scheme's round complexity can be reduced by randomly generating the initial commitment in a preagreed manner. Moreover it can also be combined with a multiplicatively homomorphic encryption scheme as the one we assume in our voting protocol. We present this modified version in appendix C, where the signer and verifier are the same entity.

Public auditability The purpose of the CBS in the proposed voting scheme is to "mark" a ballot as valid if it is accompanied by an encryption of the same credential σ that was generated during registration. If any other credential σ' is

used, the ballot is marked as invalid, indicating that the voter is under coercion. The CBS scheme can be used to convey this bit of information to the verifier, but by design it hides it from the user. As a result we cannot apply it as-is, since this will lead to loss of verifiability. We overcome this by introducing *Publicly Auditable* CBS, which adds auditability using NIZK proofs of correctness during signing and verifiaction. In particular the CBS conditional bit is implicitly computed and embedded in the signature by applying the PET functionality on the registered and voting credentials.

The PACBS scheme operates in a group $\mathbb{G}$ of prime order q, where the DDH is hard. During the parameter generation phase random group elements (g_1, g_2, v, h_1) are selected. These elements are the public parameters of the protocol and are denoted as $\texttt{params}_{\text{CBS}}$. A signing key $s \in \mathbb{Z}_q$ and an encryption key $z \in \mathbb{Z}_q$ are also selected. These secret keys are collectively denoted as $\texttt{sk}_{\text{CBS}}$. The corresponding public keys are $k := g_1^s$ and $h := h_1^z$, denoted as $\texttt{pk}_{\text{CBS}}$.

The PACBS signing protocol in Figure 1 assumes two random oracles $\mathcal{H}_1, \mathcal{H}_2$.

Common input: $\texttt{params}_{\text{CBS}}$, $\texttt{pk}_{\text{CBS}}$, $C_1, C_2 \in \mathbb{G}^2$, $\mathcal{H}_1 : \mathbb{G}^4 \to \mathbb{G}$, $\mathcal{H}_2 : \mathcal{M} \times \mathbb{G} \to \mathbb{Z}_q$
$\mathcal{U}$'s private input: $m \in \mathcal{M}$
$\mathcal{S}$'s private input: $s \in \mathbb{Z}_q$ s.t. $k = g_1^s$

$\mathcal{U}$ executes the Blind Algorithm:

- Compute $x := \mathcal{H}_1(C_1, C_2)$;
- Pick random $u_1, u_2, d \leftarrow_R \mathbb{Z}_q$ and compute $x^* := x g_1^{u_1} g_2^{u_2} v^d$, $e^* := \mathcal{H}_2(m, x^*)$ and $e := e^* - d$;
- Send e to $\mathcal{S}$.

$\mathcal{S}$ executes the Sign Algorithm:

- Compute $x := \mathcal{H}_1(C_1, C_2)$;
- Pick random $y_2 \leftarrow_R \mathbb{Z}_q$ as the second part of the CBS;
- Compute $\nu := x g_2^{-y_2} v^{-e}$;
- Pick random $t \in \mathbb{Z}_q$ and compute $N := \mathtt{E}_h(\nu; t)$;
- Pick random blinding factor $\alpha \in \mathbb{Z}_q$ and compute $W := (C_2/C_1)^\alpha$ and apply signing key to compute $B := (N \cdot W)^s$ with:

$$\pi_1 \leftarrow \mathsf{NIZK}\{(h_1, h, \nu, N), (t) : N = \mathtt{E}_h(\nu; t)\}$$
$$\pi_2 \leftarrow \mathsf{NIZK}\{(C_1, C_2, W), (\alpha) : W = (C_2/C_1)^\alpha\}$$
$$\pi_3 \leftarrow \mathsf{NIZK}\{(h, k, N, W, B), (s) : B = (N \cdot W)^s \ \wedge \ k = g^s\}$$

- Set $\texttt{bsig} := (B, N, W, y_2, \pi_1, \pi_2, \pi_3)$ and send $\texttt{bsig}$ to the $\mathcal{U}$.

$\mathcal{U}$ executes the Unblind Algorithm:

- Verify π_1, π_2, π_3;
- Unblind by computing $sig_1 := B \cdot \mathtt{E}_h(k^{u_1})$ and $sig_2 := y_2 + u_2$.
- Set $\texttt{sig} := (x^*, e^*, sig_1, sig_2)$ and output $(m, \texttt{sig})$.

Fig. 1. The Publicly Auditable CBS Sign protocol PACBS Sign

The signer obtains after registration an encryption of the valid credential $C_1 := \mathtt{E}(\sigma)$. During voting the voter provides an encryption of the voting credential $C_2 := \mathtt{E}(\sigma')$ along with a blinded version e of the message being signed (the vote). The value (C_2/C_1) is blinded with a random $\alpha \in \mathbb{Z}_q$ and multiplied with the signature. The essence of this procedure is that (C_2/C_1) is an encryption of a random element unless $\sigma = \sigma'$ in which case it is an encryption of 1. In the former case the random element will 'corrupt' the signature. In the latter case the signature will be valid, since it is homomorphically multiplied by 1. Every interested entity can verify that the signer did not deviate from the protocol by checking the transcript and the proofs. Thus, an honest voter who knows the input and in particular whether C_1, C_2 are encryptions of the same plaintext knows that his output corresponds to a valid signature.

The PACBS verification algorithm is given in Figure 2. The verifier $\mathcal{V}$, given a message a signature and key pair, outputs whether the signature is valid or not in a way that every other entity can be convinced about it. In reality, $\mathcal{V}$ embeds the PET functionality inside the signature verification equation, which will again hold only if the credentials supplied are the same.

Public input: $\mathtt{params}_{\mathrm{CBS}}$, $\mathtt{pk}_{\mathrm{CBS}}$, $m \in \mathcal{M}, \mathtt{sig}_{\mathrm{CBS}}$ and $\mathcal{H}_2 : \mathcal{M} \times \mathbb{G} \to \mathbb{Z}_q$
Signer's Private input: $\mathtt{sk}_{\mathrm{CBS}}$

- If $\mathcal{H}_2(m, x^*) \neq e^*$ then $\mathcal{S}$ sends $\perp$.
- Otherwise $\mathcal{S}$ picks a random $\beta \in \mathbb{Z}_q$ and computes

$$\text{validity} := x^* \cdot g_2^{-sig_2} \cdot v^{-e^*}; \quad M := \mathtt{E}_h(\text{validity}; r_1); \quad V := M^s;$$

$$R := \Big(\frac{V}{sig_1}\Big)^{\beta} \quad \text{and} \quad \text{result} := \mathtt{D}_z(R)$$

$$\pi_1 \leftarrow \mathsf{NIZK}\big\{(h_1, h, \text{validity}), (r_1) : M = \mathtt{E}_h(\text{validity}; r_1)\big\}$$

$$\pi_2 \leftarrow \mathsf{NIZK}\big\{(k, g_1, V, M), (s) : V = M^s\big\}$$

$$\pi_3 \leftarrow \mathsf{NIZK}\Big\{(V, sig_1, R), (\beta) : R = \Big(\frac{V}{sig_1}\Big)^{\beta}\Big\}$$

$$\pi_4 \leftarrow \mathsf{NIZK}\big\{(h_1, h, \text{result}, R), (z) : \text{result} = \mathtt{D}_z(R)\big\}$$

- $\mathcal{S}$ sends $M, V, R, \text{result}, \pi_1, \pi_2, \pi_3, \pi_4$ to $\mathcal{V}$
- If a proof is not correct $\mathcal{V}$ outputs $\perp$.
- Otherwise $\mathcal{V}$ outputs 1 (valid) iff result = 1.

Fig. 2. The Publicly Auditable CBS Verify protocol PACBS Verify

4 The voting protocol

Our scheme builds on the variation of FOO presented in [10] that is based on public key encryption instead of commitments, thus reducing the number of communication rounds. We use an extra authorization phase, where the issued credentials are secretly marked as valid or invalid using PACBS. Our main idea is that the validity checks for the credentials are (implicitly) done during vote authorization and not during tallying. The protocol has a one-time *Registration*

phase and in each election there are three phases, namely *Authorization*, *Voting* and *Tallying*. To achieve the security properties we take advantage of the separation between Authorization and Voting phases. We stress that each phase starts only after the previous one has ended. A simplified view of the workflow of our protocol is depicted in Figure 3. In the Authorization phase the *EA* checks for

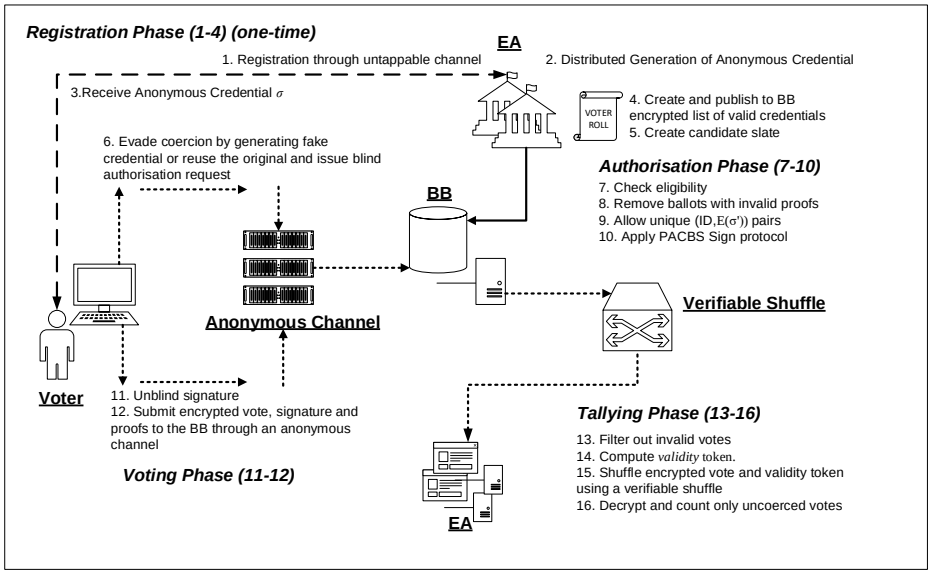

Fig. 3. Framework architecture and workflow

the validity of the supplied voter credentials. This can be done in constant time as the voter identity is known (but not the vote) and the *EA* has access to the voter roll. As a result, it compares the voter supplied credential $\mathtt{E}_h(\sigma')$ with the voter roll version $\mathtt{E}_h(\sigma)$ and finds out if the voter is under coercion. Since the identity of the voter is known at this point, the *EA* can use it to check eligibility by inspecting if there is a corresponding credential in the voter roll. Moreover it can be used to group all the ID and credential pairs so that only one is kept, according to some predefined rule (e.g. last credential counts). Finally the voter and the *EA* interact according to the PACBS Sign protocol and obtain a validity token on the blinded ballot. While the honest voter knows that $\sigma = \sigma'$ and can be sure that the signature will be valid, a coercer without this information cannot know if the ballot will be counted.

In the Voting phase, the voter casts the (unblinded) signature and the ballot to the BB. In the Tallying phase the *EA* must act as the verifier in PACBS and counts the votes only if the signature is valid. However, the ballot and result pairs must also be shuffled, so that the coercer loses track. Only then can they be decrypted to yield the final tally. This cannot be achieved directly by the PACBS Verify algorithm in Figure 2 because shuffling must occur before de-

cryption. Consequently the *EA*, actually uses a slight variation of PACBS Verify, which does not decrypt the final result nor create the proof of correct decryption, since these take place "outside" of the algorithm. We denote this alternate verification procedure EncVerify. In this stage neither the ID nor any credential information is present so the ballot cannot be linked to a voter.

A detailed description is given in Figure 4. Correctness follows by inspecting Figure 1, Figure 2 and Figure 4. We assume that honest voters intentionally issue invalid votes during authorization to thwart forced abstention attacks.

Distributed EA The *EA* can be modeled as a set of mutually distrustful parties executing secure protocols. In particular, the parameters for the protocol can be securely generated using standard techniques. The keys can be computed using a verifiable secret sharing scheme. The credentials can be generated as in [19]. Apart from the PACBS Sign and PACBS Verify all other actions performed by the *EA* (Decryption, Shuffle) are also standard and the checking for doubly issued credentials can be performed using PETs. The PACBS Sign and PACBS Verify can easily be extended by essentially performing the same protocols with each key share and combining the results.

Performance Our analysis closely resembles [21]. Excluding the elimination of double votes all computations are linear in the number of votes. If $|ID_i|$ denotes the number of votes cast with ID_i and $m = \max_i |ID_i|$ then the number of computations is $\mathcal{O}(m^2 n)$. This can be further reduced to $\mathcal{O}(mn)$ using a method like the blind hashtables of [17], since the tagging attack is not applicable in this phase. In any case, assuming that the number of duplicates per voter will be constant in practice - i.e. $m = \mathcal{O}(1)$ - then the number of computations becomes linear in the number of voters n.

5 Security Analysis

Threat model Since our work is an extension of [12], our assumptions follow theirs closely. Firstly we require trusted implementation in software and hardware. While the amount of trust required can be decreased by using techniques such as Benaloh challenges and code-voting as in [18,36,32], it cannot be completely disregarded. This is easier said than done, but it is a common practice in the vast majority of proposed voting protocols at our level of abstraction.

We assume two types of adversaries, one computationally bounded that acts during or shortly after the election and one that is computationally unbounded and acts in the future. The former models the security requirements which are vital during the election such as integrity, verifiability and coercion resistance while the latter models our requirement for everlasting privacy.

As far as present adversaries are concerned, we assume that they can perform only probabilistic polynomial time computations and for which our computational assumptions hold. To prove verifiability we assume that the adversary fully controls the election authorities and corrupts voters of his choice [32].

As far as coercion resistance is concerned, the adversary can fully control a subset of the voters by impersonating them, but there exists another subset with

Common input: $\texttt{params}_{\text{CBS}}$
***EA*'s private input:** $\texttt{sk}_{\text{CBS}}$
***EA*'s public keys:** $\texttt{pk}_{\text{CBS}}$
Voter's private input: $v \in \mathbf{C}$

One Time Registration Phase:

- The *EA* generates the voter credential $\sigma \leftarrow_R \mathbb{G}$ and computes: $C_1 := \texttt{E}_h(\sigma)$ along with $\delta \leftarrow \mathsf{DVP}\{C_1 = \texttt{E}_h(\sigma)\}$.
- The *EA* publishes the encrypted credential $\mathsf{BB} \Leftarrow (\text{ID}, C_1)$ and sends σ, δ to the voter using an untappable channel.

Election Setup Phase:

- The *EA* publishes the candidate slate $\mathbf{C} \subset \mathbb{G}$ by assigning a random group element to each candidate and a list of IDs denoted $\mathcal{I}$ corresponding to the voters with a right to vote in the election.

Authorization Phase:

- The voter computes a new credential $\sigma' \leftarrow_R \mathbb{G}$ and $C_2 := \texttt{E}_h(\sigma'; r_1)$ along with:

$$\pi_1 \leftarrow \mathsf{NIZK}\{(g_1, h, C_2), (\sigma', r_1) : C_2 = \texttt{E}_h(\sigma'; r_1)\}$$

- The voter encrypts his vote as $C := \texttt{E}_h(v; r_v)$ along with:

$$\pi_2 \leftarrow \mathsf{NIZK}\{(g_1, h, \mathbf{C}, C), (v, r_v) : v \in \mathbf{C} \ \wedge \ C = \texttt{E}_h(v; r_v)\}$$

- The voter invokes Blind (Figure 1) for $m := C$ to get:

$$e := \mathsf{Blind}(\texttt{params}_{\text{CBS}}, \texttt{pk}_{\text{CBS}}, C, C_1, C_2)$$

- The voter posts (ID, C_2, e, π_1) to the BB.
- The *EA* checks the validity of the proof and that $\text{ID} \in \mathcal{I}$.
- The *EA* checks that no other request C_2' with the same supplied credential σ' is submitted for ID. If some condition fails the request is ignored as double and the EA publishes

$$\pi_{C_2} \leftarrow \mathsf{NIZK}\{(h_1, h, C_2, C_2')(z) : \texttt{D}_z(C_2/C_2') = 1\}$$

- Otherwise the *EA* publishes $\mathsf{BB} \Leftarrow \texttt{bsig}$ where:

$$\texttt{bsig} := \mathsf{Sign}_{\mathsf{CBS}}(\texttt{params}_{\text{CBS}}, \texttt{sk}_{\text{CBS}}, e, C_1, C_2)$$

- The voter computes:

$$\texttt{sig} := \mathsf{Unblind}(\texttt{params}_{\text{CBS}}, \texttt{pk}_{\text{CBS}}, \texttt{bsig})$$

Voting Phase:

- The voter appends to the BB the vote tuple

$$\mathsf{BB} \Leftarrow (C, \texttt{sig}, \pi_2)$$

Tallying Phase:

- To prevent double casting if more than one lines with correct proofs contains C the *EA* keeps only the last submitted.
- For each submitted ballot with valid proofs the *EA* calls

$$\mathsf{EncVerify}_{\mathsf{CBS}}(\texttt{params}_{\text{CBS}}, \texttt{sk}_{\text{CBS}}, C, \texttt{sig})$$

  and publishes the result tuples $\texttt{R} = (M, V, R, \pi_1, \pi_2, \pi_3)$ from Figure 2.
- The *EA* appends $L \Leftarrow (C, \texttt{R})$ where L is a designated section of the BB
- Then it executes $L' := \{(C', \texttt{R}')\} := \mathsf{Shuffle}(L)$.
- The *EA* verifiably decrypts all pairs. A vote is counted iff $\texttt{D}_z(R') = 1$

Fig. 4. The voting protocol

uncertain behavior. As in [12] each uncontrolled voter has a moment of privacy. The adversary can corrupt a subset of the voting authorities, which consist of mutually distrusting agents. Moreover he is capable of controlling the Bulletin Board and all other public channels, but there exist anonymous channels, where the identity of the sender of a message cannot be discovered. Finally there are honest participants, maybe nonprofit organizations, that cast invalid votes with valid voter IDs in order to thwart a forced abstention attack.

The future adversary is computationally unbounded and so can break any cryptographic assumption. Her goal is to gain information about the votes of a subset of the voters. We make the assumption that she can gain no information about the identity of the voter by the anonymous channel.

Verifiability We follow the end-to-end verifiability definition proposed by Kiayias *et al.* [32], which can be viewed as a computational variant of the KTV framework as summarized in [34]. The adversarial goal against system's integrity is to cause deviation from the intended tally of all the honest voters while election auditing remains successful without complains. We consider an adversary that controls the *EA* and a subset of the voters. All the voters who did not participate in the election are considered to be compromised. This is because a malicious (registration) authority can always impersonate absent voters without the PKI assumption.

Our scheme achieves end-to-end verifiability against a fully corrupted *EA* under the random oracle model. As a standard requirement, we assume the existence of a trusted BB. Although the voters' clients are assumed to be honest for current protocol description, it is easy to add the Benaloh challenge mechanism [14] to prevent the malicious clients from tampering the ballot as the patch for Helios [18]. The voter needs to verify that her submitted ballot was recorded correctly on the BB and was taken as an input of the shuffle/mix-net.

During registration (Figure 3 - step 2), the consistency between the voter's credential σ and the published $\mathtt{E}_h(\sigma)$ is guaranteed by the DVP, which is intentionally not universally verifiable to enable coercion resistance. In the authorization phase, the signatures for the validity of the credential (Figure 3 - step 10) are verifiable due to the design of PACBS Sign and the proof published for each unprocessed request. More specifically, *EA* shows that the produced signature is valid if and only if the submitted credential matches the recorded ones. In the tallying phase, the public auditability property of the PACBS Verify protocol, the verifiable shuffle and the proof of correct decryption prevent the authority from deviating in any way from the protocol specification. Finally, everyone can check if the total number of valid signatures are less than or equal to the number of voters, n. This would prevent the malicious *EA* from inserting additional valid signatures. Since the honest voters' signatures are all cast, recorded, and tallied correctly, the rest valid signatures can be viewed as the adversarial ones. Hence, the malicious *EA* cannot add more votes even if she has the signing key.

Eligibility The eligibility property is based on the resistance to Strong One More Forgery property of the signature scheme, since a valid signature is required

for the vote to be counted. This implies the strong assumption that the adversary is restricted to a polylogarithmic number of honest authorizations.

Privacy Our protocol satisfies vote privacy. In the authorization phase the encrypted vote is blinded when posted to the BB. As a result there is no way to recover the selection of the voter even if the *EA* is fully corrupted. In the voting phase the privacy of the vote depends on the privacy of the actions performed by the *EA* (Decryption, Shuffle). Without the assumption of an anonymous channel our system offers Helios [18] level privacy under similar trust assumptions. However, assuming an anonymous channel privacy protection becomes complete.

Everlasting privacy Our scheme easily meets the requirements for practical everlasting privacy set in [27], despite the fact that there are no private channels between the participants. A future adversary with access to the data in the BB, but without access to the untappable channels and to network related information will not be able to associate authorization requests and votes because of the blindness of the signatures. Moreover the tallying phase where there are neither voter identities nor credentials present, matches the Helios without identities case of [27] which is proved to satisfy practical everlasting privacy. However if we assume an anonymous channel as in [4,35] our scheme has complete everlasting privacy. In the authorization phase, the perfect blindness of the signature scheme ensures that no information regarding the vote is leaked. Furthermore, when the ballots are posted in the BB, despite being only computationally protected, they are cast through an anonymous channel and contain no information about the identity of the voter. Moreover, the encrypted vote and signatures cannot be associated with any particular execution of the signing protocol that validated it. As a result a semi-honest unbounded adversary, watching all the public interactions cannot associate any voter with his vote.

Coercion Resistance Our scheme is Coercion Resistant. In particular if a coercer requests a credential σ from a voter, its validity cannot be proved. As a result the validity of the signature issued for this credential is unknown to the coercer, due to the properties of PACBS. Moreover multiple votes cast with the same ID in the authorization phase, protect from a forced abstention attack. The reasoning is similar to [22,26]. In the tallying phase, shuffling and PACBS Verify ensure the coercer loses track of his submitted vote and the only information he gets is the final tally. A detailed analysis is given in appendix A.

6 Conclusion

In this paper we presented a new approach to provide coercion resistance in an efficient manner and combine it with everlasting privacy. Our protocol is based on minimal assumptions: a single use of an untappable channel and the existence of an anonymous channel. We utilized Conditional Blind Signatures [40], a recent primitive that allows a signer to inject a bit of secret information to a blind signature that controls if it should validate or not, which we improved for our purposes. Our scheme is proved secure under the JCJ [12] coercion resistance

framework. The perfect blindness provided by CBS allows for stronger privacy guarantees; combined with a perfectly anonymous channel it provides the everlasting privacy property. In a future version of this work we plan to augment the intuitive security analysis presented here, using rigorous definitions and proofs.

Acknowledgements The authors would like to thank Peter Browne Roenne and the anonymous reviewers for their helpful comments and suggestions.

References

1. David Chaum. Blind signatures for untraceable payments. In D. Chaum, R.L. Rivest, and A.T. Sherman, editors, *CRYPTO '82*, pages 199–203, 1983.
2. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO' 86*, pages 186–194, 1986.
3. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO '89*, pages 239–252, 1989.
4. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *ASIACRYPT '92*, pages 244–251, 1992.
5. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO '92*, pages 31–53, 1992.
6. David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO '92*, pages 89–105. Springer-Verlag, 1993.
7. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, pages 174–187, 1994.
8. Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT '96*, LNCS, pages 143–154, 1996.
9. Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In *CRYPTO '97*, pages 150–164, 1997.
10. Miyako Ohkubo, Fumiaki Miura, Masayuki Abe, Atsushi Fujioka, and Tatsuaki Okamoto. An improvement on a practical secret voting scheme. In *Information Security*, LNCS, pages 225–234. 1999.
11. Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT '00*, pages 162–177. Springer-Verlag, 2000.
12. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 61–70. ACM, 2005.
13. Warren D. Smith. New cryptographic voting scheme with best-known theoretical properties. In *Frontiers in Electronic Elections (FEE 2005)*, June 2005.
14. Josh Benaloh. Simple verifiable elections. In *EVT'06*, 2006.
15. Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *CRYPTO'06*, pages 373–392, 2006.
16. Roberto Araújo, Sébastien Foulle, and Jacques Traoré. A practical and secure coercion resistant scheme for remote elections. In *Frontiers of Electronic Voting*, 2007.
17. Stefan G. Weber, Roberto Araujo, and Johannes Buchmann. On coercion-resistant electronic elections with linear work. In *ARES*, pages 908–916. IEEE, 2007.
18. Ben Adida. Helios: web-based open-audit voting. In *Proceedings of the 17th conference on Security symposium*, pages 335–348. USENIX Association, 2008.

19. Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *IEEE Security and Privacy Symposium*, 2008.
20. Roberto Araújo, Narjes Ben Rajeb, Riadh Robbana, Jacques Traoré, and Souheib Yousfi. Towards practical and secure coercion-resistant electronic elections. In *CANS*, pages 278–297, 2010.
21. Reto E. Koenig, Rolf Haenni, and Stephan Fischli. Preventing board flooding attacks in coercion-resistant electronic voting schemes. In *SEC*, 2011.
22. Michael Schlapfer, Rolf Haenni, Reto E. Koenig, and Oliver Spycher. Efficient vote authorization in coercion-resistant internet voting. In *VOTE-ID*, 2011.
23. Dominique Schröder and Dominique Unruh. Security of blind signatures revisited. *IACR Cryptology ePrint Archive*, page 316, 2011.
24. Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *EUROCRYPT 2012*, pages 263–280, 2012.
25. Oliver Spycher, Reto Koenig, Rolf Haenni, and Michael Schlapfer. A new approach towards coercion-resistant remote e-voting in linear time. In *FC 2011*, 2012.
26. Jeremy Clark Urs and Hengartner. Selections: Internet voting with over-the-shoulder coercion-resistance. In *FC 2011*, 2012.
27. Myrto Arapinis, Véronique Cortier, Steve Kremer, and Mark Ryan. Practical everlasting privacy. In *Principles of Security and Trust, POST 2013*, 2013.
28. Roberto Araújo and Jacques Traoré. A practical coercion resistant voting scheme revisited. In *VOTE-ID*, pages 193–209, 2013.
29. Johannes A. Buchmann, Denise Demirel, and Jeroen van de Graaf. Towards a publicly-verifiable mix-net providing everlasting privacy. In *FC 2013*, 2013.
30. Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In *ESORICS 2013*, pages 481–498, 2013.
31. Gurchetan S Grewal, Mark D Ryan, Sergiu Bursuc, and Peter Y.A. Ryan. Caveat coercitor: Coercion-evidence in electronic voting. In *IEEE Security and Privacy Symposium*. IEEE, 2013.
32. Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In *EUROCRYPT 2015*, pages 468–498, 2015.
33. Roberto Araújo, Amira Barki, Solenn Brunet, and Jacques Traoré. Remote electronic voting can be efficient, verifiable and coercion-resistant. In *FC'16 Workshops, BITCOIN, VOTING, WAHC*, 2016.
34. V. Cortier, D. Galindo, R. Ksters, J. Mller, and T. Truderung. Sok: Verifiability notions for e-voting protocols. In *IEEE Security and Privacy Symposium*, pages 779–798, 2016.
35. Philipp Locher, Rolf Haenni, and Reto E. Koenig. Coercion-resistant internet voting with everlasting privacy. In *FC'16 Workshops, BITCOIN,VOTING,WAHC*, 2016.
36. Peter Y.A.Ryan, Peter B. Roenne, and Vincenzo Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. In *FC'16 Workshops, BITCOIN, VOTING, WAHC*, 2016.
37. Panagiotis Grontas, Aris Pagourtzis, and Alexandros Zacharakis. Coercion resistance in a practical secret voting scheme for large scale elections. In *ISPAN-FCST-ISCC 2017*, pages 514–519, 2017.
38. Vincenzo Iovino, Alfredo Rial, Peter B Roenne, and Peter Ryan. Using selene to verify your vote in jcj. In *FC'17 Workshops, BITCOIN, VOTING, WAHC*, 2017.
39. Nan Yang and Jeremy Clark. Practical governmental voting with unconditional integrity and privacy. In *FC'17 Workshops, BITCOIN, VOTING, WAHC*, 2017.
40. Alexandros Zacharakis, Panagiotis Grontas, and Aris Pagourtzis. Conditional blind signatures. In *7th International Conference on Algebraic Informatics (short version)*, 2017. Full version available on: `http://eprint.iacr.org/2017/682`.

A Analysis of coercion resistance

We prove the coercion resistance property of the proposed voting scheme by closely following the JCJ techniques. We slightly modify the games c-resist and c-resist-ideal of JCJ to account for the extra authorization phase. We treat the auth functionality as a function that provides a valid or invalid ballot in relation to its private input.

Firstly, we examine the options of a coerced voter. Such a voter, can simply supply the adversary with a fake random element of $\mathbb{G}$. Having a fake credential, the signature she receives will be invalid, a fact undetectable by $\mathcal{A}$ due to the design of the CBS protocol. The voter can authenticate and cast her real vote during her moment of privacy. The coercer cannot decrypt the corresponding entry in the voter roll and find out if he was presented with the real one or not. If he tries to vote, when he receives the signature he will not be able to validate it himself. In the tallying phase the shuffle will make him lose track of the vote.

The games c-resist and c-resist-ideal. The only change in the c-resist game of JCJ is the extra authorization phase. During this phase voters supply the essential information, based on their secret credential, and they get a ballot which can be either valid or invalid. We assume that all messages are exchanged via the bulletin board and thus are available to the adversary.

In the c-resist-ideal game an extra ideal functionality is needed, which we call idauth and its purpose is to authenticate and cast the ballots to the BB. This functionality gets the inputs of the players and computes the ballots as follows:

- It gives only one valid ballot for each valid credential.
- For honest voters it extracts the underlying credential and responds with a valid/invalid ballot based on it.
- Adversarial requests with credentials of corrupted players are answered normally. For the credential of the coerced voter the validity of the ballot computed is determined by the coin toss.
- The outputs are written directly to the BB.

In both games, λ is the security parameter, n is the number of voters and n_V is the number of voters the adversary can corrupt. Further inputs include the candidate slate $\mathbf{C}$ and the distribution D that governs the behaviour of the uncontrolled voters. In c-resist-game, the adversary corrupts a subset V of the voters and obtains their credentials after registration. We denote by U the rest of the voters. They perform the registration process and receive their credential. Subsequently the coercer chooses the voter j to coerce and her uncoerced vote β. Of course the voter should not be already corrupted and the vote should be a valid choice. A coin is flipped and if it is 0, the coerced voter fools $\mathcal{A}$ by invoking the functionality fakekey and generating a fake credential while voting with her real one. If it is 1 she obeys and gives her real credential sk_j to $\mathcal{A}$. Honest voters perform the authorization phase and cast their votes, while $\mathcal{A}$ invokes the auth functionality with any input of his choice using the information gathered so far.

The tallying phase is then performed and $\mathcal{A}$ tries to guess the result of the coin, using the data on the BB, the final tally $\mathbf{X}$ and the proofs provided. In c-resist-ideal-game the same things happen with minor differences. The keys obtained by $\mathcal{A}$ do not assist him with the choice of the voter to coerce. He is always given the real credential and auth is replaced with idauth. Finally, the only data $\mathcal{A}$ can use to guess the coin flip is the final tally $\mathbf{X}$ and the number of invalid votes Γ.

We now present the simulation for the proof that our scheme is coercion resistant.

1. **Input:** The simulator $\mathcal{S}$ takes as input the elements g_1, g_2, h_1, h_2 of a group $\mathbb{G}$ of order q and a vector w from a distribution D, which mirrors $\mathcal{A}$'s uncertainty. Each element of w is a set of valid and invalid votes, taking into account that each voter casts more than one ballot. $\mathcal{S}$ tries to answer whether (g_1, g_2, h_1, h_2) is a DH quadruple or not.
2. **Parameter generation:** Initially the $\mathcal{S}$ creates the M-El Gamal encryption key by randomly choosing $x_1, x_2 \in \mathbb{Z}_q$ and computing $h = g_1^{x_1} g_2^{x_2}$. The public key is (g_1, g_2, h). He then creates a signing key pair for the CBS scheme by choosing $g_3, g_4, y \leftarrow_R \mathbb{G}$, $s \in \mathbb{Z}_q$ and $k = g_3^s$. The secret key is s and the public key is (g_3, g_4, y, k).
3. **Registration:** Each voter is assigned a random $\sigma_i \leftarrow_R \mathbb{G}$. Using the public key, $\mathcal{S}$ publishes the voter roll. Finally, the candidate slate $\mathbf{C}$ is published.
4. **Corruption:** $\mathcal{A}$ corrupts voters.
5. **Coercion:** $\mathcal{A}$ chooses the player to coerce and her honest vote (j, β). The appropriate tests are performed in (j, β) according to the games' definitions.
6. **Coin Flip:** $\mathcal{S}$ chooses $b \leftarrow_R \{0, 1\}$. If $b = 0$, $\mathcal{A}$ is given a random group element $\sigma^* \leftarrow_R \mathbb{G}$, else she is given the real credential $\sigma^* \leftarrow \sigma_j$
7. **Authorization Requests:** $\mathcal{S}$ issues the signature requests for the honest voters according to w. For each element of w she issues $(\mathtt{E}_h(\sigma_i), ID_i, \mathrm{PoK}_1)$ where $\mathtt{E}_h(\sigma_i) = (h_1^{u_i}, h_2^{u_i}, h_1^{u_i x_1} h_2^{u_i x_2} \sigma_i)$ for random u_i and the proof PoK_1 is simulated by the programmability of the random oracle by using standard techniques. $\mathcal{A}$ issues his authorization requests.
8. **Double requests elimination:** Using the secret key x_1, x_2, $\mathcal{S}$ decrypts and eliminates double requests with the same credential.
9. **Authorization:** $\mathcal{S}$ simulates this phase using her CBS signing key. The messages are encrypted votes according to w. Encryptions are done in the same way as before. $\mathcal{A}$ is given signatures in a straightforward manner.
10. **Vote Casting:** $\mathcal{S}$ submits ballots for the honest voters. $\mathcal{A}$ submits ballots for the corrupt and the coerced voters.
11. **Tallying:** Using his secret keys and standard techniques for proofs, $\mathcal{S}$ simulates tallying in a straightforward manner.
12. **Guess:** $\mathcal{A}$ decides b'
13. **Output:** $\mathcal{S}$ outputs 1 iff $b = b'$.

Let's examine the view of $\mathcal{A}$. Apart from the data he produces, in the authorization phase he sees the encrypted credentials with the proofs that accompany them, and the signatures given. These include a message x uniformly distributed

in $\mathbb{G}$, an encrypted first part of a signature and the second part of the signature which is a uniformly distributed element in $\mathbb{Z}_q$. In the tallying phase he sees the encrypted ballots, their proofs and the signatures. The signatures include two random elements x^*, sig_2 and an encrypted first part. Finally he gets the intermediate results and the tally with the proof. Apart from the encrypted messages and the proofs, all other data are random and do not assist him in deciding b.

Suppose that the input of $\mathcal{S}$ is a Diffie-Hellman (DH) tuple. Then all the encryptions done by $\mathcal{S}$ are valid and the view of $\mathcal{A}$ is the same as the c-resist experiment. If the input is not a DH tuple then every encryption $\mathcal{S}$ did results in uniformly distributed elements in $\mathbb{G}^3$. $\mathcal{A}$'s view is the same as in the c-resist-ideal experiment.

These imply that

$$\mathbf{Adv}^{\text{c-resist}}_{ES,\mathcal{A}} = |\Pr[\mathcal{S}=1|\mathrm{DH}(g_1,g_2,h_1,h_2)] - \Pr[\mathcal{S}=1|\neg\mathrm{DH}(g_1,g_2,h_1,h_2)]|$$

which is equal to $\mathbf{Adv}^{DDH}_{\mathcal{S}}$ and so it is negligible if the DDH assumption holds.

Algorithm 1: c-resist

Input : $n, n_V, \mathbf{C}, D, \lambda$
Output: $result \in \{0,1\}$

$(V,U) \leftarrow \mathcal{A}(\mathbf{corrupt})$
$\{(sk_i, pk_i) \leftarrow \mathrm{reg}(sk_R, ID_i, \lambda)\}_{i\in[n]}$
$(j,\beta) \leftarrow \mathcal{A}(\{sk_i\}_{i\in V}, \mathbf{Coerce})$
if $\beta \notin \boldsymbol{C}$ or $j \notin U$ **then**
  output 0
end
$b \leftarrow_R \{0,1\}$
if $b = 0$ **then**
  $sk^* \leftarrow \mathrm{fakekey}(pk_T, sk_j, pk_j)$
  $\mathsf{BB} \Leftarrow$
   $\mathrm{auth}(sk_j, pk_j, sk_T, pk_T, \mathbf{C}, \beta, \lambda)$
else
  $sk^* \leftarrow sk_j$
end

$\mathsf{BB} \Leftarrow$
 $\mathrm{auth}(sk_i, pk_i, sk_T, pk_T, \mathbf{C}, D, \lambda)\}_{i\in U\setminus\{j\}}$
 $\mathsf{BB} \Leftarrow$
 $\mathcal{A}^{\mathrm{auth}(\cdot)}(\{sk_i\}_{i\in V}, sk^*, pk_T, \mathbf{C}, \mathsf{BB})$
$(\mathbf{X}, P) \leftarrow$
 $\mathrm{tally}(sk_T, \mathsf{BB}, \mathbf{C}, \{pk_i\}_{i\in V\cup U}, \lambda)$
$b' \leftarrow \mathcal{A}(\mathbf{X}, P, \mathsf{BB}, \mathbf{Guess})$
output $b == b'$

Algorithm 2: c-resist-ideal

Input : $n, n_V, \mathbf{C}, D, \lambda$
Output: $result \in \{0,1\}$

$(V,U) \leftarrow \mathcal{A}(\mathbf{corrupt})$
$\{(sk_i, pk_i) \leftarrow \mathrm{reg}(sk_R, ID_i, \lambda)\}_{i\in[n]}$
$(j,\beta) \leftarrow \mathcal{A}(\mathbf{Coerce})$
if $\beta \notin \boldsymbol{C}$ or $j \notin U$ **then**
  output 0
end
$b \leftarrow_R \{0,1\}$
$sk^* \leftarrow sk_j$
if $b = 0$ **then**
  $\mathsf{BB} \Leftarrow$
   $\mathrm{idauth}(sk_j, pk_j, sk_T, pk_T, \mathbf{C}, \beta, \lambda)$
end

$\{\mathsf{BB} \Leftarrow$
 $\mathrm{idauth}(sk_i, pk_i, sk_T, pk_T, \mathbf{C}, D, \lambda)\}_{i\in U\setminus\{j\}}$
 $\mathsf{BB} \Leftarrow$
 $\mathcal{A}^{\mathrm{idauth}(\cdot)}(\{sk_i\}_{i\in V}, sk^*, pk_T, \mathbf{C})$
 $(\mathbf{X}, P) \leftarrow$

 $\mathrm{tally}(sk_T, \mathsf{BB}, \mathbf{C}, \{pk_i\}_{i\in V\cup U}, \lambda)$
 $b' \leftarrow \mathcal{A}(\mathbf{X}, P, \Gamma, \mathbf{Guess})$
output $b == b'$

Finally, we must note that the exact level of protection each voter receives depends on the size of the anonymity set, i.e. the number of decoy votes cast

with their ID by other honest voters or organizations. We plan to incorporate this analysis in future versions of our work.

B Plain Okamoto-Schnorr CBS Scheme

We briefly present the simple Okamoto Shnorr CBS Scheme from [40]. The secret signing key consists of the values $s_1, s_2 \in \mathbb{Z}_q$ as in [5] with corresponding public verification key $v = g_1^{-s_1} g_2^{-s_2}$. During the signing and unblinding phases the public key k of the verifier is used. For the verification algorithm, the verifier checks the verification equation using the hash of the message and the commitment using the secret key $s \in \mathbb{Z}_q$. If the secret signer bit is 1, then the signature will be valid, otherwise the verification equation will not hold. Thus the verifier will learn the secret bit of the signer. We also assume the existence of a random oracle $\mathcal{H}$.

Common input: $\mathbb{G}, g_1, g_2, v, k \in \mathbb{G}$, $\mathcal{H} : \mathcal{M} \times \mathbb{G} \to \mathbb{Z}_q$
Signers's private input: $s_1, s_2 \in \mathbb{Z}_q : v = g_1^{-s_1} g_2^{-s_2}$ and $b \in \{0, 1\}$
Verifier's private input: $s \in \mathbb{Z}_q : k = g_1^s$
Recipient's private input: m

Commitment Phase. The Signer:

- Picks random $r_1, r_2 \leftarrow_R \mathbb{Z}_q$;
- Computes $x := g_1^{r_1} g_2^{r_2}$;
- Sends x to the recipient.

Blinding Phase. The Recipient:

- Selects blinding factors $u_1, u_2, d \leftarrow_R \mathbb{Z}_q$;
- Computes $x^* := x g_1^{u_1} g_2^{u_2} v^d, e^* := \mathcal{H}(m, x^*), e := e^* - d$;
- Sends e to the signer.

Signing Phase. The Signer:

- Computes $y_1 := r_1 + e s_1$, $y_2 := r_2 + e s_2$;
- If $b = 1$ then computes $(bsig_1, bsig_2) := (k^{y_1}, y_2)$;
- If $b = 0$ then selects randomly $(bsig_1, bsig_2) \leftarrow_R \mathbb{G} \times \mathbb{Z}_q$;
- Outputs $(x, e, bsig_1, bsig_2)$.

Unblinding Phase. The Recipient:

- Unblinds by computing $sig_1 := bsig_1 \cdot k^{u_1}$ and $sig_2 := bsig_2 + u_2$;
- Outputs $(m, x^*, e^*, sig_1, sig_2)$.

Verification Phase. The Verifier:

- Computes $e^{*'} := \mathcal{H}(m, x^*)$
- Computes ${y_1}' := sig_1$ and ${y_2}' := sig_2$;
- Checks if $x^{*s} = {y_1}' {g_2}^{{y_2}' \cdot s} v^{e^* \cdot s}$ and $e^{*'} = e^*$.

Fig. 5. The original CBS Scheme based on Okamoto Schnorr signatures

C Modified Okamoto-Schnorr CBS Scheme

The protocol in Figure 5 can be combined with a multiplicatively homomorphic encryption scheme. It can also be made more practical if the parties agree in a common method to randomly generate the commitment message x. Moreover, we can let the signer play the role of the verifier, as a way to send the secret bit to oneself in the future.

We present this modified version in Figure 6. Note that the unblinding of the first part of the signature, still occurs on the exponent, but this time in encrypted form.

Common input: $\mathbb{G}, g_1, g_2, h_1, v, x \in \mathbb{G}$, public keys k, h, $\mathcal{H} : \mathcal{M} \times \mathbb{G} \to \mathbb{Z}_q$
Signers's private input: $s \in \mathbb{Z}_q : k = g_1^s$, $z \in \mathbb{Z}_q : h = h_1^z$ and $b \in \{0, 1\}$
Recipient's private input: m

Blinding:

- $\mathcal{U}$ chooses random $u_1, u_2, d \leftarrow_R \mathbb{Z}_q$
- $\mathcal{U}$ computes $x^* := x g_1^{u_1} g_2^{u_2} v^d$, $e^* := \mathcal{H}(m, x^*)$ and $e := e^* - d$;
- $\mathcal{U}$ sends e to $\mathcal{S}$

Signing:

- $\mathcal{S}$ randomly selects $bsig_2 \leftarrow_R \mathbb{Z}_q$;
- If $b = 1$ then compute $bsig_1 := \mathtt{E}_h((x \cdot g_2^{-bsig_2} \cdot v^{-e})^s)$;
- If $b = 0$ then select randomly $bsig_1 \leftarrow \mathbb{G}^2$;
- Output $(x, e, bsig_1, bsig_2)$.

Unblinding:

- $\mathcal{U}$ computes $sig_1 := bsig_1 \cdot \mathtt{E}_h(k^{u_1})$ and $sig_2 := bsig_2 + u_2$;
- Output $(m, x^*, e^*, sig_1, sig_2)$.

Verification:

- $\mathcal{V}$ recomputes e^* as $\mathcal{H}(m, x^*)$. If $e^* \neq \mathcal{H}(m, x^*)$ $\mathcal{V}$ outputs 0.
- $\mathcal{V}$ decrypts the first part of the signature to receive ${y_1}' := \mathtt{D}_z(sig_1)$
- $\mathcal{V}$ sets ${y_2}' := sig_2$
- $\mathcal{V}$ checks if $x^{*s} = {y_1}' {g_2}^{{y_2}' \cdot s} v^{e^* \cdot s}$ and outputs 1 iff it holds

Fig. 6. Modified Conditional Blind Signatures