zkVoting : Zero-knowledge proof based coercion-resistant and E2E verifiable e-voting system

Seongho Park
Hanyang University
Seoul, Republic of Korea
seonghopark@hanyang.ac.kr

Jaekyoung Choi
Zkrypto
Seoul, Republic of Korea
cjk@zkrypto.com

Jihye Kim
Kookmin University
Seoul, Republic of Korea
jihyek@kookmin.ac.kr

Hyunok Oh
Hanyang University
Seoul, Republic of Korea
hoh@hanyang.ac.kr

***Abstract*—We introduce *zkVoting*, a coercion-resistant e-voting system that utilizes a fake keys approach based on a novel nullifiable commitment scheme. This scheme allows voters to receive both real and fake commitment keys from a registrar. Each ballot includes this commitment, but only the tallier can efficiently discern the fake ballots, simplifying the tally process to $\mathcal{O}(n)$ and ensuring coercion resistance. *zkVoting* also preserves voter anonymity by ensuring each ballot conceals the voter's identity. Additionally, by integrating zero-knowledge proofs, *zkVoting* achieves end-to-end (E2E) verifiability. We formally prove its security and demonstrate its practicality for real-world applications, with a ballot casting time of 2.3 seconds and a tally time of 3.9 milliseconds per ballot.**

1. Introduction

Remote electronic voting (e-voting) systems enhance efficiency and accessibility but face significant privacy challenges, such as voter coercion. Adversaries may compel voters to favor specific candidates, abstain, or vote under supervision, eroding fundamental democratic principles. Moreover, verifiability is indispensable in e-voting systems. Extensive research has explored verifiability criteria, highlighting its importance in ensuring election integrity [1]–[10]. While all ballots are published on the (public) bulletin board, individual verifiability allows voters to confirm that their ballots are posted as intended. Universal verifiability permits anyone to validate the tally results. End-to-end (E2E) verifiability assures voters that their votes are cast as intended, recorded as cast, and tallied as recorded.

Various e-voting systems designed to ensure coercion resistance and verifiability have been proposed [11]–[14]. The initial method to counteract coercion introduced by Juels, Catalano, and Jakobsson [11] (JCJ), employs fake credentials. This technique allows voters to create multiple fake credentials along with their genuine ones, enabling them to submit spurious ballots to mislead coercers. This strategy assumes that: 1) no coercion attacks occur during the registration phase; 2) voters can securely store and conceal cryptographic keys; and 3) they can convincingly lie if coerced. The JCJ system [11], as implemented in Civitas [12], requires a trusted registrar and tallier to ensure coercion resistance. The time complexity of the tally phase, driven by the plaintext equivalence test (PET) [15], is $\mathcal{O}({n_b}^2)$, where n_b denotes the total number of ballots. The system provides universal verifiability, predicated on the assumption of a trusted registrar.

Another approach involves revoting, where voters cast ballots multiple times using the same credentials, as explored in recent studies [13], [14], [16]. This method assumes that: 1) voters possess inalienable authentication to cast ballots; and 2) there is a designated period for revoting after potential coercion. Systems based on this approach require the additional assumption of a trusted voting server to ensure coercion resistance. VoteAgain [14] achieves coercion resistance with the aid of a trusted registrar, voting server, and tallier, and it reduces the time complexity of the tally phase to $\mathcal{O}(n_b \log n_b)$. Meanwhile, Loki [16] introduces a mechanism for flexible vote updating, eliminating the need for a specific revoting period post-coercion. This system attains coercion resistance through reliance on a trusted voting server and tallier, with a tally time complexity of $\mathcal{O}(n_v)$, where n_v denotes the number of voters. In Loki, voters and the trusted voting server collaboratively manage a shared list of real ballots, periodically re-randomizing each voter's latest or second-to-last ballot to thwart coercion. This approach, akin to employing fake credentials, encourages voters to deceive coercers about their true ballot list. However, it demands considerable computational resources from the voting server for re-randomizing ballots, which poses scalability challenges in large-scale elections.

We introduce *zkVoting*, a coercion-resistant e-voting system that advances from the traditional fake credentials approach to an innovative fake keys method. Central to this system is the newly introduced "nullifiable commitment scheme," a novel cryptographic building block conceptualized for the first time in this paper. In this scheme, a trusted entity equipped with a master secret key issues both real and fake commitment keys to voters. Each voter is allowed only one real commitment key but may receive any number of fake keys. Each key, whether real or fake, serves as a component of a casting key, enabling a voter to cast a single ballot. In scenarios of coercion, a voter may use a fake key to cast a vote. The trusted entity can perform a nullification operation using the master secret key,

where ballots committed with a real key retain their value, whereas those committed with a fake key are transformed to open with a *zero* value. This capability allows the trusted entity to efficiently identify and disregard ballots containing fake keys, significantly streamlining the tallying process and reducing computational complexity to $\mathcal{O}(n_b)$.

This architecture allows *zkVoting* to ensure end-to-end (E2E) verifiability *without* the need to trust the registrar, marking a significant divergence from most existing systems that rely on a shared secret voting key for coercion resistance. Unlike traditional systems [11], [14], [16] where the relationship between the voter and registrar typically requires trust to achieve verifiability, *zkVoting* constrains the registrar's role to ensuring privacy-related aspects such as ballot secrecy and coercion resistance. The unique setup of the casting key, comprising the voter's secret key and a commitment key (either real or fake), further underpins this trust architecture. The voter independently generates and retains the secret key, and the registrar, accessing only the voter's public key information, issues the commitment key. This arrangement allows the registrar to distinguish between real and fake keys but prevents it from casting votes due to the absence of access to the secret key.

To address potential ballot stuffing and ensure the integrity of the voting process, the registrar must also provide proof that the number of real keys issued matches the total number of eligible voters. This verification is facilitated by integrating a homomorphic property into the nullifiable commitment scheme, which permits the aggregation of commitments, each corresponding to the number of eligible voters after a nullifying process. The homomorphic nature of the commitment also simplifies the proof of tally results for universal verifiability and allows greater flexibility in choosing encryption schemes beyond homomorphic encryption.

1.1. Contributions

Our main contributions are as follows:

- We introduce a novel concept of a nullifiable commitment scheme. We formally define nullifiable encryption schemes, construct an efficient homomorphic nullifiable commitment scheme, and prove its security under the discrete log and decisional Diffie-Hellman assumptions.
- We present the security definitions of e-voting systems including coercion resistance in KTV [17] which is the enhanced model presented from JCJ [11]. We also present the security properties of individual verifiability, universal verifiability, E2E verifiability, eligibility verifiability, and voter anonymity.
- We develop a compiler to construct *zkVoting*, a coercion-resistant E2E verifiable e-voting system utilizing a homomorphic nullifiable commitment scheme and basic cryptographic building blocks. This system enables voters to cast indistinguishable fake ballots, evading coercion. It simplifies the tallying process with a linear time complexity and proves the result with a constant time and size complexity.
- We provide security proofs for our e-voting system compiled under the security assumptions of hybrid encryptions, nullifiable commitments, collision-resistant hash functions, and zero-knowledge proof systems. Our system ensures coercion resistance, E2E verifiability, eligibility verifiability, and voter anonymity, requiring weaker trust assumptions compared to prior works.
- We construct *zkVoting*, leveraging an efficient homomorphic nullifiable commitment. Our performance evaluation shows remarkable efficiency: ballot generation with a zero-knowledge proof in 2.3 seconds on a smartphone, ballot decryption in 3.9 ms per ballot, and tally proof generation in 360 ms on a server. Note that ballot decryption can be performed in parallel, further enhancing the tally efficiency.
- We incorporate a mechanism to detect malfunctioning devices using any trusted device. Upon casting a ballot, a receipt is issued to verify the proper functioning of the voter's device. The receipt contains the serial number (which locates its transaction in the blockchain) and the symmetric key (used for encrypting the message). Using any trusted device, the voter can confirm whether their ballot has been accurately cast as intended by decrypting the transaction and whether there are no duplicate ballots by checking the serial number. Note that this receipt is meaningful only to the voter who knows the real/fake key information.

1.2. Security properties in e-voting systems

Before discussing previous e-voting systems, we outline the fundamental properties of a secure e-voting system, as defined in existing literature [1]–[8], [11], [17], [18]. The security properties of an e-voting system are primarily categorized into privacy and verifiability and defined as follows:

- *Ballot privacy*(*BP*) : a ballot does not reveal the voter's choice.
- *Voter anonymity*(*VA*) : a ballot does not reveal the identity of voters.
- *Receipt freeness*(*RF*) : a voter is not able to prove their choice.
- *Coercion resistance*(*CR*) : a voter is able to cast a vote despite any influence.
- *Individual verifiability*(*IV*) : a voter must be able to verify their ballot containing their vote is included on the bulletin board.
- *Universal verifiability*(*UV*) : anyone must be able to verify the tally result represents all the ballots on the bulletin board.
- *End-to-end verifiability*(*E2E V*) : a voter can identify their unique ballot in a certain way and verify their ballot is cast-as-intended, their choice is recorded-as-cast and the result is tallied-as-recorded.
- *Eligibility verifiability*(*EV*) : anyone can verify the ballot is generated only from eligible voters with a voting right and all voters cast at most one vote.

TABLE 1: Security comparison of e-voting systems : Helios [19], BeleniosRF [20], Provotum [21], JCJ [11], Civitas [12], VoteAgain [14], Loki [16], and our *zkVoting*.

Properties	[19]	[20]	[21]	[11]/ [12]	[14]	[16]	*zkVoting*
BP	✓	✓	✓	✓	✓	✓	✓
VA	✗	✗	✗	✓	✓	✗	✓
RF	✗	✓	✗	✓	✓	✓	✓
CR	✗	✗	✗	✓	✓	✓	✓
IV	✓	✓	✓	✗	✓	✓	✓
UV	✓	✓	✓	✓	✓	✓	✓
E2E V	✓	✗	✓	✗	✓	✓	✓
EV	✗	✓	✓	✓	✓	✓	✓

1.3. Related work

Table 1 presents and compares the evaluation results of previous e-voting systems [11], [12], [14], [19]–[21], assessed based on the criteria outlined in §1.2. Some studies prioritizing coercion resistance often fall short of ensuring verifiability. On the other hand, research focusing on verifiability sometimes does so at the expense of privacy.

The most well-known e-voting system is Helios [19]. Helios presumes all voters to be honest and operates under the assumption of low coercion environments. However, Helios ensures E2E verifiability by issuing the receipt of the ballot. BeleniosRF [20] aims to achieve receipt freeness and E2E verifiability by improving Helios. However, the authority is responsible for key distribution and thus needs to be fully trusted. The authority with users' secret keys not only compromises voter anonymity but can also generate a proxy ballot before the voter creates their ballot. Eligibility verifiability is asserted by fully trusting the authority. Provotum [21] is a blockchain-based e-voting protocol that provides E2E verifiability, with voters keeping the transaction hash of the block. A trusted third party is responsible for verifying the eligibility of voters. Consequently, only eligible voters can cast a ballot, thereby ensuring eligibility verifiability. However, it does not guarantee receipt freeness and coercion resistance.

We provide a detailed analysis of various coercion-resistant e-voting systems in Table 2, including those utilizing fake credentials [11], [12] and those based on the revoting approach [14], [16]. Systems that employ the fake credentials approach exhibit high time complexity during the tally phase. In contrast, systems utilizing the revoting approach introduce a significant assumption: the necessity of a new trusted party. Specifically, VoteAgain requires a trusted tally server, and Loki depends on a trusted voting server.

JCJ [11] is the first coercion resistance e-voting system simultaneously achieving individual, universal, and eligibility verifiability. Civitas [12] is the implementation of the JCJ protocol. Its approach involves the use of fake credentials to deceive a coercer and requires extra computations to differentiate whether the ballot was generated with a fake credential which takes $\mathcal{O}({n_b}^2)$ computations where n_b is the number of ballots.

The other approach to achieve coercion resistance is enabling voters to re-vote using the same credential called revoting. VoteAgain [14] claims that using one credential and allowing revoting can achieve coercion resistance with E2E and eligibility verifiability. It reduces the time complexity to $\mathcal{O}(n_b \log n_b)$ for the tally server to filter the genuine ballots among n_b total ballots. However, it requires the strong assumption that all entities participating in the election should be trusted to ensure coercion resistance [22].

Loki [16] is the latest coercion-resistant voting system introducing a new paradigm called flexible vote updating which is built upon the revoting approach allowing voters to revote but eliminating the conventional assumption that voters require a specific period to revote after being coerced. Instead, voters and the trusted voting server confidentially share the intentions of voters. Loki demonstrates a time complexity of $\mathcal{O}(n_v)$, and E2E verifiability with an honest registrar. Nonetheless, Loki is applicable only for small-scale voting scenarios, since it receives revoted ballots or re-randomizes ballots for all voters every period. Additionally, each ballot contains the identity of the voter, which does

TABLE 2: Detailed security comparison of coercion-resistant e-voting systems where n_b is the number of ballots and n_v is the number of voters. All systems require a trusted bulletin board.

	JCJ [11] / Civitas [12]	VoteAgain [14]	Loki [16]	*zkVoting*
Trusted party for Verifiability	registrar	registrar	registrar	none
Trusted party for *BP*	registrar, tallier	registrar, tallier	registrar, tallier	registrar, tallier
Trusted party for *VA*	registrar, tallier	registrar	not ensured	none
Trusted party for *CR*	registrar, tallier	registrar, voting server, tallier	voting server, tallier	registrar, tallier
Paradigm for *CR*	fake credentials	revoting	flexible vote updating	fake keys
Time complexity for the tally phase	$\mathcal{O}(n_b^2)$	$\mathcal{O}(n_b \log n_b)$	$\mathcal{O}(n_v)$	$\mathcal{O}(n_b)$

not provide voter anonymity.

1.4. Outline

The remainder of this paper is organized as follows: Section 2 defines the notion of a nullifiable commitment scheme with homomorphism and instantiates a homomorphic nullifiable commitment scheme. Section 3 formally defines an e-voting system and its associated security properties. In Section 4, we propose *zkVoting*, a secure generic e-voting system using a homomorphic nullifiable commitment scheme and prove its security. We also instantiate a specific construction of *zkVoting*. Section 5 offers evaluations and discussions. Finally, Section 6 wraps up our study.

2. Nullifiable commitment

We introduce a novel concept named nullifiable commitment schemes. In this commitment scheme, a manager who has a master secret key and a master public key pair issues a commitment key. This commitment key can be either a real or a fake commitment key which is indistinguishable. Using the master secret key, the commitment can be nullified, which transforms the commitment into an alternative form of a commitment. Specifically, after nullified, the new commitment generated from a fake key is opened with a zero message while one from a real key is opened with the original message. We distinguish the nullifiable commitment key ck^* from the standard real/fake commitment keys ck, where ck^* is exclusively used to open all nullified commitments. While multiple ck's can be issued as requested by the user, after being nullified by a manager, each commitment can be opened with a unique nullifiable commitment key ck^*.

In nullifiable commitment schemes, the manager, upon generating a master key pair, issues commitment keys to the user. To ensure the integrity of key issuance, the manager interacts with the user to validate the authenticity of the received keys through a proving process (KeyProve described below). While this interactive design enables the user to verify the authenticity of received keys, universal verification is precluded, making it impossible for any party other than the user to distinguish between real and fake keys. The concept of deniability, widely discussed in cryptographic literature (e.g., [23]–[26]), plays a crucial role here. In our context, deniability ensures that a user can plausibly deny having received a fake commitment key and present it as genuine. This feature is facilitated by a simulation algorithm (SimKeyProve described below).

2.1. Protocols and properties

Definition 2.1. A nullifiable commitment scheme $\Pi_{\mathsf{NC}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyProve}, \mathsf{SimKeyProve}, \mathsf{Commit}, \mathsf{Nullify}, \mathsf{Open})$ is a set of protocols, which operates as follows:

- $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}^*)$: on input a security parameter λ, outputs a master public key mpk, a master secret key msk, and a nullifiable commitment key ck^*.
- $\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, b \in \{0,1\}) \rightarrow \mathsf{ck}$: takes a master public key mpk, master secret key msk and bit b and returns a commitment key ck. If $b = 1$, it outputs a real commitment key, otherwise, it outputs a fake one.
- $\mathsf{KeyProve}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}, b \in \{0,1\}) \rightarrow (\pi_{\mathsf{ck}}, (1/0))$: is an interactive protocol between a manager and a committer. Let the manager be a prover $\mathcal{P}$ takes a master public key mpk, a master secret key msk, a commitment key ck, and a bit b. Let the committer be a verifier $\mathcal{V}$ that takes a master public key mpk, a public key ck, and a bit b. Through the interaction, $\mathcal{V}$ can verify whether ck is correctly generated and $\mathcal{P}$ knows msk. If the verification passes, it returns 1. Otherwise, it returns 0.
- $\mathsf{SimKeyProve}(\mathsf{mpk}, \mathsf{ck}, b' \in \{0,1\}) \rightarrow (\pi'_{\mathsf{ck}}, (1/0))$: is a simulation algorithm that a verifier(committer) $\mathcal{V}$ always generates simulated proof π'_{ck} without knowing a master secret key msk. A user can prove that ck is a real commitment key even if it is a fake one, and vice versa. The simulated proof always passes the verification progress.
- $\mathsf{Commit}(\mathsf{ck}, \mathsf{m}; r) \rightarrow \mathsf{cm}$: takes a commitment key ck, a message m and a random r. It outputs a commitment cm.
- $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, \mathsf{m}, r) \rightarrow (1/0)$: takes a commitment key ck, a commitment cm, a message m and a random r. If cm is a valid commitment of m using a random r, it outputs 1; otherwise, it outputs 0.
- $\mathsf{Nullify}(\mathsf{msk}, \mathsf{cm}) \rightarrow \mathsf{cm}^*$: a manager transforms a standard commitment cm to a nullifiable commitment cm^* using a master secret key msk. Specifically, if cm is created with a fake commitment key, cm^* is opened with a message value set to zero. Conversely, if cm is formed with a real commitment key, cm^* is opened with the original message committed in cm.
- $\mathsf{Open}_{\mathsf{null}}(\mathsf{ck}^*, \mathsf{cm}^*, \mathsf{m}, r) \rightarrow (1/0)$: takes a nullifiable commitment key ck^*, nullified commitment cm^*, message m and random r. If cm^* is a nullified commitment from a real commitment, it outputs 1. Otherwise, it outputs 0.

For defining its security properties, given a security parameter 1^λ, we denote a negligible function as $\mathsf{negl}(\lambda)$. PPT refers to a probabilistic polynomial time. A nullifiable commitment scheme Π_{NC} should satisfy the following properties:

Hiding. The commitment from nullifiable commitment schemes ensures hiding even against a holder of the master secret key. In Figure 3 at Appendix A, the hiding property is defined as a game $\mathsf{Game}^{\mathsf{Hiding}}_{\mathcal{A},\Pi_{\mathsf{NC}}}$ between a challenger and a PPT adversary $\mathcal{A}$. $\mathcal{A}$ receives $(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}^*, \mathsf{ck})$ from the challenger, and submits $(\mathsf{m}_0, \mathsf{m}_1)$. A challenger randomly selects a bit $c \in \{0,1\}$ and sends a challenge commitment $\mathsf{cm} \leftarrow \mathsf{Commit}(\mathsf{ck}, \mathsf{m}_c; r)$ to $\mathcal{A}$ by choosing r randomly. The game outputs 1 if $\mathcal{A}$ correctly guesses c.

Definition 2.2. Π_{NC} satisfies hiding if $\mathbf{Adv}^{\mathsf{Hiding}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$ is negligible.

Binding. The committer, after committing a message, cannot find any other message of which commitment is equivalent to the original message commitment.

Definition 2.3. Π_{NC} ensures binding if the following holds:

$$\Pr\left[\begin{array}{c}(\mathsf{mpk},\mathsf{msk},\mathsf{ck}^*)\leftarrow\mathsf{Setup}(1^\lambda)\\ b\leftarrow\$\,\{0,1\},\mathsf{ck}\leftarrow\mathsf{KeyGen}(\mathsf{mpk},\mathsf{msk},b),\\ (\mathsf{m}_0,r_0,\mathsf{m}_1,r_1)\leftarrow\mathcal{A}(\mathsf{mpk},\mathsf{msk},\mathsf{ck}^*,\mathsf{ck}):\\ \mathsf{m}_0\neq\mathsf{m}_1\wedge\mathsf{m}_0,\mathsf{m}_1\neq\perp\wedge\\ \mathsf{Commit}(\mathsf{ck},\mathsf{m}_0;r_0)=\mathsf{Commit}(\mathsf{ck},\mathsf{m}_1;r_1)\end{array}\right]\leq\mathsf{negl}(\lambda).$$

Nullifiability. Nullifiability in nullifiable commitment schemes is connected with the correctness of the scheme. All commitments can be opened using Open. After nullification, using $\mathsf{Open}_{\mathsf{null}}$, a nullified commitment is opened with the original message m if the original commitment was generated using a real commitment key. If not, the nullified commitment will be opened with a zero message.

Definition 2.4. Π_{NC} satisfies nullifiability if the following holds:

$$\Pr\left[\begin{array}{c}(\mathsf{mpk},\mathsf{msk},\mathsf{ck}^*)\leftarrow\mathsf{Setup}(1^\lambda),\\ b\leftarrow\$\,\{0,1\},\mathsf{ck}\leftarrow\mathsf{KeyGen}(\mathsf{mpk},\mathsf{msk},b),\\ \forall\mathsf{m},r,\mathsf{cm}\leftarrow\mathsf{Commit}(\mathsf{ck},\mathsf{m};r),\\ \mathsf{Open}(\mathsf{ck},\mathsf{cm},\mathsf{m},r)=1:\\ \mathsf{Open}_{\mathsf{null}}(\mathsf{ck}^*,\mathsf{Nullify}(\mathsf{msk},\mathsf{cm}),b\cdot\mathsf{m},r)=1\end{array}\right]=1.$$

Indistinguishability of keys (IK). Consider two commitment keys, where each key, independently, can be either real or fake, or both could be of the same type. Given a commitment committed using one of these keys, the probability of correctly determining which key was used is negligible. This is a similar concept of key indistinguishability in encryption schemes (i.e, IK-CPA encryption scheme [27]), whereas ours is indistinguishability of keys for the commitment framework. Key indistinguishability is defined as a game in Figure 4 in Appendix A. In $\mathsf{Game}^{\mathsf{IK}}_{\mathcal{A},\Pi_{\mathsf{NC}}}$, $\mathcal{A}$ gets two commitment keys and submits a message. From a challenge commitment that is committed to the message using one of the two keys, $\mathcal{A}$ tries to guess which key was chosen.

Definition 2.5. Π_{NC} ensures indistinguishability of keys if $\mathbf{Adv}^{\mathsf{IK}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$ is negligible

Key deniability. While a msk holder can prove to a ck holder whether ck is real or fake using the interactive protocol KeyProve, any ck holder, without msk, can simulate proof using SimKeyProve asserting that the commitment key is real, even if it is a fake commitment key, and vice versa. This disables any party not participating in the interactive protocol distinguishing whether the commitment key is real or fake. Namely, if the ck holder is coerced to reveal their real key and proof from KeyProve, the ck holder can deny by submitting a fake key with a simulated proof from SimKeyProve.

Definition 2.6. In a nullifiable commitment scheme Π_{NC}, the key is deniable if for any offline PPT adversary $\mathcal{A}$, the following is negligible:

$$\left|\Pr\left[\begin{array}{c}(\mathsf{mpk},\mathsf{msk},\mathsf{ck}^*)\leftarrow\mathsf{Setup}(1^\lambda),\\ b\leftarrow\$\,\{0,1\},\mathsf{ck}\leftarrow\mathsf{KeyGen}(\mathsf{mpk},\mathsf{msk},b),\\ (\pi_{\mathsf{ck}},1)\leftarrow\mathsf{KeyProve}(\mathsf{mpk},\mathsf{msk},\mathsf{ck},b):\\ \mathcal{A}(\mathsf{mpk},\mathsf{ck}^*,\mathsf{ck},\pi_{\mathsf{ck}})=1\end{array}\right]-\Pr\left[\begin{array}{c}(\mathsf{mpk},\mathsf{msk},\mathsf{ck}^*)\leftarrow\mathsf{Setup}(1^\lambda),b\leftarrow\$\,\{0,1\},\\ \mathsf{ck}\leftarrow\mathsf{KeyGen}(\mathsf{mpk},\mathsf{msk},b),b'\leftarrow\$\,\{0,1\},\\ (\pi'_{\mathsf{ck}},1)\leftarrow\mathsf{SimKeyProve}(\mathsf{mpk},\mathsf{ck},b'):\\ \mathcal{A}(\mathsf{mpk},\mathsf{ck}^*,\mathsf{ck},\pi'_{\mathsf{ck}})=1\end{array}\right]\right|$$

Homomorphism. Nullifiable commitment schemes exhibit homomorphic properties when homomorphic operations are applied to their corresponding messages, which can also be nullifiable. When homomorphism is applied, the behavior depends on the nature of the original commitment: if it is a real commitment, then the original message will be included in the aggregation. Conversely, if the commitment is fake, a zero message will be aggregated instead. Consequently, the resulting aggregated commitment can be opened to reveal the cumulatively aggregated nullifiable messages.

Definition 2.7. A nullifiable commitment scheme, Π_{NC} ensures homomorphism if the following holds:

$$\Pr\left[\begin{array}{c}(\mathsf{mpk},\mathsf{msk},\mathsf{ck}^*)\leftarrow\mathsf{Setup}(1^\lambda),\\ \forall i,b_i\leftarrow\$\,\{0,1\},\mathsf{ck}_i\leftarrow\mathsf{KeyGen}(\mathsf{mpk},\mathsf{msk},b_i),\\ \mathsf{cm}_i\leftarrow\mathsf{Commit}(\mathsf{ck}_i,\mathsf{m}_i;r_i),\\ \mathsf{Open}(\mathsf{ck}_i,\mathsf{cm}_i,\mathsf{m}_i,r_i)=1,\\ \mathsf{cm}^*(=\sum\mathsf{cm}_i^*)\leftarrow\mathsf{Nullify}(\mathsf{msk},\sum\mathsf{cm}_i):\\ \mathsf{Open}_{\mathsf{null}}(\mathsf{ck}^*,\mathsf{cm}^*,\sum b_i\cdot\mathsf{m}_i,\sum r_i)=1\end{array}\right]=1.$$

2.2. Construction

We instantiate a homomorphic nullifiable commitment scheme Π_{NC} over an elliptic curve group $\mathbb{G}$ with prime order q where the discrete log (DL) assumption and the decisional Diffie-Hellman (DDH) assumption hold. We utilize the cryptographic hash function H. The construction is described in Construction 1.

Theorem 2.1. If the DL assumption and DDH assumption hold in the elliptic curve group $\mathbb{G}$, then for all probabilistic polynomial time adversaries $\mathcal{A}$, Π_{NC} in Construction 1, satisfies hiding, binding, nullifiability, indistinguishability of keys, key deniability, and homomorphism.

We provide the formal proof in Appendix B.

3. Secure e-voting systems

3.1. Entities and assumptions

An e-voting system is a set of protocols executed among a group of entities. The entities can be classified into three types, based on their roles and functionalities within the voting system.

Authority. An authority oversees the entire voting system, conducting the setup and registration phases. In the tally phase, the authority gathers all ballots, tallies the results, and publishes them.

Voters. A voter has a right to cast a ballot. They should register to cast a ballot. All voter has their id to participate in the election.

Public Bulletin Board ($\mathcal{BB}$). The bulletin board is public and all entities can access it. All ballots from voters are posted on the bulletin board.

Assumptions. When a voter registers their public key and obtains casting keys, the voter needs to undergo authentication with their id. This procedure occurs either offline or, in the case of an online scenario, assumes the use of an authenticated private channel. Once authenticated, each voter is provided with one real casting key and as many fake keys as desired. The number of casting keys received by a voter is known only to the voter and the authority, excluding any third party. Throughout the voting process, an anonymous communication channel like the mix-net [28]–[31] is presumed, aligning with a foundational premise common in most e-voting systems [11], [12], [14], [20].

We posit a period of non-interference for voters before the election ends. This assumption ensures that every voter has an opportunity to cast their ballot free from coercion, safeguarding the integrity of their vote. It precludes scenarios of constant surveillance or complete control over voters.

Threat Model. We consider a PPT coercer who tries to coerce voters into either voting for a specific candidate or abstaining from voting altogether with or without asking for the voter's casting key. A coercer possesses the capability to coerce any voter, potentially extending this coercion to all voters. A voter provides any information that the coercer requests *including* the casting key.

3.2. Protocols

An e-voting system $\Pi_{\mathsf{Vote}} = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{VerifyBallot}, \mathsf{Tally}, \mathsf{VerifyTally})$ is a set of protocols that operate as follows and can be classified into four phases:

Setup phase.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{PP}, \mathsf{SK})$: The authority generates an election key pair $(\mathsf{PP}, \mathsf{SK})$ and opens an election by setting public parameters PP such as PK, the start and end times, candidates, and eligible voters. The authority returns PP and SK.

Register phase.

- $\mathsf{Register}(\mathsf{PP}, \mathsf{SK}, id) \to (sk_{id}, pk_{id}, \mathsf{ck})$: A voter has their own identifier id, generates a key pair (sk_{id}, pk_{id}), registers pk_{id} for the election and obtain a casting key ck to cast a ballot.

Voting phase.

- $\mathsf{Vote}(\mathsf{PP}, (id, sk_{id}, pk_{id}, \mathsf{ck}, \mathsf{m})) \to B$: The voter encrypts a message m, casts their ballot B, and post to the bulletin board $\mathcal{BB}$.
- $\mathsf{VerifyBallot}(\mathsf{PP}, \mathsf{PK}, B) \to (1/0)$: Anyone can verify the ballot on $\mathcal{BB}$.

Tally phase.

- $\mathsf{Tally}(\mathsf{PP}, \mathsf{SK}) \to \mathcal{T}$: The authority collects all the ballots on $\mathcal{BB}$ and decrypts them. The authority tallies the messages and publishes the tally $\mathcal{T}$.
- $\mathsf{VerifyTally}(\mathsf{PP}, \mathcal{T}) \to (1/0)$: Anyone can verify the tally.

Construction 1 Proposed nullifiable commitment scheme construction

---

$\underline{\mathsf{Setup}(1^\lambda) :}$
$\rho \leftarrow\!\$\ \mathbb{Z}_q^*;\ g_1, g_2, g_3 \leftarrow\!\$\ \mathbb{G};\ g_4 = g_1^\rho$
$\mathsf{mpk} = (g_1, g_2, g_3, g_4);\ \mathsf{msk} = \rho;$
$\mathsf{ck}^* = (g_2/g_1^\rho, g_3);$
return $(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}^*);$

$\underline{\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, b \in \{0,1\}) :}$
parse $\mathsf{mpk} = (g_1, g_2, g_3, g_4);$
$h_1 \leftarrow\!\$\ \mathbb{G};\quad h_2 = h_1^{\mathsf{msk}} g_3^b;$
return $\mathsf{ck} = ((g_1, h_1), (g_2, h_2));$

$\underline{\mathsf{Commit}(\mathsf{ck}, \mathsf{m}; r) :}$
parse $\mathsf{ck} = ((g_1, h_1), (g_2, h_2));$
$C_1 = g_1^r h_1^{\mathsf{m}};\quad C_2 = g_2^r h_2{}^{\mathsf{m}};$
return $\mathsf{cm} = (C_1, C_2);$

$\underline{\mathsf{Nullify}(\mathsf{msk}, \mathsf{cm}) :}$
parse $\mathsf{cm} = (C_1, C_2);$
$\mathsf{cm}^* = C_2/(C_1^{\mathsf{msk}});$
return $\mathsf{cm}^*;$

$\underline{\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, \mathsf{m}, r) :}$
parse $\mathsf{ck} = ((g_1, h_1), (g_2, h_2));\ \mathsf{cm} = (C_1, C_2);$
if $C_1 = g_1^r h_1^{\mathsf{m}} \wedge C_2 = g_2^r h_2{}^{\mathsf{m}}$ **then return** 1;
else return 0;

$\underline{\mathsf{Open}_{\mathsf{null}}(\mathsf{ck}^*, \mathsf{cm}^*, \mathsf{m}, r) :}$
parse $\mathsf{ck}^* = (g^*, h^*);$
if $\mathsf{cm}^* = g^{*r} h^{*\mathsf{m}}$ **then return** 1;
else return 0;

$\underline{\mathsf{KeyProve}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}, b \in \{0,1\}) :}$
parse $\mathsf{mpk} = (g_1, g_2, g_3, g_4);$
parse $\mathsf{ck} = ((g_1, h_1), (g_2, h_2));$
$\mathcal{V} : c \leftarrow\!\$\ \mathbb{Z}_q^*;\ sends\ d \leftarrow H(c);$
$\mathcal{P} : t \leftarrow\!\$\ \mathbb{Z}_q^*;\ sends\ p = h_1^t;$
$\mathcal{V} : sends\ c;$
$\mathcal{P} :$ **assert** $d = H(c);\ sends\ k = t + \mathsf{msk} \cdot c;$
$\mathcal{V} :$ **if** $h_1^k = p(h_2/g_3^b)^c$ **then**
return $(\pi_{ck} = (p, k, c), 1);$ **else return** 0

$\underline{\mathsf{SimKeyProve}(\mathsf{mpk}, \mathsf{ck}, b' \in \{0,1\}) :}$
parse $\mathsf{mpk} = (g_1, g_2, g_3, g_4);$
parse $\mathsf{ck} = ((g_1, h_1), (g_2, h_2));$
$(k', c') \leftarrow\!\$\ \mathbb{Z}_q^*;$
$p' = h_1^{k'}/(h_2/g_3^{b'})^{c'};$
return $(\pi'_{ck} = (p', k', c'), 1);$

3.3. Security properties

From here, we write a set $\{x_1, x_2, \cdots, x_N\}$ as a vector $\mathbf{X}$. The operation of appending element e to a set $\mathbf{X}$ will be represented as $\mathbf{X} \Leftarrow e$.

Ballot privacy. The game $\mathsf{Game}^{BP}_{\mathcal{A},\Pi_{\mathsf{Vote}}}$ in Figure 5 in Appendix A defines a game for ballot privacy. An adversary $\mathcal{A}$ selects two messages of equal length, and a challenging voter $\mathcal{V}$ flips a coin b and casts a ballot with a message m_b. $\mathcal{V}$ provides a ballot and $\mathcal{A}$ makes a guess b' about the value b.

Definition 3.1. For any PPT adversary $\mathcal{A}$, let the advantage be $\mathbf{Adv}^{BP}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda) = \left|\Pr[\mathsf{Game}^{BP}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda) = 1] - \frac{1}{2}\right|$. The voting system satisfies ballot privacy if $\mathbf{Adv}^{BP}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda)$ is negligible.

Coercion resistance. The coercion resistance game $\mathsf{Game}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}$ is outlined in Figure 1, formulizing KTV framework [17] which is a strengthened version of JCJ's definition [11]. JCJ's limitation lies in its inability to distinguish between a voter intentionally casting a specific message and being forced to do so. KTV framework addresses this by ensuring voters can vote freely despite coercion attempts.

For each coercion attack $CA_i \in \mathbf{CA}$, the attacker wants voters to be coerced by running a corrupted strategy cs'_i. However, voters can use the counter strategy cs_i and evade CA_i with probability 1. Coercion resistance is characterized by the attacker's minimal chance of distinguishing between the voters' defensive counter strategy cs_i and the manipulated, corrupted strategy cs'_i.

$\mathsf{Game}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_i, cs_i, cs'_i)$:

..... *Setup and register phase*
$(\mathsf{PP}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda)$;
All voters register for the election and obtain a casting key.
..... *Voting phase*
Honest voters vote at any time during the voting phase.
$\mathcal{A}$ performs a coercion attack CA_i to a targeted voter $\mathcal{V}$.
$b \leftarrow\$ \{0, 1\}$;
if $b = 0$ **then** $\mathcal{V}$ performs a counter strategy cs_i.
else $\mathcal{V}$ performs a corrupted strategy cs'_i.
end if
..... *Tally phase*
$\mathcal{T} \leftarrow \mathsf{Tally}(\mathsf{PP}, \mathsf{SK})$;
..... *Challenging phase*
$b' \leftarrow \mathcal{A}(\mathsf{PP}, \mathcal{T})$;
return $b = b'$;

Figure 1: Coercion resistance game $\mathsf{Game}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}$

Definition 3.2. For all PPT adversaries $\mathcal{A}$ and all coercion attack $CA_i \in \mathbf{CA}$, let $\mathbf{Succ}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_i, cs_i)$ be the probability of the voter successfully evading coercion attack CA_i using the counter strategy cs_i. Let the corrupted strategy cs'_i be the strategy operated by the voter when coerced. Let the advantage of the attacker distinguishing two strategies cs_i and cs'_i be $\mathbf{Adv}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_i, cs_i, cs'_i) = \left|\Pr[\mathsf{Game}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_i, cs_i, cs'_i) = 1] - \frac{1}{2}\right|$. Then, Π_{Vote} is coercion-resistant if for all coercion attack $CA_i \in \mathbf{CA}$, the followings hold:

- $\mathbf{Succ}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_i, cs_i) = 1$
- $\mathbf{Adv}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_i, cs_i, cs'_i) \leq \mathsf{negl}(\lambda)$

Voter anonymity. The goal of $\mathcal{A}$ against voter anonymity is to distinguish the voter's id associated with a specific ballot, even when $\mathcal{A}$ has access to the authority's secret key. This implies that voter anonymity is maintained even if $\mathcal{A}$ can get the message of the ballot, ensuring the voter associated with a particular ballot remains unidentified. We present voter anonymity using a game, $\mathsf{Game}^{VA}_{\mathcal{A},\Pi_{\mathsf{Vote}}}$, between a challenging voter $\mathcal{V}$ and a PPT adversary $\mathcal{A}$, as shown in Figure 6 in Appendix A. $\mathcal{V}$ provides n public-private key and casting key pairs to $\mathcal{A}$. $\mathcal{A}$ selects two public keys and make vote queries to $\mathcal{V}$, providing a message m. $\mathcal{V}$ flips a coin b, casts a ballot B_b and provides it to $\mathcal{A}$. $\mathcal{A}$ makes a guess b' about the value b. The success of $\mathcal{A}$ in guessing b should be negligible to ensure voter anonymity.

Definition 3.3. For any PPT adversary $\mathcal{A}$, let the advantage be $\mathbf{Adv}^{VA}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda) = \left|\Pr[\mathsf{Game}^{VA}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda) = 1] - \frac{1}{2}\right|$. Π_{Vote} ensures voter anonymity if $\mathbf{Adv}^{VA}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda)$ is negligible.

Individual verifiability. After the voting phase, the voter must be able to verify that their ballot is uploaded on the bulletin board and was not deleted or changed.

Universal verifiability. When the election ends and the authority publishes the tally, anyone can verify the tally includes all ballots on the bulletin board.

E2E verifiability. E2E verifiability is a combination of three steps; the voter can verify their ballot was cast-as-intended, their ballot was recorded-as-cast, and the election result was tallied-as-recorded.

Eligibility verifiability. Anyone can verify that the ballot was cast by an eligible voter.

4. *zkVoting*

In this section, we propose a compiler that generically builds a coercion-resistant and E2E-verifiable e-voting system *zkVoting* by using homomorphic nullifiable commitment scheme(§2) and other building blocks in §4.1.

4.1. Cryptographic building blocks

Definition 4.1 (Hybrid encryption)**.** Given a public key encryption scheme $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a symmetric key encryption scheme $\Pi_{\mathsf{SKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, a hybrid encryption scheme $\Pi_{\mathsf{HBE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ works as follows:

- $\mathsf{Gen}(1^\lambda) \rightarrow (\mathsf{pk}, \mathsf{sk})$: using $\Pi_{\mathsf{PKE}}.\mathsf{Gen}$, outputs a public key pk and a secret key sk.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}; k) \rightarrow \mathcal{CT}$: takes $k \leftarrow \Pi_{\mathsf{SKE}}.\mathsf{Gen}$, a public key pk, and a message m and outputs ciphertext $\mathcal{CT} \leftarrow (\Pi_{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{pk}, k), \Pi_{\mathsf{SKE}}.\mathsf{Enc}(k, \mathsf{m}))$.

- $\mathsf{Dec}(\mathsf{sk}, \mathcal{CT} = (C_1, C_2)) \rightarrow \mathsf{m}$: decrypts a message $\mathsf{m} \leftarrow \Pi_{\mathsf{SKE}}.\mathsf{Dec}(\Pi_{\mathsf{PKE}}.\ \mathsf{Dec}(\mathsf{sk}, C_1), C_2)$.

Both Π_{PKE} and Π_{SKE} are IK-CPA [27] and IND-CPA secure encryption scheme. Its ciphertexts are indistinguishable under the chosen plaintext attack. Also, from the ciphertext, no adversary can identify which key was used.

Definition 4.2 (Zero-knowledge proof)**.** For some relation $\mathcal{R}$, a zero-knowledge proof scheme $\Pi_{\mathsf{ZKP}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ works as follows:

- $\mathsf{Setup}(1^\lambda, \mathcal{R}) \rightarrow \mathsf{param}$: generates public parameters.
- $\mathsf{Prove}(\mathcal{R}, \mathsf{param}, \Phi; w) \rightarrow \pi$: given public parameters, a statement Φ and witness w, the prover $\mathcal{P}$ generates proof π that $(\Phi; w) \in \mathcal{R}$.
- $\mathsf{Verify}(\mathcal{R}, \mathsf{param}, \Phi, \pi) \rightarrow (1/0)$: given public parameters, a statement Φ and proof π, the verifier $\mathcal{V}$ confirms if π is valid.

A zero-knowledge proof should satisfy completeness, soundness, and zero-knowledge. *Completeness* means that if for any statement $(\Phi; w) \in \mathcal{R}$, $\mathcal{V}$ always accepts any proof $\pi \leftarrow \mathcal{P}(\Phi; w)$, that is, $\forall(\Phi; w) \in \mathcal{R}, \mathcal{V}(\Phi, \pi) = 1$. *Soundness* implies that for any statement $\Phi \notin \mathcal{R}$, the probability that any cheating $\hat{\mathcal{P}}$ can produce a valid proof $\pi \leftarrow \hat{\mathcal{P}}(\Phi)$ such that $\mathcal{V}(\Phi, \pi) = 1$ is negligible. *Zero-knowledge* represents that for all $(\Phi; w) \in \mathcal{R}$, there exists a PPT simulator $\mathcal{S}$ with access only to the public statement Φ that can output a valid proof $\pi' \leftarrow \mathcal{S}(\Phi)$ such that $\mathcal{V}(\Phi, \pi') = 1$.

If a system is only secure against a computationally bounded adversarial prover then it becomes an *argument* system. If a system proves that the prover knows a witness then it provides *knowledge soundness*. An argument system is considered *succinct* if the proof size and the verifying time are succinct.

A membership proof scheme is a cryptographic method used to prove that a certain element belongs to a specific set.

Definition 4.3 (Membership proof)**.** The membership proof scheme $\Pi_{\mathsf{M}} = (\mathsf{BuildSet}, \mathsf{Verify})$ for a set $\mathbf{S}$ is a set of algorithms, which operates as follows:

- $\mathsf{BuildSet}(\mathbf{S}) \rightarrow rt$: takes a set $\mathbf{S}$ and outputs a set ID rt.
- $\mathsf{Verify}(rt, e) \rightarrow (1/0)$: takes a rt, and an element e, and outputs 1 if e is a member of the set.

4.2. Compiling coercion-resistant secure e-voting systems

To enhance coercion resistance in the electronic voting system, the commitment keys of the nullifiable commitment scheme serve as casting keys. The inclusion of a fake casting commitment key allows voters to submit a fake ballot, creating a mechanism to resist coercion. Each casting key is authorized to cast one ballot. While only one real casting key is permitted, multiple fake casting keys can be issued to counteract potential coercion attacks. At the end of the registration phase, the casting keys are uploaded to the bulletin board and used to verify voter eligibility.

The central concept for ensuring verifiability is to use a zero-knowledge proof system (ZKP) to demonstrate that the algorithm has been executed as intended, thereby ensuring that all entities, such as voters and authorities, follow the protocol. In scenarios such as clash attacks [32] where two voters vote for the same candidate, malicious voting devices or authority could deceive the voters with the same ballot. Two voters do not realize they look at the same ballot. Therefore, voter names or pseudonyms should be attached to the ballot to satisfy E2E verifiability [10]. To achieve E2E verifiability, we embrace the concept of a serial number [33]. The serial number is bound to the secret key of the voter and the casting key using a cryptographic hash function H. This serial number also prevents double voting. These techniques enable the construction of E2E verifiable e-voting systems, providing a high level of confidence in the integrity of the election. Every encrypted ballot is associated with a commitment using ZKP, and the tallying process is efficiently proven by the tallier through homomorphically aggregated and nullifiable commitments.

The full construction Π_{Vote} works as follows:

Setup phase.

- $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{PP}, \mathsf{SK})$: the authority generates a key pair $(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}^*) \leftarrow \Pi_{\mathsf{NC}}.\mathsf{Setup}(1^\lambda)$ for a nullifiable commitment, another key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Pi_{\mathsf{HBE}}.\mathsf{Gen}(1^\lambda)$ for a hybrid encryption and public parameters for all relations $\mathsf{param} \leftarrow \Pi_{\mathsf{ZKP}}.\mathsf{Setup}(1^\lambda, \mathcal{R})$ for zero-knowledge proofs. Each relation is defined below. Then, the authority sets public values PP including a master public key mpk, a nullifiable commitment key ck^*, a public key pk, public parameters param, a unique election ID e and other information for the voting system such as the start time, end time, and a list of n_v number of eligible voters. The authority returns $\mathsf{PP} = (\mathsf{mpk}, \mathsf{ck}^*, \mathsf{pk}, \mathsf{param}, e)$ and $\mathsf{SK} = (\mathsf{msk}, \mathsf{sk})$.

Register phase.

- $\mathsf{Register}(\mathsf{PP}, \mathsf{SK}, id) \rightarrow (sk_{id}, pk_{id}, \mathsf{ck})$: a voter has their own ID id and obtains a casting key ck which is a commitment key in a nullifiable commitment scheme. The voter sets a secret key sk_{id} and generates a public key $pk_{id} = H(sk_{id})$ and proof $\pi_{id} \leftarrow \Pi_{\mathsf{ZKP}}.\mathsf{Prove}(\mathcal{R}_{id}, \mathsf{param}, pk_{id}; sk_{id})$. Then the voter registers a public key and sets a bit $b \in \{0, 1\}$ meaning that the voter requires a real casting commitment key if $b = 1$ or a fake one if $b = 0$. The authority verifies $\Pi_{\mathsf{ZKP}}.\mathsf{Prove}(\mathcal{R}_{id}, \mathsf{param}, pk_{id}, \pi_{id})$ and issues a casting key ck with proof $\pi_{\mathsf{ck}} \leftarrow \Pi_{\mathsf{ZKP}}.\mathsf{Prove}(\mathcal{R}_{ck}, \mathsf{param}, \mathsf{mpk}, \mathsf{ck}, b; \mathsf{msk})$ guaranteeing the correctness of ck. Proof π_{ck} can be generated using $\Pi_{\mathsf{NC}}.\mathsf{KeyProve}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}, b)$. Here, the voter can obtain a fake casting key multiple times but a real key only once. At the end of the register phase, the authority collects all voters' casting keys and public keys $\mathbf{CKlist} \Leftarrow (\mathsf{ck}, pk_{id})$. They are then uploaded to the bulletin board $\mathcal{BB}$ in the form of a set $rt = \Pi_{\mathsf{M}}.\mathsf{BuildSet}(\mathbf{CKlist})$.

While a voter can only obtain a real casting key once, some might attempt to request it multiple times. For this reason, the authority secretly stores $\mathbf{CKdb} \Leftarrow (id, (\mathsf{ck}, pk_{id}), b)$ to verify whether the voter has previously issued a real casting key. Furthermore, to guarantee that the authority does not issue more real keys than the total number of voters, ensuring each voter receives only one real key, the authority provides proof. It establishes that a nullifiable commitment $\overline{\mathsf{cm}}^*$ from the aggregated commitment $\overline{\mathsf{cm}}$, which is the summation of commitment cm_i committed using message 1, equals the total number of eligible voters n_v, since a message from a real commitment is considered. This is confirmed by $\pi_{total} = \Pi_{\mathsf{ZKP}}.\mathsf{Prove}(\mathcal{R}_{total}, \mathsf{ck}^*, \overline{\mathsf{cm}}, n_v; \mathsf{msk})$, where $\mathsf{cm}_i \leftarrow \Pi_{\mathsf{NC}}.\mathsf{Commit}(\mathsf{ck}, 1; 0)$ and $\overline{\mathsf{cm}} \leftarrow \sum \mathsf{cm}_i$.

Voting phase.

- $\mathsf{Vote}(\mathsf{PP}, (id, pk_{id}, sk_{id}, \mathsf{ck}, \mathsf{m})) \rightarrow B$: the voter generates a unique serial number of their ballot, $sn = H(e, \mathsf{ck}, sk_{id})$. The voter encrypts a ciphertext $\mathcal{CT} \leftarrow \Pi_{\mathsf{HBE}}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}; k)$ and a commitment $\mathsf{cm} \leftarrow \Pi_{\mathsf{NC}}.\mathsf{Commit}(\mathsf{ck}, \mathsf{m}; r)$ with a random r. Then, the voter generates proof $\pi_{vote} \leftarrow \Pi_{\mathsf{ZKP}}.\mathsf{Prove}(\mathcal{R}_{vote}, \mathsf{PP}, rt, e, sn, \mathcal{CT}, \mathsf{cm}; \mathsf{m}, r, sk_{id}, pk_{id}, \mathsf{ck})$. The voter takes a receipt of a serial number and a random symmetric key k which is used in the hybrid encryption scheme. Also, the voter generates a ballot $B = (e, sn, \mathcal{CT}, \mathsf{cm}, \pi_{vote})$ and sends it to the bulletin board $\mathcal{BB}$. By utilizing the receipt, the voter can verify whether their device functions correctly as intended. After casting a ballot and uploading it to the bulletin board, voters can use any trusted device to locate their ballot using a serial number. They can then decrypt the corresponding portion of the ciphertext using k and open the commitment, ensuring that their voting device has accurately encrypted the ballot as intended.
- $\mathsf{VerifyBallot}(\mathsf{PP}, \mathsf{PK}, B) \rightarrow (1/0)$: $\mathcal{BB}$ checks if there exist duplicated ballots with the same serial number, and verifies the validity of the ballot through $\Pi_{\mathsf{ZKP}}.\mathsf{Verify}(\mathcal{R}_{vote}, \mathsf{PP}, rt, e, sn, \mathcal{CT}, \mathsf{cm})$. The ballot is uploaded on $\mathcal{BB}$ if VerifyBallot outputs 1.

Tally phase.

- $\mathsf{Tally}(\mathsf{PP}, \mathsf{SK}) \rightarrow \mathcal{T}$: the authority downloads all the ballots on $\mathcal{BB}$. The authority decrypts each ballot's ciphertext and nullifies the corresponding commitment. Now, the authority can open the nullifiable commitment with m, r. If a commitment successfully opens, it indicates that the commitment was real (i.e. $b = 1$). The authority proceeds to tally the message, computing $(\mathsf{m}_{tally} = \sum b_i \mathsf{m}_i)$. All nullifiable commitments and corresponding randoms are aggregated to $\mathsf{cm}^*_{sum}, r_{sum}$. Subsequently, the authority generates a proof $\pi_{tally} \leftarrow \Pi_{\mathsf{ZKP}}(\mathcal{R}_{tally}, \mathsf{PP}, \mathsf{cm}_{sum}, \mathsf{m}_{tally}; \mathsf{msk}, r_{sum})$. This method streamlines the process by nullifying all commitments and utilizing the homomorphic properties of the nullifiable commitment scheme. Instead of verifying each ballot individually, the authority produces a single comprehensive proof that only real ballots are tallied.
- $\mathsf{VerifyTally}(\mathsf{PP}, \mathcal{T}) \rightarrow (1/0)$: Anyone can download all the ballots on $\mathcal{BB}$, and verify the tally using zero-knowledge proof.

Relations. We describe the following relations for use in our zero-knowledge proofs:

- $\mathcal{R}_{id}$ ensures the voter generates a valid $pi_{id} = H(sk_{id})$.

$$\mathcal{R}_{id} = \{(pk_{id}; sk_{id}) : pk_{id} = H(sk_{id})\}.$$

- $\mathcal{R}_{ck}$ ensures that the authority issues a casting key that the voter has required (knowledge of b) and knowledge of msk. Proof can be made $\pi_{ck} \leftarrow \Pi_{\mathsf{NC}}.\mathsf{KeyProve}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}, b)$.

$$\mathcal{R}_{ck} = \{(\mathsf{mpk}, \mathsf{ck}, b; \mathsf{msk}) : \mathsf{ck} = \Pi_{\mathsf{NC}}.\mathsf{Gen}(\mathsf{mpk}, \mathsf{msk}, b)\}$$

- $\mathcal{R}_{total}$ guarantees that the authority issues n_v real casting keys to ensure that each voter receives a single real casting key respectively.

$$\mathcal{R}_{total} = \left\{ \begin{array}{c} (\mathsf{ck}^*, \overline{\mathsf{cm}}, n_v; \mathsf{msk}) : \\ \overline{\mathsf{cm}}^* \leftarrow \Pi_{\mathsf{NC}}.\mathsf{Nullify}(\mathsf{msk}, \overline{\mathsf{cm}}), \\ \Pi_{\mathsf{NC}}.\mathsf{Open}_{\mathsf{null}}(\mathsf{ck}^*, \overline{\mathsf{cm}}^*, n_v, 0) = 1 \end{array} \right\}$$

- $\mathcal{R}_{vote}$ ensures that the voter generated the ballot honestly.

$$\mathcal{R}_{vote} = \left\{ \begin{array}{c} \begin{pmatrix} \mathsf{PP}, rt, e, sn, \mathcal{CT}, \mathsf{cm}; \\ \mathsf{m}, r, sk_{id}, pk_{id}, k, \mathsf{ck} \end{pmatrix} : \\ pk_{id} = H(sk_{id}) \wedge sn = H(e, \mathsf{ck}, sk_{id}) \wedge \\ \Pi_{\mathsf{M}}.\mathsf{Verify}(rt, (\mathsf{ck}, pk_{id})) = 1 \wedge \\ \mathcal{CT} = \Pi_{\mathsf{HBE}}.\mathsf{Enc}(\mathsf{pk}, \mathsf{m}; k) \wedge \\ \mathsf{cm} = \Pi_{\mathsf{NC}}.\mathsf{Commit}(\mathsf{ck}, \mathsf{m}; r) \end{array} \right\}$$

- $\mathcal{R}_{tally}$ ensures that the authority knows the master secret key to identify the real ballots and collects all of them.

$$\mathcal{R}_{tally} = \left\{ \begin{array}{c} (\mathsf{PP}, \mathsf{cm}_{sum}, \mathsf{m}_{tally}; \mathsf{msk}, r_{sum}) : \\ \mathsf{cm}^*_{sum} = \Pi_{\mathsf{NC}}.\mathsf{Nullify}(\mathsf{msk}, \mathsf{cm}_{sum}), \\ \Pi_{\mathsf{NC}}.\mathsf{Open}_{\mathsf{null}}(\mathsf{ck}^*, \mathsf{cm}^*_{sum}, \mathsf{m}_{tally}, r_{sum}) = 1 \end{array} \right\}$$

4.3. Security analysis

Coercion scenarios. We categorize coercion attacks $\mathbf{CA}$ into four types. For each $CA_i \in \mathbf{CA}$, voters are capable of running the counter strategy cs_i and evading CA_i with a probability of 1. Furthermore, the attacker can distinguish between the strategies of the voters who use the counter strategy cs_i and the corrupted strategy cs'_i with negligible probability.

The four types of coercion attacks are: 1) requiring a casting key to cast a ballot on behalf of the voter, 2) requiring a casting key to be absent, 3) instructing the choice, or 4) forcing the voter to be absent. In each scenario, a PPT adversary $\mathcal{A}$ tries to coerce the voter $\mathcal{V}$ to cast a β message ballot in some way, but the voter $\mathcal{V}$ can establish the counter strategy cs and corrupted strategy cs' indistinguishable from the coercion attacker $\mathcal{A}$ to cast a α message ballot. Note that $\mathcal{BB}$ is public, and $\mathcal{A}$ can observe $\mathcal{BB}$ and count the number of uploaded ballots. This means that the number of ballots cast in one scenario should be the same.

In each case, voters can evade coercion attacks by following instructions using fake casting keys and can cast their

ballot using a real casting key as they wish. Specifically, voters can evade $\mathbf{CA}$ as follows: 1) and 2) provide a fake casting key, 3) cast a ballot with a fake casting key, or 4) be absent with a fake casting key, and later cast a ballot using a real casting key. The attacker cannot distinguish whether the given casting key is real or fake. Since key deniability holds for our nullifiable commitment scheme, through the SimKeyProve algorithm, the voter can generate proof that any casting key is a real casting key. The attacker cannot distinguish whether the ballot was cast using a real casting key or a fake key. Even if they could, since our voting system provides ballot privacy, the attacker only gets the knowledge that the ballot is valid and the attacker cannot determine whose casting key was used. This makes it meaningless for the attacker to instruct voters or buy ballots.

lemma 4.1. For a PPT adversary $\mathcal{A}$ performing a coercion attack $CA_1 \in \mathbf{CA}$, then the voter can set a counter strategy cs_1 against a corrupted strategy cs'_1 where the followings hold:

- $\mathbf{Succ}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_1, cs_1) = 1$
- $\mathbf{Adv}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_1, cs_1, cs'_1) \leq \mathsf{negl}(\lambda)$

Proof. In the first scenario, the coercive attacker $\mathcal{A}$ performs a coercion attack $CA1$ that $\mathcal{A}$ requires a casting key of the targeted voter $\mathcal{V}$. $\mathcal{A}$ tries to cast a ballot with a message β on behalf of $\mathcal{V}$ using the acquired casting key. Then the voter $\mathcal{V}$ performs the counter strategy cs_1 as follows:

1) The voter submits a fake casting key.
2) $\mathcal{A}$ casts a ballot with a message β using the given key.
3) The voter casts an intended ballot with a message α using a real casting key.

The corrupted strategy cs'_1 works as follows:

1) The voter submits a real casting key.
2) $\mathcal{A}$ casts a ballot with a message β using the given key.
3) The voter casts a ballot with any message using a fake casting key.

It is obvious that by running the counter strategy (if $b = 0$) the voter can evade coercion with probability 1. Therefore, we have, $\mathbf{Succ}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_1, cs_1) = 1$. Since Π_{NC} is an indistinguishable commitment scheme and the voter can simulate proof, whether $\mathcal{A}$ acquired a real key or fake key, $\mathcal{A}$ cannot distinguish which strategy $\mathcal{V}$ performed. The total number of ballots on $\mathcal{BB}$ is perfectly identical. We also have $\mathbf{Adv}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_1, cs_1, cs'_1) \leq \mathsf{negl}(\lambda)$. □

lemma 4.2. For a PPT adversary $\mathcal{A}$ performing a coercion attack $CA_2 \in \mathbf{CA}$, then the voter can set a counter strategy cs_2 against a corrupted strategy cs'_2 where the followings hold:

- $\mathbf{Succ}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_2, cs_2) = 1$
- $\mathbf{Adv}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_2, cs_2, cs'_2) \leq \mathsf{negl}(\lambda)$

Proof. In the second scenario, the coercive attacker $\mathcal{A}$ performs a coercion attack CA_2 that $\mathcal{A}$ requires a casting key of the targeted voter $\mathcal{V}$. $\mathcal{A}$ tries to be absent using the acquired casting key. Then the voter $\mathcal{V}$ performs the counter strategy cs_2 as follows:

1) The voter submits a fake casting key.
2) The voter casts an intended ballot with a message α using a real casting key.

The corrupted strategy cs'_2 works as follows:

1) The voter submits a real casting key.
2) $\mathcal{A}$ does nothing to be absent using the given key.
3) The voter casts a ballot with any message using a fake casting key.

It is obvious that by running the counter strategy (if $b = 0$) the voter can evade coercion with probability 1. Therefore, we have, $\mathbf{Succ}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_2, cs_2) = 1$. The total number of ballots on $\mathcal{BB}$ is perfectly identical, and indistinguishable holds in Π_{NC}, $\mathcal{A}$ cannot distinguish strategies. We also have $\mathbf{Adv}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_2, cs_2, cs'_2) \leq \mathsf{negl}(\lambda)$. □

lemma 4.3. For a PPT adversary $\mathcal{A}$, given a coercion attack $CA_3 \in \mathbf{CA}$, then the voter can set a counter strategy cs_3 and a corrupted strategy cs'_3 where the following hold:

- $\mathbf{Succ}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_3, cs_3) = 1$
- $\mathbf{Adv}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_3, cs_3, cs'_3) \leq \mathsf{negl}(\lambda)$

Proof. In the second scenario, the coercion attack CA_3 of $\mathcal{A}$ is to instruct $\mathcal{V}$ to cast a ballot β. Then the voter $\mathcal{V}$ runs a counter strategy cs_3 as follows:

1) $\mathcal{V}$ casts a coerced ballot with a message β using a fake key.
2) $\mathcal{V}$ casts a intended ballot with a message α using a real key.

For the corrupted strategy cs'_3, $\mathcal{V}$ should cast any ballot regardless of the message, to make two strategies indistinguishable, working as follows:

1) $\mathcal{V}$ casts a coerced ballot with a message β using a real key.
2) $\mathcal{V}$ casts any ballot using a fake key.

If $\mathcal{V}$ runs the counter strategy then $\mathcal{V}$ can evade coercion with probability 1, $\mathbf{Succ}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_3, cs_3) = 1$. From the tally, no information on $\mathcal{V}$'s ballot is leaked. Owing to ballot privacy, $\mathcal{A}$ cannot distinguish the ballots and which strategy $\mathcal{V}$ performed. We also have $\mathbf{Adv}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_3, cs_3, cs'_3) \leq \mathsf{negl}(\lambda)$. □

lemma 4.4. For a PPT adversary $\mathcal{A}$, given a coercion attack $CA_4 \in \mathbf{CA}$, if ballot privacy holds, then the voter can set a counter strategy cs_4 and a corrupted strategy cs'_4 where the following hold:

- $\mathbf{Succ}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_4, cs_4) = 1$
- $\mathbf{Adv}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_4, cs_4, cs'_4) \leq \mathsf{negl}(\lambda)$

Proof. The coercion attack CA_4 of $\mathcal{A}$ is to force the voter to be absent from the election. Then the voter $\mathcal{V}$ runs a counter strategy cs_4 as follows:

1) $\mathcal{V}$ abstains from voting using a fake casting key.
2) $\mathcal{V}$ casts an intended ballot with a message α using a real key.

For the corrupted strategy cs'_4, $\mathcal{V}$ should cast any ballot regardless of the message, to make two strategies indistinguishable, working as follows:

1) $\mathcal{V}$ abstains from voting and does not cast a ballot using a real key.
2) $\mathcal{V}$ casts a ballot with any message using a fake key.

$\mathcal{V}$ runs the counter strategy and can evade coercion with probability 1. We have $\mathbf{Succ}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_4, cs_4) = 1$. From the tally and proof, no information on $\mathcal{V}$'s ballot is leaked. Owing to ballot privacy and voter anonymity, the ballots leak no information. $\mathcal{A}$ cannot distinguish which strategy $\mathcal{V}$ runs. Therefore, we also have $\mathbf{Adv}^{CR}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda, CA_4, cs_4, cs_4') \leq \mathsf{negl}(\lambda)$. □

Theorem 4.1. Given a nullifiable commitment scheme Π_{NC}, a hybrid encryption scheme Π_{HBE}, a cryptographic hash function H, and a zero-knowledge proof system Π_{ZKP}, a voting system Π_{Vote}= (Setup, Register, Vote, VerifyBallot, Tally, VerifyTally) ensures ballot privacy, receipt freeness, coercion resistance, E2E verifiability, eligibility verifiability, and voter anonymity.

Proof. **Ballot privacy**. In the ballot $B = (e, sn, \mathcal{CT}, \mathsf{cm}, \pi_{vote})$, the voter's message is included in the ciphertext $\mathcal{CT}$, the commitment cm and the zero-knowledge proof π_{vote}. An adversary $\mathcal{A}$ wins the game if $\mathcal{A}$ distinguishes between ciphertexts $\mathcal{CT}_0$ and $\mathcal{CT}_1$, between commitments cm_0 and cm_1, or between zero-knowledge proofs $\pi_{vote,0}$ and $\pi_{vote,1}$. We can construct auxiliary adversaries $\mathcal{B}$, C and $\mathcal{D}$ such that $\mathbf{Adv}^{BP}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda) \leq \mathbf{Adv}^{\mathcal{CT}}_{\mathcal{B}}(1^\lambda) + \mathbf{Adv}^{\mathsf{cm}}_{\mathcal{C}}(1^\lambda) + \mathbf{Adv}^{\pi_{vote}}_{\mathcal{D}}(1^\lambda)$. Here, $\mathcal{B}$ tries to distinguish the ciphertexts, $\mathcal{C}$ tries to distinguish commitments and $\mathcal{D}$ tries to distinguish zero-knowledge proofs. Π_{HBE} is both IK-CPA and IND-CPA secure, from definition 2.2, we know that the commitments satisfy hiding, thus, $\mathbf{Adv}^{\mathcal{CT}}_{\mathcal{B}}(1^\lambda)$ and $\mathbf{Adv}^{\mathsf{cm}}_{\mathcal{C}}(1^\lambda)$ have a negligible probability. Additionally, since π_{vote} is zero knowledge, the advantage $\mathbf{Adv}^{\pi_{vote}}_{\mathcal{D}}(1^\lambda)$ is negligible. Then, the probability of winning the game $\mathsf{Game}^{BP}_{\mathcal{A},\Pi_{\mathsf{Vote}}}$ is only a random guessing. Thus, the advantage $\mathbf{Adv}^{BP}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda)$ is negligible, and Π_{Vote} ensures ballot privacy.

Receipt freeness. Receipt freeness can be proven informally. Since the e-voting system ensures coercion resistance, it naturally ensures receipt freeness. Therefore, we focus on proving coercion resistance.

A voter gets a receipt after casting a ballot. If the voter submits the receipt to the adversary, the adversary finds the ballot on $\mathcal{BB}$ and tries to decrypt the message. The adversary may find the exact ballot on $\mathcal{BB}$ since the receipt contains a serial number. However, the adversary cannot nullify the commitment, meaning that the ballot may be a fake ballot. Thus, the attacker cannot be convinced that the voter cast the decrypted message. Therefore, the voter cannot prove their choice, and receipt freeness is ensured.

Coercion resistance. From lemmas 4.1, 4.2, 4.3, and 4.4, for all cases, $\mathcal{V}$ can perform the counter strategy with probability 1. Also, $\mathcal{A}$ cannot distinguish the counter strategy from the corrupted strategy. Thus, Π_{Vote} is coercion-resistant.

Voter anonymity. From the ballot $B = (e, sn, \mathcal{CT}, \mathsf{cm}, \pi_{vote})$, the possible way for $\mathcal{A}$ distinguishing the casting key of the ballot is to distinguish one of $sn, \mathsf{cm}, \pi_{vote}$. We have, $\mathbf{Adv}^{VA}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda) \leq \mathbf{Adv}^{sn}_{\mathcal{B}}(1^\lambda) + \mathbf{Adv}^{\mathsf{cm}}_{\mathcal{C}}(1^\lambda) + \mathbf{Adv}^{\pi_{vote}}_{\mathcal{D}}(1^\lambda)$. The serial number sn is the output of the hash function where its inputs are $(e, sk_{id}, \mathsf{ck})$. It looks like a random in $\mathcal{A}$'s view. Π_{NC} is an indistinguishable nullifiable commitment scheme that the adversary cannot guess ck. We know π_{vote} is zero-knowledge. Then, $\mathcal{A}$ can win the game $\mathsf{Game}^{VA}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda)$ only by random guessing, $1/2$. We have that $\mathbf{Adv}^{VA}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda) \leq \mathsf{negl}(\lambda)$. As a result, the voting system ensures voter anonymity.

Individual verifiability. The voter can compute the serial number of their ballots and find their ballot in the $\mathcal{BB}$. Only the ballots that have passed VerifyBallot are uploaded to the blockchain. The relation $\mathcal{R}_{vote}$ includes checks for the hash function for the serial number, the encryption process, and the commit process. Given the soundness of Π_{ZKP}, a ballot generated with different sk^*_{id} and ck^* can pass VerifyBallot with only a negligible probability.

Universal verifiability. Anyone can retrieve the inputs for the relation $\mathcal{R}_{tally}$ and execute VerifyTally. If a dishonest authority were to exclude certain ballots, include ballots not uploaded on $\mathcal{BB}$, or announce an incorrect tally, the soundness of Π_{ZKP} dictates that the verification process would be unsuccessful.

E2E verifiability. Consider an honest voter $\mathcal{V}$ who casts a ballot $\mathcal{B}$. After the voting phase, $\mathcal{BB}$ discloses all the ballots. Then, $\mathcal{V}$ can identify their ballot using their serial number $sn = H(e||\mathsf{ck}||sk_{id})$. If the ballot passes VerifyBallot, the voter is assured that their vote was cast-as-intended. If the content of the ballot matches what the voter originally cast, they can be assured it was recorded-as-cast. In the tally phase, the authority tallies all ballots and uploads the result along with the zero-knowledge proof that the tally comes from all ballots. Anyone can verify tallied-as-recorded. In Appendix C, we prove E2E verifiability using Cortier's definition [9].

Eligibility verifiability. In the relation $\mathcal{R}_{vote}$ contains the membership check to prove the membership of (ck, pk_{id}). By the soundness of Π_{ZKP}, a ballot generated using the arbitrarily generated ck' and/or pk'_{id} can pass VerifyBallot only with a negligible probability. □

4.4. Construction

In Construction 2, we instantiate *zkVoting* using the proposed nullifiable commitment scheme Π_{NC} from §2.2 and describe details for the practical instantiation. Unless otherwise specified, each phase operates as described in §4. The public bulletin board is implemented with the Blockchain bulletin board using a smart contract denoted as $\mathcal{BBB}$.

Message Format. There is no limitation on the message space for an encryption scheme. It can vary depending on the specific encryption scheme used and can be tailored to adequately represent a candidate. For our nullifiable commitment scheme to achieve homomorphism, the aggregated message is represented as a commitment with a nullifiable commitment key. Note that although a single block message is represented in Construction 1, it can be extended to support a message with multiple blocks by allowing a vector of g_3, h_1, and h_2. For instance, k

blocks in a message $\mathsf{m} = (\mathsf{m}_1||\mathsf{m}_2||\cdots||\mathsf{m}_k)$ are committed to $(C_1, C_2) = (g_1^r \prod_{i=1}^k h_{1,i}^{m_i}, g_2^r \prod_{i=1}^k h_{2,i}^{m_i})$ with $\mathsf{ck} = ((g_1, h_{1,1}, \ldots, h_{1,k}), (g_2, h_{2,1}, \ldots, h_{2,k}))$ where $h_{2,i} = h_{1,i}^{msk} g_{3,i}^b$. The keyProof π_{ck} becomes $(p_1, \ldots, p_k, k, c)$, and verifier $\mathcal{V}$ checks if $h_{1,i}^k = p_i(h_{2,i}/g_{3,i}^b)^c$ for $1 \leq i \leq k$ in KeyProve. Therefore, in designing the message format, we meticulously ensure that the size of each block is sufficiently large to prevent overflow following aggregate commitments. However, the message structure in an encryption scheme does not necessarily have to mirror the format used in commitments.

5. Experiment

We have implemented the proposed voting system. For a zero-knowledge proof system, we use the Gro16 zk-SNARK proof system in C++ [34], [35]. This system is ideal for blockchain applications due to its constant-sized proofs and efficient verification. A key aspect of this system is the trusted setup phase, where the authority prepares common random strings and uploads them to $\mathcal{BBB}$. For a collision-resistant hash function, we adopt the MiMC7 hash function [36] optimized for zk-SNARKs. For membership proofs, we employed a Merkle hash tree. The nullifi-

Construction 2 *zkVoting* construction

Setup phase

$\mathsf{Setup}(1^\lambda)$:

$(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}^*) \leftarrow \Pi_{\mathsf{NC}}.\mathsf{Setup}(1^\lambda)$; $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Pi_{\mathsf{HBE}}.\mathsf{Gen}(1^\lambda)$; $\mathsf{param} \leftarrow \Pi_{\mathsf{ZKP}}(1^\lambda, \mathcal{R}_{id}, \mathcal{R}_{ck}, \mathcal{R}_{total}, \mathcal{R}_{vote}, \mathcal{R}_{tally})$; $e \leftarrow\$ \{0,1\}^\lambda$
return $(\mathsf{PP} = (\mathsf{mpk}, \mathsf{ck}^*, \mathsf{pk}, \mathsf{param}, e), \mathsf{SK} = (\mathsf{msk}, \mathsf{sk}))$;

Register phase

Authority$(\mathsf{PP}, \mathsf{SK})$ — **Voter**(PP, id);

$\mathsf{Register}(\mathsf{PP}, \mathsf{SK}, id)$:

parse $\mathsf{PP} = (\mathsf{mpk}, \mathsf{ck}^*, \mathsf{pk}, \mathsf{param}, e)$; $\mathsf{SK} = (\mathsf{msk}, \mathsf{sk})$;

parse $\mathsf{PP} = (\mathsf{mpk}, \mathsf{ck}^*, \mathsf{pk}, \mathsf{param}, e)$;
$sk_{id} \leftarrow\$ \{0,1\}^\lambda$; $pk_{id} = H(sk_{id})$;
$\pi_{id} = \Pi_{\mathsf{ZKP}}.\mathsf{Prove}(\mathcal{R}_{id}, \mathsf{param}, pk_{id}; sk_{id})$;

$\xleftarrow{id, pk_{id}, \pi_{id}, b}$ $b \leftarrow 1(real)$ or $0(fake)$;

assert $\Pi_{\mathsf{ZKP}}.\mathsf{Verify}(\mathcal{R}_{id}, \mathsf{param}, pk_{id}, \pi_{id}) = 1$;
$\mathsf{ck} = \Pi_{\mathsf{NC}}.\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, b)$;
assert $(id, (\mathsf{ck}, pk_{id}), b) \notin \mathbf{CKdb}$;
$\mathbf{CKdb} \Leftarrow (id, (\mathsf{ck}, pk_{id}), b)$;
$\mathbf{CKlist} \Leftarrow (\mathsf{ck}, pk_{id})$;

$\xrightarrow{\mathsf{ck}}$

assert $\Pi_{\mathsf{NC}}.\mathsf{KeyProve}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}, b) = (\pi_{ck}, 1)$; $\longleftrightarrow$ assert $\Pi_{\mathsf{NC}}.\mathsf{KeyProve}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}, b) = (\pi_{ck}, 1)$;

upload (ck, pk_{id}) to $\mathcal{BBB}$;
$\overline{\mathsf{cm}} \mathrel{+}= \Pi_{\mathsf{NC}}.\mathsf{Commit}(\mathsf{ck}, 1; 0)$;

upload $\overline{\mathsf{cm}}, rt = \Pi_{\mathsf{M}}.\mathsf{BuildSet}(\mathbf{CKlist}), \pi_{total} = \Pi_{\mathsf{ZKP}}(\mathcal{R}_{total}, \mathsf{param}, ck^*, \overline{\mathsf{cm}}, n_v; \mathsf{msk})$ to $\mathcal{BBB}$;

Voting phase

$\mathsf{Vote}(\mathsf{PP}, \mathsf{ck}, \mathbf{CKlist}, pk_{id}, sk_{id}, \mathsf{m})$:

parse $\mathsf{PP} = (\mathsf{mpk}, \mathsf{ck}^*, \mathsf{pk}, \mathsf{param}, e)$;
$sn = H(e, \mathsf{ck}, sk_{id})$; $\mathsf{cm} \leftarrow \Pi_{\mathsf{NC}}.\mathsf{Commit}(\mathsf{ck}, \mathsf{m}; r)$;
$\mathcal{CT} \leftarrow \Pi_{\mathsf{HBE}}.\mathsf{Enc}(\mathsf{pk}, (\mathsf{m}, r); k)$;
$\pi_{vote} = \Pi_{\mathsf{ZKP}}.\mathsf{Prove}(\mathcal{R}_{vote}, \mathsf{PP}, rt, e, sn, \mathcal{CT}, \mathsf{cm}; \mathsf{m}, r, sk_{id}, pk_{id}, k, \mathsf{ck})$;
send B *to* $\mathcal{BBB}$;
return $(k, B = (e, sn, \mathcal{CT}, \mathsf{cm}, \pi_{vote}))$;

$\mathsf{VerifyBallot}(\mathsf{PP}, B)$:

parse $\mathsf{PP} = (\mathsf{mpk}, ck^*, \mathsf{pk}, \mathsf{param}, e)$;
parse $B = (e, sn, \mathcal{CT}, \mathsf{cm}, \pi_{vote})$;
assert $sn \notin \mathcal{BBB}$;
return $\Pi_{\mathsf{ZKP}}.\mathsf{Verify}(\mathcal{R}_{vote}, \mathsf{PP}, rt, e, sn, \mathcal{CT}, \mathsf{cm}, \pi_{vote})$;

Tally phase

$\mathsf{Tally}(\mathsf{PP}, \mathsf{SK})$:

parse $\mathsf{PP} = (\mathsf{mpk}, \mathsf{ck}^*, \mathsf{pk}, e)$; $\mathsf{SK} = (\mathsf{msk}, \mathsf{sk})$;
foreach $B_i = (e, sn_i, \mathcal{CT}_i, \mathsf{cm}_i)$ **in** $\mathcal{BBB}$ **do**
  $(\mathsf{m}_i, r_i) \leftarrow \Pi_{\mathsf{HBE}}.\mathsf{Dec}(\mathsf{sk}, \mathcal{CT}_i)$; $\mathsf{cm}_{sum} \mathrel{+}= \mathsf{cm}_i$; $r_{sum} \mathrel{+}= r_i$;
  if $\Pi_{\mathsf{NC}}.\mathsf{Open}_{\mathsf{null}}(\mathsf{ck}^*, \Pi_{\mathsf{NC}}.\mathsf{Nullify}(\mathsf{msk}, \mathsf{cm}_i), \mathsf{m}_i, r_i) = 1$ **then**
    $\mathsf{m}_{tally} \mathrel{+}= \mathsf{m}_i$; **endif endforeach**
$\pi_{tally} = \Pi_{\mathsf{ZKP}}.\mathsf{Prove}(\mathcal{R}_{tally}, \mathsf{PP}, \mathsf{cm}_{sum}, \mathsf{m}_{tally}; \mathsf{msk}, r_{sum})$;
return $\mathcal{T} = (\mathsf{m}_{tally}, \pi_{tally})$;

$\mathsf{VerifyTally}(\mathsf{PP}, \mathcal{T})$:

parse $\mathcal{T} = (\mathsf{m}_{tally}, \pi_{tally})$;
foreach $B_i = (e, sn_i, \mathcal{CT}_i, \mathsf{cm}_i)$ **in** $\mathcal{BBB}$ **do**
  $\mathsf{cm}_{sum} \mathrel{+}= \mathsf{cm}_i$; **endforeach**
return $\Pi_{\mathsf{ZKP}}.\mathsf{Verify}(\mathcal{R}_{tally}, \mathsf{PP}, \mathsf{cm}_{sum}, \mathsf{m}_{tally}, \pi_{tally})$;

able commitment scheme was implemented over an elliptic curve, following the methodologies in [37], [38]. Given that this curve has an order of 254 bits, it supports messages up to 254 bits in length. Consequently, we set the maximum number of eligible voters, n_v, to 2^{16}, and the number of candidates to 15.

TABLE 3: Analysis for relations

	$\mathcal{R}_{id}$	$\mathcal{R}_{total}$	$\mathcal{R}_{vote}$	$\mathcal{R}_{tally}$
# of constraints	365	2,243	15,829	6,111
CRS size	126 kB	0.9MB	5.7 MB	2.4 MB

TABLE 3 presents an analysis of the relations. $\mathcal{R}_{vote}$ contains proof of membership and encryption, leading to a large number of constraints and having the largest CRS size. An increase in CRS size not only lengthens the proving time but also impacts the overall size of the application.

The authority is implemented as a server using Intel Xeon Gold 6242R CPU with 250 GB RAM. $\mathcal{BBB}$ is designed as a blockchain smart contract (Ethereum [39] ganache) using the Truffle framework. Voting devices are shown in TABLE 4.

After voters download the application, they prove their identity and register for the election. Figure 2(a) illustrates the time cost for the register phase on various devices.

Figure 2(b) depicts the execution time for the voting phase. The voting can be completed within a few seconds on every device. Especially on higher-performance devices C, D, and E, the voting time is less than 3s. The voting transaction is processed in $\mathcal{BBB}$, which requires 347,363 gas in Ethereum. Note that since the gas fee of 350K gas is quite expensive in public blockchains, a dedicated sidechain will be appropriate for $\mathcal{BBB}$.

Figure 2(c) presents the execution time for the tally phase where x-axis represents the number of ballots. The tally process consists of ballot decryption and tally proof generation. It requires 3.9ms to decrypt each ballot and 360ms to generate the tally proof on the server. Note that the tally can be improved using parallel processing.

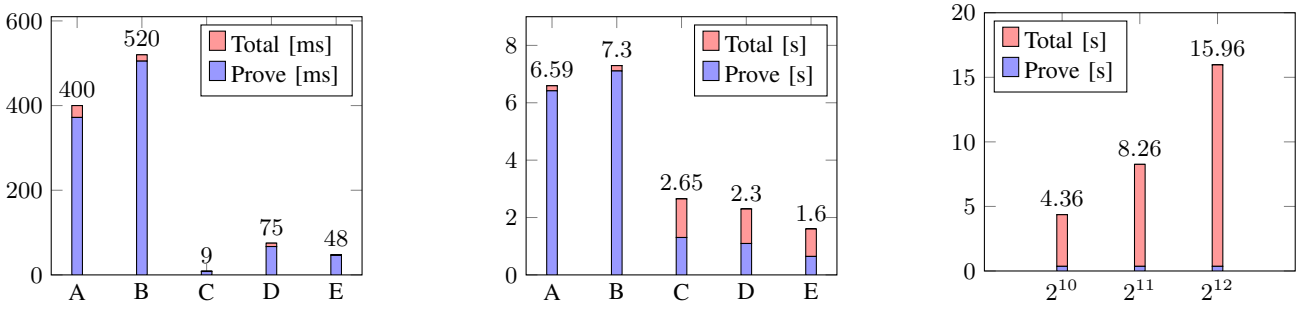
Comparison with other coercion-resistant e-voting systems. When compared to existing systems, Civitas [12] achieves a casting time of 345ms and tallies 500 ballots in about 5 hours (36 sec per ballot). In contrast, VoteAgain [14] has a casting time of 1.6 seconds and can tally 100,000 ballots in 13.5 minutes (8.1 ms per ballot). In Loki [16], it takes approximately 78 ms to generate a ballot with additional periodic obfuscation time which is close to the ballot generating time. For tallying the ballot, it takes 7.1 ms per ballot.

TABLE 4: Voting devices

Voting devices		Specification	
A	Device1 (Galaxy S22+)	OS:	Android 13
		Processor:	Qualcomm Snapdragon 8 Gen1
		RAM:	8 GB LPDDR5 SDRAM
B	Device2 (Galaxy Note 10+)	OS:	Android 12
		Processor:	Samsung Exynos 9825
		RAM:	12 GB LPDDR4X SDRAM
C	Device3 (iPhone 12 Mini)	OS:	iOS 16.03
		Processor:	Apple A14 Bionic
		RAM:	4 GB LPDDR4X SDRAM
D	Device4 (iPhone 14 Pro)	OS:	iOS 16.03
		Processor:	Apple A16 Bionic
		RAM:	6 GB LPDDR5 SDRAM
E	Device5 (MacBook Pro 16)	OS:	macOS Ventura 13.3.1
		Processor:	Apple M1 Pro
		RAM:	32 GB

6. Conclusion

In this paper, we introduce a novel cryptographic tool, a nullifiable commitment scheme, designed specifically for situations involving coercion. Leveraging this scheme, we present *zkVoting*, a coercion-resistant, E2E-verifiable e-voting system. Our system offers enhanced security properties while streamlining the tallying process. We provide comprehensive theoretical frameworks and formal security proofs to validate the robustness of our system. Furthermore, we underscore the practicality of *zkVoting* by elaborating on its seamless integration into smartphone and blockchain applications, showcasing its readiness for real-world deployment.



(a) Register phase (b) Voting phase (c) Tally phase

Figure 2: Performance with 16 Merkle tree depth and 15 candidates: (a) register time and (b) voting time on various devices, and (c) tally time by varying the number of ballots. Note that the y-axis represents the execution time in seconds.

References

[1] S. T. Ali and J. Murray, "An overview of end-to-end verifiable voting systems," *CoRR*, vol. abs/1605.08554, 2016. [Online]. Available: http://arxiv.org/abs/1605.08554

[2] H. Jonker, S. Mauw, and J. Pang, "Privacy and verifiability in voting systems: Methods, developments and trends," *Computer Science Review*, vol. 10, pp. 1–30, 2013.

[3] R. Küsters, T. Truderung, and A. Vogt, "Verifiability, privacy, and coercion-resistance: New insights from a case study," in *2011 IEEE Symposium on Security and Privacy*, 2011, pp. 538–553.

[4] J. Lee, J. Choi, J. Kim, and H. Oh, "Saver: Snark-friendly, additively-homomorphic, and verifiable encryption and decryption with rerandomization," Cryptology ePrint Archive, Report 2019/1270, 2019, https://eprint.iacr.org/2019/1270.

[5] V. Cortier, D. Galindo, S. Glondu, and M. Izabachene, "Election verifiability for helios under weaker trust assumptions," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 327–344.

[6] V. Cortier, D. Galindo, R. Küsters, J. Mueller, and T. Truderung, "Sok: Verifiability notions for e-voting protocols," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 779–798.

[7] V. Cortier and J. Lallemand, "Voting: You can't have privacy without individual verifiability," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 53–66.

[8] S. Kremer, M. Ryan, and B. Smyth, "Election verifiability in electronic voting protocols," in *Computer Security – ESORICS 2010*, D. Gritzalis, B. Preneel, and M. Theoharidou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 389–404.

[9] V. Cortier, D. Galindo, S. Glondu, and M. Izabachène, "Election verifiability for helios under weaker trust assumptions," in *Computer Security - ESORICS 2014*, M. Kutyłowski and J. Vaidya, Eds. Cham: Springer International Publishing, 2014, pp. 327–344.

[10] R. Küsters and J. Müller, "Cryptographic security analysis of e-voting systems: Achievements, misconceptions, and limitations," in *Electronic Voting: Second International Joint Conference, E-Vote-ID 2017, Bregenz, Austria, October 24-27, 2017, Proceedings 2*. Springer, 2017, pp. 21–41.

[11] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, 2005, pp. 61–70.

[12] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a secure voting system," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 354–368.

[13] D. Achenbach, C. Kempka, B. Löwe, and J. Müller-Quade, "Improved coercion-resistant electronic elections through deniable re-voting," {*USENIX*} *Journal of Election Technology and Systems ({JETS})*, vol. 3, pp. 26–45, 2015.

[14] W. Lueks, I. Querejeta-Azurmendi, and C. Troncoso, "VoteAgain: A scalable coercion-resistant voting system," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 1553–1570. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/lueks

[15] M. Jakobsson and A. Juels, "Mix and match: Secure function evaluation via ciphertexts," in *Advances in Cryptology—ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security Kyoto, Japan, December 3–7, 2000 Proceedings 6*. Springer, 2000, pp. 162–177.

[16] R. Giustolisi, M. S. Garjan, and C. Schuermann, "Thwarting last-minute voter coercion," *Cryptology ePrint Archive*, 2023.

[17] R. Küsters, T. Truderung, and A. Vogt, "A game-based definition of coercion resistance and its applications," *Journal of Computer Security*, vol. 20, no. 6, pp. 709–764, 2012.

[18] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi, "A comprehensive analysis of game-based ballot privacy definitions," Cryptology ePrint Archive, Paper 2015/255, 2015, https://eprint.iacr.org/2015/255. [Online]. Available: https://eprint.iacr.org/2015/255

[19] B. Adida, "Helios: Web-based open-audit voting." in *USENIX security symposium*, vol. 17, 2008, pp. 335–348.

[20] P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo, "Beleniosrf: A non-interactive receipt-free electronic voting scheme," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1614–1625. [Online]. Available: https://doi.org/10.1145/2976749.2978337

[21] C. Killer, B. Rodrigues, E. J. Scheid, M. Franco, M. Eck, N. Zaugg, A. Scheitlin, and B. Stiller, "Provotum: A blockchain-based and end-to-end verifiable remote electronic voting system," in *2020 IEEE 45th Conference on Local Computer Networks (LCN)*. IEEE, 2020, pp. 172–183.

[22] T. Haines, J. Müller, and I. Querejeta-Azurmendi, "Scalable coercion-resistant e-voting under weaker trust assumptions," in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, 2023, pp. 1576–1584.

[23] R. Bendlin, J. B. Nielsen, P. S. Nordholt, and C. Orlandi, "Lower and upper bounds for deniable public-key encryption," in *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*. Springer, 2011, pp. 125–142.

[24] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, "Deniable encryption," in *Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*. Springer, 1997, pp. 90–104.

[25] R. Canetti, S. Park, and O. Poburinnaya, "Fully deniable interactive encryption," in *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part I 40*. Springer, 2020, pp. 807–835.

[26] A. O'Neill, C. Peikert, and B. Waters, "Bi-deniable public-key encryption," in *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*. Springer, 2011, pp. 525–542.

[27] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval, "Key-privacy in public-key encryption," in *Advances in Cryptology — ASIACRYPT 2001*, C. Boyd, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 566–582.

[28] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," in *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, ser. Lecture Notes in Computer Science, vol. 1233. Springer, 1997, pp. 103–118.

[29] D. Boneh and P. Golle, "Almost entirely correct mixing with applications to voting," in *Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 68–77.

[30] A. Faonio and L. Russo, "Mix-nets from re-randomizable and replayable cca-secure public-key encryption," in *International Conference on Security and Cryptography for Networks*. Springer, 2022, pp. 172–196.

[31] S. S. Chow, J. K. Liu, and D. S. Wong, "Robust receipt-free election system with ballot secrecy and verifiability." in *NDSS*, vol. 8, 2008, pp. 81–94.

[32] R. Kusters, T. Truderung, and A. Vogt, "Clash attacks on the verifiability of e-voting systems," in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 395–409.

[33] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE symposium on security and privacy*. IEEE, 2014, pp. 459–474.

[34] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology – EUROCRYPT 2016*, M. Fischlin and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 305–326.

[35] SCIPR-Lab, "libsnark : a c++ library for zk-snark proofs," 2014, https://github.com/scipr-lab/libsnark.

[36] M. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen, "Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity," in *Advances in Cryptology – ASIACRYPT 2016*, J. H. Cheon and T. Takagi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 191–219.

[37] A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. PAPAMANTHOU, R. Pass, S. ABHI, and E. SHI, "cøcø: A framework for building composable zero-knowledge proofs," Cryptology ePrint Archive, Report 2015/1093, 2015. http://eprint. iacr. org . . . , Tech. Rep., 2015.

[38] D. J. Bernstein, "Curve25519: New diffie-hellman speed records," in *Public Key Cryptography - PKC 2006*, M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 207–228.

[39] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

Appendix A. Security games

A.1. Nullifiable commitment

Hiding. In Figure 3, let the advantage of the hiding game be $\mathbf{Adv}^{\mathsf{Hiding}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) = \left|\Pr[\mathsf{Game}^{\mathsf{Hiding}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) = 1] - \frac{1}{2}\right|$.

$\underline{\mathsf{Game}^{\mathsf{Hiding}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda):}$

$(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}^*) \leftarrow \mathsf{Setup}(1^\lambda);$
$\mathsf{ck} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, b \leftarrow\$ \{0,1\});$
$(\mathsf{m}_0, \mathsf{m}_1, aux) \leftarrow \mathcal{A}(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}^*, \mathsf{ck});$
$c \leftarrow\$ \{0,1\}; r \leftarrow\$ \{0,1\}^\lambda;$
$\mathsf{cm} \leftarrow \mathsf{Commit}(\mathsf{ck}, \mathsf{m}_c; r);$
$c' \leftarrow \mathcal{A}(\mathsf{cm}, aux);$
return $c = c';$

Figure 3: Hiding game $\mathsf{Game}^{\mathsf{Hiding}}_{\mathcal{A},\Pi_{\mathsf{NC}}}$

Indistinguishability of keys (IK). From the game in Figure 4, let the advantage of IK game be $\mathbf{Adv}^{\mathsf{IK}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) = \left|\Pr[\mathsf{Game}^{\mathsf{IK}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) = 1] - \frac{1}{2}\right|$.

A.2. Secure e-voting systems

Ballot Privacy.
Voter Anonymity.

$\underline{\mathsf{Game}^{\mathsf{IK}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda):}$

$(\mathsf{mpk}, \mathsf{msk}, \mathsf{ck}^*) \leftarrow \mathsf{Setup}(1^\lambda);$
$\mathsf{ck}_0 \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, b \leftarrow\$ \{0,1\});$
$\mathsf{ck}_1 \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, b' \leftarrow\$ \{0,1\});$
$(\mathsf{m}, aux) \leftarrow \mathcal{A}(\mathsf{mpk}, \mathsf{ck}^*, \mathsf{ck}_0, \mathsf{ck}_1);$
$c \leftarrow\$ \{0,1\}; r \leftarrow\$ \{0,1\}^\lambda;$
$\mathsf{cm} \leftarrow \mathsf{Commit}(\mathsf{ck}_c, \mathsf{m}; r);$
$c' \leftarrow \mathcal{A}(\mathsf{cm}, aux);$
return $c = c';$

Figure 4: IK game $\mathsf{Game}^{\mathsf{IK}}_{\mathcal{A},\Pi_{\mathsf{NC}}}$

$\underline{\mathsf{Game}^{BP}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda):}$

$(\mathsf{PP}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda);$
$(\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}(\mathsf{PP}); \quad b \leftarrow\$ \{0,1\};$
$B_b \leftarrow \mathcal{V}^{\mathsf{Vote}(\mathsf{PP},(id, sk_{id}, pk_{id}, \mathsf{ck}, \mathsf{m}_b))};$
$b' \leftarrow \mathcal{A}(\mathsf{PP}, B_b);$
return $b = b';$

Figure 5: Ballot privacy game $\mathsf{Game}^{BP}_{\mathcal{A},\Pi_{\mathsf{Vote}}}$

Appendix B. Security proofs for Algorithm 1

We prove the security of Π_{NC} under the Discrete Logarithm(DL) assumption and the Decisional Diffie-Hellman(DDH) assumption. The DL assumption asserts that given a cyclic group $\mathbb{G}$ of prime order q with generator g, for randomly chosen element $h \in \mathbb{G}$, it is computationally infeasible to determine the unique integer x such that $h = g^x \in \mathbb{G}$ in polynomial time. The DDH assumption is also defined over a cyclic group $\mathbb{G}$. It states that the distinguishing between the tuple (g^x, g^y, g^{xy}) and a random tuple (g^x, g^y, g^z) is computationally hard for randomly chosen $x, y, z \leftarrow\$ \mathbb{Z}_q$.

Theorem 2.1. If the DL assumption and DDH assumption hold in the elliptic curve group $\mathbb{G}$, then for all probabilistic polynomial time adversaries $\mathcal{A}$, Π_{NC} in Construction 1, satisfies hiding, binding, nullifiability, indistinguishability of keys, key deniability, and homomorphism.

Proof. **Hiding**. Since a nullifiable commitment scheme has a real commitment and a fake commitment, we can define the following games from the following cases: given two commitments, a hiding adversary $\mathcal{A}$ distinguishing 1) fake commitments, 2) a fake commitment and a real commitment 3) real commitments.

1) $\mathsf{Game}^{\mathsf{Hiding-FF}}$: This is a hiding game where the challenge commitments are always fake. The advantage of $\mathcal{A}$ guessing a message from the commitments is $\mathbf{Adv}^{\mathsf{Hiding-FF}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) = \left|\Pr[\mathsf{Game}^{\mathsf{Hiding-FF}}(1^\lambda) = 1] - \frac{1}{2}\right|$.

$\underline{\mathsf{Game}^{VA}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda):}$

$(\mathsf{PP},\mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda);$
$(\mathbf{sk_{id}},\mathbf{pk_{id}},\mathbf{ck}) \leftarrow \mathcal{V}^{\mathsf{Register}(\mathsf{PP},\mathsf{SK},id)};$
$(\mathsf{m},(pk_{id,0},ck_0),(pk_{id,1},ck_1)) \leftarrow A(\mathsf{PP},\mathsf{SK},(\mathbf{pk_{id}},\mathbf{ck}));$
$b \leftarrow\$ \{0,1\}; \quad B_b \leftarrow \mathcal{V}^{\mathsf{Vote}(\mathsf{PP},(id,\mathsf{sk}_{id,b},pk_{id,b},\mathsf{ck}_b,\mathsf{m}))};$
$b' \leftarrow \mathcal{A}(\mathsf{PP},\mathsf{SK},B_b);$
return $b = b';$

Figure 6: Voter anonymity game $\mathsf{Game}^{VA}_{\mathcal{A},\Pi_{\mathsf{Vote}}}$

2) $\mathsf{Game}^{\mathsf{Hiding-FR}}$: $\mathsf{Game}^{\mathsf{Hiding-FR}}$ is a hiding game in which one of the commitments is fake and the other is real. The advantage of $\mathcal{A}$ guessing a message from the commitments is $\mathbf{Adv}^{\mathsf{Hiding-FR}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) = \left|\Pr[\mathsf{Game}^{\mathsf{Hiding-FR}}(1^\lambda)=1] - \frac{1}{2}\right|$.
3) $\mathsf{Game}^{\mathsf{Hiding-RR}}$: In this game, the challenge commitments are always real commitments. The advantage of $\mathcal{A}$ guessing a message from the commitments is $\mathbf{Adv}^{\mathsf{Hiding-RR}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) = \left|\Pr[\mathsf{Game}^{\mathsf{Hiding-RR}}(1^\lambda)=1] - \frac{1}{2}\right|$.

Then, we get, $\mathbf{Adv}^{\mathsf{Hiding}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) \leq \mathbf{Adv}^{\mathsf{Hiding-FF}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) + \mathbf{Adv}^{\mathsf{Hiding-FR}}_{\mathcal{A},\Pi_{\mathsf{NC}}} + \mathbf{Adv}^{\mathsf{Hiding-RR}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$.

We now show that all advantages on the RHS are negligible, thus concluding that $\mathbf{Adv}^{\mathsf{Hiding}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$ is negligible and Π_{NC} ensures hiding.

$\mathsf{Game}^{\mathsf{Hiding-FF}}$ is a game where the adversary $\mathcal{A}$ tries to guess a message from two fake commitments, $\mathsf{cm}_0 = \mathsf{Commit}(\mathsf{ck}_0,\mathsf{m}_0,r_0)$ and $\mathsf{cm}_1 = \mathsf{Commit}(\mathsf{ck}_1,\mathsf{m}_1,r_1)$. Since $\mathbb{G}$ is a cyclic group, we can map $g_2 = g_1^a, g_3 = g_1^c$. Suppose a message m_0 is committed using a fake key. The casting key is $\mathsf{ck}_0 = ((g_1,h_{1,0}),(g_2,h_{2,0})) = ((g_1,g_1^{x_0}),(g_1^a,g_1^{x_0\rho}))$. Then, we get $\mathsf{cm}_0 = (C_{1,0},C_{2,0}) = (g_1^{r_0}h_{1,0}^{\mathsf{m}_0},\ g_2^{r_0}h_{2,0}^{\mathsf{m}_0}) = (g_1^{r_0+x_0\mathsf{m}_0}, g_1^{ar_0+x_0\rho\mathsf{m}_0})$. From $C_{1,0} = g_1^{r_0}h_{1,0}^{\mathsf{m}_0} = g_1^{r_0+x_0\mathsf{m}_0}$, since r_0 is a random value in $\mathbb{Z}_q^*$, then $r_0 + x_0\mathsf{m}_0$, is uniformly distributed in $\mathbb{Z}_q^*$. Since the adversary cannot distinguish between $(g_1^{r_0},g_1^a,g_1^{r_0a})$ and $(g_1^{r_0},g_1^a,R)$ due to DDH where R is a random, the value $C_{2,0} = g_1^{ar_0+x_0\rho\mathsf{m}_0}$ also looks like a random element in $\mathbb{G}$.

The same technique can be used for cm_1. All elements are uniformly distributed in $\mathbb{G}$, the only probability for the adversary to win the game is through random guessing, $\frac{1}{2}$. Therefore, we can say $\mathbf{Adv}^{\mathsf{Hiding-FF}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) \leq \mathsf{negl}(\lambda)$.

$\mathsf{Game}^{\mathsf{Hiding-FR}}$ is a game where the adversary $\mathcal{A}$ tries to guess a message from a real commitment and a fake commitment, $\mathsf{cm}_0 = \mathsf{Commit}(\mathsf{ck}_0,\mathsf{m}_0,r_0)$ and $\mathsf{cm}_1 = \mathsf{Commit}(\mathsf{ck}_1,\mathsf{m}_1,r_1)$. Suppose encrypting a real commitment with a message m_0 using a real key. A real commitment key ck_0 is given as $((g_1,h_{1,0}),(g_2,h_{2,0}))$. We can replace $g_2 = g^a, g_3 = g^c$ where a,c are uniformly chosen in $\mathbb{Z}_q^*$. Then the real commitment cm_0 is given by $(C_{1,0},C_{2,0}) = (g_1^{r_0}h_{1,0}^{\mathsf{m}_0}, g_2^{r_0}h_{2,0}^{\mathsf{m}_0}) = (g_1^{r_0+x_0\mathsf{m}_0}, g_1^{ar_0+x_0\rho\mathsf{m}_0+c\mathsf{m}_0})$. Then, both $C_{1,0} = g_1^{r_0+x_0\mathsf{m}_0}$ and $C_{2,0} = g_1^{ar_0+x_0\rho\mathsf{m}_0+c\mathsf{m}_0}$ look like random elements in $\mathbb{G}$ due to DDH. As we have shown above, $(C_{1,1},C_{2,1})$also look like random elements in $\mathbb{G}$, we can say that a real commitment is indistinguishable from a fake commitment. Even the msk holder cannot distinguish between them.

Above, we prove that a fake commitment is indistinguishable from a real one. Since each commitment is indistinguishable, we have, $\mathbf{Adv}^{\mathsf{Hiding-FR}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) \leq \mathsf{negl}(\lambda)$.

$\mathsf{Game}^{\mathsf{Hiding-RR}}$ is a game in which the adversary $\mathcal{A}$ tries to guess a message from two real commitments, $\mathsf{cm}_0 = \mathsf{Commit}(\mathsf{ck}_0,\mathsf{m}_0,r_0)$ and $\mathsf{cm}_1 = \mathsf{Commit}(\mathsf{ck}_1,\mathsf{m}_1,r_1)$. Using the technique above, all real commitment looks like random elements in the elliptic curve group $\mathbb{G}$. Since all real commitments are indistinguishable from each other, even for the holder of the master secret key msk. We have, $\mathbf{Adv}^{\mathsf{Hiding-RR}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) \leq \mathsf{negl}(\lambda)$.

From above, we prove that all advantages $\mathbf{Adv}^{\mathsf{Hiding-FF}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$, $\mathbf{Adv}^{\mathsf{Hiding-FR}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$ and $\mathbf{Adv}^{\mathsf{Hiding-RR}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$ are negligible, we have that $\mathbf{Adv}^{\mathsf{Hiding}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) \leq \mathsf{negl}(\lambda)$.

Binding. We can apply a simple technique to prove binding in the Pedersen commitment scheme. A commitment is defined as $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck},\mathsf{m},r) = (g_1^r h_1^{\mathsf{m}}, g_2^r h_2{}^{\mathsf{m}})$. Using contradiction, suppose our scheme is not ensuring binding and we can compute m', r' such that $\mathsf{cm} = (g_1^{r'}h_1^{\mathsf{m}'}, g_2^{r'}h_2{}^{\mathsf{m}'})$ and $\mathsf{m} \neq \mathsf{m}'$ and $r \neq r'$. Then, we get $g_1^r h_1^{\mathsf{m}} = g_1^{r'}h_1^{\mathsf{m}'}$ and $g_2^r h_2{}^{\mathsf{m}} = g_2^{r'}h_2{}^{\mathsf{m}'}$. In the first equation, we can solve the discrete logarithm problem that $\log_{g_1}^{h_1} = (r-r')/(\mathsf{m}'-\mathsf{m})$. For the second equation, we can also solve the discrete logarithm problem that $\log_{g_2}^{h_2} = (r-r')/(\mathsf{m}'-\mathsf{m})$. These violate the DL assumption. Therefore, finding such $\mathsf{m},\mathsf{m}',r,r'$ that satisfy these conditions is infeasible, ensuring the binding of the commitment scheme.

Nullifiability. Given a master secret key msk, and a commitment $\mathsf{cm} = (g_1^r h_1^{\mathsf{m}}, g_2^r h_2^{\mathsf{m}})$, we can nullify a commitment and map it to another commitment cm^*. A manager has a master secret key and generates another commitment $\mathsf{cm}^* = \frac{C_2}{C_1^{\mathsf{msk}}}$. This commitment still ensures hiding and binding and can be opened with ck^* and a nullifiable message. From $\mathsf{cm}^* = \frac{C_2}{C_1^{\mathsf{msk}}} = \frac{g_2^r h_2^{\mathsf{m}}}{(g_1^r h_1^{\mathsf{m}})^\rho} = (\frac{g_2}{g_1^\rho})^r \cdot g_3^{b\cdot\mathsf{m}}$. Simply, r is uniformly random, and cm^* is hiding commitment including the manager. Also, it is a form of the Pedersen commitment, binding is straightforward. From cm^*, we show $\mathsf{Open}(\mathsf{ck}^*,\mathsf{cm}^*,b\cdot\mathsf{m},r) = 1$. Since this is a commitment scheme, later the committer will provide or the manager will find the original message. Then, the manager can open the commitment from $\mathsf{ck}^* = (g^*,h^*) = (g_2/g_1^\rho, g_3)$ by checking if $\mathsf{cm}^* = g^{*r}h^{*b\cdot\mathsf{m}}$. Since $\mathsf{cm}^* = (\frac{g_2}{g_1^\rho})^r \cdot g_3^{b\cdot\mathsf{m}} = g^{*r}h^{*b\cdot\mathsf{m}}$, Open outputs 1.

Indistinguishability of keys (IK). Consider a game in which, given a commitment cm, an IK adversary $\mathcal{A}$ attempts to distinguish which commitment key has been used. Given that a nullifiable commitment scheme possesses both a real commitment key and a fake commitment key, we can define the games from the following scenarios: given a commit-

ment, an IK adversary $\mathcal{A}$ distinguishing which commitment key was used between 1) two fake keys, 2) a fake key and a real key 3) two real keys.

- $\mathsf{Game}^{\mathsf{IK-FF}}$: $\mathsf{Game}^{\mathsf{IK-FF}}$ is a game where a commitment is generated with a randomly chosen commitment key among two fake keys. $\mathcal{A}$ tries to distinguish which key has been used. Let $\mathbf{Adv}^{IK-FF}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) = \left|\Pr[\mathsf{Game}^{\mathsf{IK-FF}}(1^\lambda)=1] - \frac{1}{2}\right|$ be the advantage.
- $\mathsf{Game}^{\mathsf{IK-FR}}$: $\mathsf{Game}^{\mathsf{IK-FR}}$ is a game where given a commitment generated with a randomly chosen commitment key among one fake key and one real key to $\mathcal{A}$. $\mathcal{A}$ tries to distinguish which key has been used. Let the advantage be $\mathbf{Adv}^{IK-FR}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) = \left|\Pr[\mathsf{Game}^{\mathsf{IK-FR}}(1^\lambda)=1] - \frac{1}{2}\right|$.
- $\mathsf{Game}^{\mathsf{IK-RR}}$: $\mathsf{Game}^{\mathsf{IK-RR}}$ is a game where a commitment is generated with a randomly chosen commitment key among two real keys. $\mathcal{A}$ tries to distinguish which key has been used. Let $\mathbf{Adv}^{IK-RR}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) = \left|\Pr[\mathsf{Game}^{\mathsf{IK-RR}}(1^\lambda)=1] - \frac{1}{2}\right|$ be the advantage.

Then, we get $\mathbf{Adv}^{\mathsf{IK}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) \leq \mathbf{Adv}^{\mathsf{IK-FF}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) + \mathbf{Adv}^{\mathsf{IK-FR}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda) + \mathbf{Adv}^{\mathsf{IK-RR}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$.

In the game $\mathsf{Game}^{\mathsf{IK-FF}}$, $\mathcal{A}$ is provided two fake commitment keys, $(h_{1,0}, h_{2,0})$ and $(h_{1,1}, h_{2,1})$, $\mathcal{A}$ then submits two keys along with a message m. A commitment $\mathsf{cm} = (C_{1,c}, C_{2,c})$ is created using a randomly selected key $\mathsf{ck}_c = (h_{1,c}, h_{2,c})$ where $c \in \{0,1\}$.

Consider the element $C_{1,c}$ of the commitment. Regardless of c, $C_1 = g_1^r {h_0}^{\mathsf{m}}$ becomes random since g_1^r is random. In addition, both $C_{2,0} = g_1^{ar} {h_{2,0}}^{\mathsf{m}}$, and $C_{2,1} = g_1^{ar} {h_{2,1}}^{\mathsf{m}}$ are random since g_1^{ar} is indistinguishable from random due to DDH for given $g_2 = g_1^a$, and g_1^r. Therefore, the probability that $\mathsf{Game}^{\mathsf{IK-FF}}(1^\lambda) = 1$ is equivalent to random guessing, which is $\frac{1}{2}$. We have $\mathbf{Adv}^{IK-FF}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$ is negligible.

In the game $\mathsf{Game}^{\mathsf{IK-FR}}$, suppose ck_0 is a fake key and ck_1 is a real key. Using the same technique as above, we have $C_{2,c} = g_2^r h_{2,c}^{\mathsf{m}} = g_1^{ar_0} h_{2,c}^{\mathsf{m}}$. Since g_1^{ar} is random due to DDH, $C_{2,c}$ is random. $\Pr[\mathsf{Game}^{\mathsf{IK-FR}}(1^\lambda) = 1] = \frac{1}{2}$, and hence, $\mathbf{Adv}^{IK-FR}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$ is negligible.

In the game $\mathsf{Game}^{\mathsf{IK-RR}}$ where both ck_0 and ck_1 are all real keys, due to DDH, the adversary cannot distinguish the commitment from random without knowing opening key r. Then, we can get $\Pr[\mathsf{Game}^{\mathsf{IK-RR}}(1^\lambda) = 1] = \frac{1}{2}$, implying that $\mathbf{Adv}^{IK-RR}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$ is negligible.

All advantages in the RHS are negligible, therefore, $\mathbf{Adv}^{\mathsf{IK}}_{\mathcal{A},\Pi_{\mathsf{NC}}}(1^\lambda)$ is negligible.

Key deniability. From a simulator $\mathsf{SimKeyProve}$ that outputs an accepting proof π'_{ck} without a master secret key msk.

From π'_{key}, we have $p' = h_1^{k'}/(h_2/g_3^{b'})^{c'}$. Since k' and c' are randomly chosen from $\mathbb{Z}_q^*$, $g_3^{b'c'}$ can be represented to h_1^x for any random r. We can say that $p' = h_1^r$ where r is random. Comparing to the real proof $\pi_{key} = (h_1^t)$, t is also a random. Each element in the real proof set (h_1^t, k, c, b) has the same distribution as ones in the simulated proof set (h_1^r, k', c', b'). The two proofs are indistinguishable and $\mathsf{SimKeyProve}$ can always outputs 1, since $h_1^{k'} = p'(h_2/g^{b'})^{c'}$. For a fake commitment key $\mathsf{ck} = ((g_1,h_1),(g_2,h_2)) = ((g_1,h_1),(g_2,h_1^\rho)$ and its proof π_{ck}, we can also simulate a proof that ck is a real key. From $\mathsf{SimKeyProve}(\mathsf{mpk},\mathsf{ck},b'=1)$, we set random k', c' and generate $p' = h_1^{k'}/(h_2/g_3^{b'})^{c'} = h_1^{k'}/(h_1^\rho/g_3)^{c'} = h_1^r$ for some random r. This simulated proof is indistinguishable from real proof. And, p' can pass the verification $h_1^k = p(h_2/g_3^b)^c$. If the committer claims this is a real commitment key, $b=1$ in the verification. Then, we can get $h_1^{k'} = (h_1^{k'}/(h_1^\rho/g_3)^{c'})(h_2/g_3)^{c'} = h_1^{k'}$, passing the verification. The opposite case is straightforward.

Homomorphism. Consider two commitments where $h = g_1^x$ $\mathsf{cm}_0 = (g_1^{r_0} h_{1,0}^{\mathsf{m}_0}, g_2^{r_0} h_{2,0}^{\mathsf{m}_0}) = (g_1^{r_0+x_0\mathsf{m}_0}, g_2^{r_0} g_1^{x_0\rho\mathsf{m}_0} g_3^{b_0\mathsf{m}_0})$ and $\mathsf{cm}_1 = (g_1^{r_1} h_{1,1}^{\mathsf{m}_1}, g_2^{r_1} h_{2,1}^{\mathsf{m}_1}) = (g_1^{r_1+x_1\mathsf{m}_1}, g_2^{r_1} g_1^{x_1\rho\mathsf{m}_1} g_3^{b_1\mathsf{m}_1})$. We can aggregate two commitments $\sum \mathsf{cm} = \mathsf{cm}_0 + \mathsf{cm}_1 = (g_1^{r_0+x_0\mathsf{m}_0+r_1+x_1\mathsf{m}_1}, g_2^{r_0+r_1} g_1^{x_0\rho\mathsf{m}_0+x_1\rho\mathsf{m}_1} g_3^{b_0\mathsf{m}_0+b_1\mathsf{m}_1})$. Then, a manager can nullify commitments and map to new commitments $\mathsf{cm}^* = \mathsf{Nullify}(\mathsf{msk}, \sum \mathsf{cm}) = C_2/(C_1^{\mathsf{msk}}) = \frac{g_2^{r_0+r_1} g_1^{x_0\rho\mathsf{m}_0+x_1\rho\mathsf{m}_1} g_3^{b_0\mathsf{m}_0+b_1\mathsf{m}_1}}{(g_1^{r_0+x_0\mathsf{m}_0+r_1+x_1\mathsf{m}_1})^\rho} = (g_2/g_1^\rho)^{r_0+r_1} g_3^{b_0\mathsf{m}_0+b_1\mathsf{m}_1}$. We can now open cm^* with $\mathsf{ck}^*, \sum b \cdot \mathsf{m}, \sum r$, $\mathsf{Open}(\mathsf{ck}^*, \mathsf{cm}^*, \sum b\cdot\mathsf{m}, \sum r) = (g_2/g_1^\rho)^{\sum r} g_3^{\sum b\cdot\mathsf{m}} = \mathsf{cm}^*$.

Thus, we can infer that homomorphism always holds. □

Appendix C. Security proofs for Verifiability

We adapt Cortier's definition of verifiability [9] to ensure the integrity of *zkVoting*. In a nutshell, a voting scheme is verifiable if the election result reflects the votes of 1) all honest voters who have verified that their ballots appear correctly on the bulletin board; 2) a subset of honest voters who did not verify their ballots; and 3) at most n_c voters under the control of an adversary. Notably, in our scheme, a registrar is not trusted to ensure verifiability, which is known as strong verifiability meaning that both individual verifiability and universal verifiability are ensured. In addition, in scenarios such as clash attacks [32] where two voters vote for the same candidate, malicious voting devices or authority could deceive the voters with the same ballot. Two voters do not realize they are looking at the same ballot. This can satisfy both individual verifiability and universal verifiability but is not sufficient for E2E verifiability. Malicious authority could replace one ballot in an undetectable way. Therefore, voter names or pseudonyms should be attached to the ballot to satisfy E2E verifiability [10]. Our system attaches a serial number to ballots and allows the voter to trace their ballot through the entire voting system, ensuring each ballot is unique.

The verifiability experiment is formally illustrated in Figure 7. We use $\mathbb{U}$ to denote the set of $(id, \mathsf{pk}_{id}, \mathsf{sk}_{id}, \mathsf{ck}_{id}, B)$ pairs and C to denote the set of $(id, \mathsf{pk}_{id}, \mathsf{ck}_{id}, B)$ where n_c be the number of the set of corrupted voters. Let $H = \{(\mathsf{pk}_i^h, \mathsf{ck}_i^h, \mathsf{m}_i^h, *)\}_{i=1}^{n_h}$ correspond to voters who have checked that their ballots

$\underline{\mathsf{Exp}^{ver}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda):}$

$(\mathsf{PP},\mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda)$;

$(\mathcal{BB},\mathcal{T}) \leftarrow \mathcal{A}^{\mathsf{O}}(\mathsf{SK})$;

if $\mathsf{VerifyTally}(\mathsf{PP},\mathcal{T}) = 0$ **then return** 0;

if $\mathsf{m}_{tally} = 0$ **then return** 0;

$H = \{(\mathsf{pk}_i^h, \mathsf{ck}_i^h, \mathsf{m}_i^h, *)\}_{i=1}^{n_h}$; $H' = \{(\mathsf{pk}_i^{h'}, \mathsf{ck}_i^{h'}, \mathsf{m}_i^{h'}, *)\}_{i=1}^{n_{h'}}$;

if $\exists\{(\mathsf{pk}_i^c, \mathsf{ck}_i^c, \mathsf{m}_i^c, *)\}_{i=1}^{n_c} \subseteq C$,

$\exists\{(\mathsf{pk}_i^{h'}, \mathsf{ck}_i^{h'}, \mathsf{m}_i^{h'}, *)\}_{i=1}^{n_{h'}} \subseteq H'$

$s.t.\ \mathsf{m}_{tally} = \rho(\{(\mathsf{pk}_i^h, \mathsf{ck}_i^h, \mathsf{m}_i^h, B_i^h)\}_{i=1}^{n_h} \cup$

$\{(\mathsf{pk}_i^{h'}, \mathsf{ck}_i^{h'}, \mathsf{m}_i^{h'}, B_i^{h'})\}_{i=1}^{n_{h'}} \cup \{(\mathsf{pk}_i^c, \mathsf{ck}_i^c, \mathsf{m}_i^c, B_i^c)\}_{i=1}^{n_c})$ **then return** 0;

else return 1;

$\underline{\mathsf{O}^{\mathsf{Corrupt}}(id):}$

if $(id,*,*,*,*) \notin \mathbb{U}$ **then return** 0;

$C \Leftarrow \{(id, \mathsf{pk}_{id}, \mathsf{ck}_{id}, *)\}$;

return $(\mathsf{pk}_{id}, \mathsf{sk}_{id}, \mathsf{ck}_{id})$;

$\underline{\mathsf{O}^{\mathsf{Vote}}(id,\mathsf{m}):}$

if $(id,*,*,*,*) \notin \mathbb{U}$ *or* $(id,*,*,*) \in C$ **then return** 0;

$(k,B) \leftarrow \mathsf{Vote}(\mathsf{PP}, \mathsf{ck}_{id}, \mathbf{CKlist}, \mathsf{pk}_{id}, \mathsf{sk}_{id}, \mathsf{m})$;

$H \cup H' \Leftarrow (\mathsf{pk}_{id}, \mathsf{ck}_{id}, \mathsf{m}, B)$;

return (k,B);

Figure 7: Verifiability experiment $\mathsf{Exp}^{ver}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda)$

will be counted and $H' = \{(\mathsf{pk}_i^{h'}, \mathsf{ck}_i^{h'}, \mathsf{m}_i^{h'}, *)\}_{i=1}^{n_{h'}}$ correspond to voters that have not checked their ballots. We adapt the result function ρ for the election result. The adversary $\mathcal{A}$ takes as input the authority's secret key SK, and can query the following oracles:

- $\mathsf{O}^{\mathsf{Corrupt}}(id)$, which allows the adversary $\mathcal{A}$ to corrupt a voter id.
- $\mathsf{O}^{\mathsf{Vote}}(id,\mathsf{m})$, which allows the adversary $\mathcal{A}$ to let an honest voter id cast a vote for m.

Note that $\mathcal{A}$ does not need the $\mathsf{O}^{\mathsf{Register}}$ oracle since he controls the authority and thus can register arbitrarily. The adversary wins if the result $\mathcal{T}$ verifies but violates any of the following conditions: 1) for each voter that has checked his ballot, the ballot is counted; 2) a subset of votes by honest voters that did not check their ballots are counted; 3) at most n_c corrupted votes are counted.

Definition C.1. We say that Π_{Vote} is E2E verifiable if the advantage of any PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{Exp}^{ver}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda) = 1]$ is negligible and the ballot contains a unique identifier.

Theorem C.1. Assume that the underlying ZKPs are complete, sound, and zero-knowledge, and the security of the cryptographic hash function. A voting system Π_{Vote} provides E2E verifiability.

Proof. Let the adversary $\mathcal{A}$ output a set of votes, and the tally result $\mathcal{T}$ including the proof. Let $\{B_1,\cdots,B_{n_b}\}$ be the valid ballots on the $\mathcal{BB}$. By the soundness of the ZKP relation $\mathcal{R}_{tally}$ and the homomorphism and nullifiability property of the nullifiable commitment scheme, we can conclude that $\mathcal{T}$ is the correct tally of $\{B_1,\cdots,B_{n_b}\}$ if $\mathsf{VerifyTally}(\mathsf{PK},\mathcal{T})$ returns 1.

Now, we prove that for each ballot $B_i \in \mathcal{BB}$, it is one of the following sets:

- $H = \{(\mathsf{pk}_i^h, \mathsf{ck}_i^h, \mathsf{m}_i^h, *)\}_{i=1}^{n_h}$, the votes of the honest voters who have checked their ballots.
- $H' = \{(\mathsf{pk}_i^{h'}, \mathsf{ck}_i^{h'}, \mathsf{m}_i^{h'}, *)\}_{i=1}^{n_{h'}}$, the votes of the honest voters who have not checked their ballots.
- $C = \{(\mathsf{pk}_i^c, \mathsf{ck}_i^c, \mathsf{m}_i^c, *)\}_{i=1}^{n_c}$, the votes of the corrupted voters.

Due to the cryptographic hash function H, it is impossible for an adversary to forge a valid secret key sk_{id} that corresponds to a given public key pk_{id} where $\mathsf{pk}_{id} = H(\mathsf{sk}_{id})$. The ZKP relation $\mathcal{R}_{total}$ ensures that the registrar can only generate at most n_v number of real casting keys. The soundness of the ZKP relation $\mathcal{R}_{ck}$ ensures that the casting keys cks are formed correctly as the voter required. Another ZKP relation, $\mathcal{R}_{vote}$, verifies the membership of each voter's casting key in the system. As a result, the adversary cannot produce a valid proof for $\mathcal{R}_{vote}$. Therefore, any ballot that does not align with the ballots of honest voters will be identified and excluded from the count. This rigorous verification process ensures that only ballots associated with verified casting keys and legitimate voter identities are tallied.

We now prove that the adversary cannot drop the votes of the honest voters who have checked their ballots. By the soundness of the ZKP relation $\mathcal{R}_{vote}$, the adversary cannot cast a valid ballot created with a ck for pk_{id} due to the cryptographic hash function H, it is impossible for an adversary to forge a valid secret key sk_{id} that corresponds to a given public key pk_{id} where $\mathsf{pk}_{id} = H(\mathsf{sk}_{id})$. We can have that if the result m_{tally} verifies, then it must correspond to the result of the tally function ρ computed on all the votes by honest voters who checked their ballots, at most n_c votes cast by corrupted voters, and a subset of votes cast by honest voters who did not check their ballots. We can observe that $\mathsf{Exp}^{ver}_{\mathcal{A},\Pi_{\mathsf{Vote}}}(1^\lambda) = 1]$ is negligible and each ballot contains a unique serial number, therefore, Π_{Vote} ensures E2E verifiability.

□