Haechi: Simple Commitment-based Keyless In-person Verifiable Elections

Jiwon Kim[1], Michael Naehrig[2], Bharath Namboothiry[3], Olivier Pereira[4], and Josh Benaloh[2]

[1]University of Michigan
[2]Microsoft Research
[3]University of Pennsylvania
[4]UCLouvain

Abstract

Verifiable election systems allow voters to verify that their secret ballots were accurately recorded and counted, while enabling anyone to verify the announced tallies. However, two practical obstacles stand between this guarantee and actual verification. First, public election records can be prohibitively large: for an election with one million ballots, deployed systems may publish more than 100GB of cryptographic data. Second, most existing systems protect ballot privacy through encryption, introducing key management—usually one of the most cumbersome and error-prone components of election administration.

We present Haechi, a keyless verifiable election scheme for in-person voting, based on cryptographic vector commitments rather than encryption. At the core of Haechi is a new zero-knowledge proof of ballot well-formedness based on Compressed Σ-protocol theory. Our proof scheme proves the validity of an entire ballot using only a logarithmic number of group elements in the number of the ballot options. This allows all of a voter's selections to be encoded into a single vector commitment.

With this approach, Haechi reduces the data required for public verification by more than an order of magnitude compared to prior in-person verifiable election systems. Haechi thereby makes verifications for large-scale election considerably more practical. We also give a post-quantum instantiation of Haechi based on lattice commitments, which preserves the same keyless workflow at the cost of larger election records.

Contents

1 Introduction **4**

2 Related Work **5**

3 Preliminaries **6**

4 Overview **7**
4.1 Threat Model & Goals . . . 7
4.2 Overview of Procedure . . . 8

5 Commitment-Based Verifiable Elections **8**

6 Proof of Ballot Well-Formedness **11**

7 Security and Integrity **14**
7.1 Ballot Privacy . . . 14
7.2 Cast-as-intended Verifiability . . . 15
7.3 Recorded-as-cast Verifiability . . . 16
7.4 Tallied-as-recorded Verifiability . . . 17

8 Implementation and Evaluation **18**

9 Post-Quantum Instantiation of Haechi **21**
9.1 Background . . . 21
9.2 The commitment scheme . . . 22
9.3 Encoding ballots . . . 23
9.4 Proving well-formedness . . . 24
9.5 Aggregation: proving the tally . . . 25
9.6 Parameters and sizes . . . 28

10 Discussion & Future Work **30**

A Tally Correctness Proof **35**

B Proof of Theorem 1 **37**

C Full Protocol of Π_1^{comp} **42**

D Proof of Ballot Well-formedness for Various Election Rules **43**
D.1 Approval Voting . . . 43
D.2 Cumulative Voting . . . 46

E Security Analysis of the Amortized Proof Π_1^{AM} **49**

F Generators for Vector Commitments **50**
F.1 Finite Field Groups . . . 50
F.2 Elliptic Curve Groups . . . 51

G Proof Preliminaries **52**

H Full Proof of Ballot Privacy (Theorem 2) **53**

I Full Proof of Cast-as-Intended Verifiability (Theorem 3) **58**

J Full Proof of Tallied-as-Recorded Verifiability (Theorem 5) **58**

K Receipt-freeness of Haechi **63**

1 Introduction

End-to-end verifiable elections enable public verifiability by publishing cryptographic evidence of the integrity of the election tallies. Individual voters can verify that their ballots were cast as they intended and recorded on the bulletin board correctly, while *any* member of the public can verify that the announced tally result is correctly computed from the published record—all without compromising ballot privacy.

A long line of work [1,12,13,18,21,23,25,26,29,31,35,40,43,44,52] has proposed various approaches with these properties. Several of them have progressed beyond prototypes and have been used in real elections. Helios [1] and Belenios [29] have been used in numerous university and professional-association elections; Scantegrity II [25] was used in 2009 and 2011 in municipal elections in Takoma Park, Maryland; and vVote and ElectionGuard have been deployed in conjunction with conventional voting systems in Australian and US public elections [18,33].

Despite this extensive body of work and deployment, existing systems suffer from two important practical limitations. First, the size of the election record is a critical obstacle to actual election verifications. For instance, the election record in the current version of Helios [1], used by the International Association for Cryptologic Research (IACR), requires 1664 bytes/candidate in approval voting (the ballot format that results in the shortest ballots) for each voter, which amounts to 166.4GB of data for an election with 100 candidates and a million ballots. The use of early versions of ElectionGuard for Risk Limiting Audits resulted in election records of about 1TB for a million voters [16], which was a motivation for the modification of proof formats in ElectionGuard 2.0 [18].

The second limitation is key management. Many recent systems, for both in-person and remote voting (e.g. internet voting), encrypt ballots and therefore require secure generation, distribution, and storage of keys. These systems need to determine who manages the keys and how they are managed. This is not only an operational challenge but also one of the greatest challenges for system design and implementation: key management problems forced the IACR to re-run its 2025 election [45], leading to global coverage [54].

In this work, we present Haechi, a keyless in-person end-to-end verifiable election scheme with short election records: in-person voting remains, by far, the most common approach to government elections in the world.[1] Haechi relies on an electronic ballot-collection device (such as a precinct scanner) in the polling place that commits to each ballot it receives and gives the commitment to the corresponding voter. The device keeps running totals of the votes together with randomness that it generated to form commitments to each ballot. After voting ends, it announces the vote totals and proves in zero knowledge that the aggregate commitment opens to them. Anyone can verify the proof and, by the homomorphic property of commitments, check that the published tally result corresponds to the committed ballots.

While several verifiable voting systems based on perfectly hiding commitments have been proposed in the past (see Table 1 and §2), these systems did not bring any important benefits for ballot size. This is principally due to the difficulties of building zero-knowledge (ZK) proofs of ballot well-formedness (e.g., no ballot contains more votes than allowed) for which prior systems used one commitment and ZK proof per candidate per voter, resulting in a record size linear in the product of numbers of candidates and voters.

Haechi addresses this challenge with a new zero-knowledge proof for vector commitments based on Compressed Σ-Protocol Theory [5]. Our proof scheme allows all selections in a ballot to be packed into a single vector commitment and a single proof of ballot validity, whose size is logarithmic in the number of candidates. We further introduce an amortization technique that proves the well-formedness of all voters' ballots at essentially the cost of proof of a single ballot.

Our proof scheme can support various voting rules, including the most common cases of approval voting, selection of 1-out-of-n, but also less common cases like cumulative voting, in which the voter can share k votes among n candidates and possibly assign multiple votes to a single candidate.

We implement Haechi in Rust, instantiate it with elliptic-curve and finite-field groups, and evaluate its performance. For a 1-out-of-100 contest, and using identical groups, the size of a ballot record goes from 115,264 bytes in ElectionGuard [18] (instantiated by encryption), the state-of-the-art in-person voting scheme, down to 7,688 bytes with Haechi. Assuming 1 million voters, the election record is 115GB vs. 7.6GB. With amortizing proof techniques we introduce, this is further reduced to 576MB, which is nearly

[1] The International Institute for Democracy and Electoral Assistance surveys only 32 out of 204 countries offering in-country postal voting to some or all voters [9, p. 160], and only 12 out of 178 countries offering remote electronic voting to some or all voters [9, p. 137].

200× improvement from the previous work.

Summary of our contributions.

- We construct Haechi, the first in-person verifiable election scheme with *homomorphic vector* commitments. We show a construction with Pedersen vector commitments for the most common election rule (choose 1 option).
- Along the way, we introduce a new approach to zero-knowledge proofs of ballot well-formedness based on Compressed-Σ protocols. Unlike most prior approaches that prove each selection independently, our proof jointly enforces the well-formedness of the entire selection vector, leading to substantial size improvements.
- We implement and evaluate Haechi in the Rust. For an election with 100 candidates and 1M voters, Haechi can improve the bulletin board size by 200× over prior work.
- We formalize ballot privacy for in-person verifiable elections. We also provide a rigorous analysis of Haechi's security and integrity.
- We show a post-quantum instantiation of Haechi based on BDLOP commitments to obtain verifiability (and privacy) against quantum adversaries, where the soundness of the Fiat–Shamir-compiled ballot proof is conjectural in the quantum random-oracle model. The construction and analysis are given in §9.

2 Related Work

Verifiable voting systems based on commitments. The first voting protocol relying on perfectly hiding commitments is probably the scheme of Cramer et al. from 1996 [31]. Pedersen commitments to the votes are published on a bulletin board, while the votes and commitment randomness are sent, using verifiable secret sharing, to a set of authorities (relying on PKI). Cuvelier et al. propose a variation to use a threshold encryption mechanism instead of secret shareing [34].

Moran and Naor also use Pedersen commitments to construct a universally verifiable voting scheme with everlasting privacy in the context of in-person voting on ballot marking devices (BMDs) [52]. At the end of the voting phase, the list of committed ballots is verifiably mixed, and the tally is obtained by opening the mixed commitments. By adopting the traditional mixnet approach, the BMD avoids the challenge of proving the well-formedness of the committed votes during the voting phase, but must support the burden of running a verifiable shuffle. This raises its own coercion concerns when the set of voting options is large, as full and possibly unique votes are released in the clear.

The DRE-I and DRE-IP systems of Hao et al. [40,53] are in-person voting schemes, in which homomorphic tallying is performed on a direct recording device (DRE). Each choice on the ballot is encoded as a separate Pedersen-like or ElGamal-like commitment, with its own proof of knowledge of a valid opening. As a result, the size of the ballot records grows linearly with the number of candidates, instead of logarithmically in our case for the compressed proof.

Modern ZK proof techniques for voting. ZK proofs are the standard tool used in verifiable elections to demonstrate that ballots are valid, i.e. a committed or encrypted vote is within a list of valid options . The traditional approach is the secret-sharing based solution of Cramer-Damgård-Schoenmakers (CDS) [30] disjunctive proof (see, among many others, [1, 18, 29, 31, 32]). However, the size of the proofs grows linearly with the number of options for each question, and with the number of questions on the ballot.

There have been several recent explorations of the use of modern sublinear ZK proof techniques for proving ballot validity. In Kryvos by Huber et al. [43], constant-size proofs based on the Groth16 [39] approach are used for proving the ballot well-formedness. While constant-size proofs are appealing, this approach assumes trusted proof setup, which is generally not desirable in verifiable elections where one should not need to trust any entity for the validity of the election outcome, and with demanding computational constraints: a computation time of 9 seconds is reported for proving the validity of a vote encoded on a Pedersen vector commitment of width 100, which may just be prohibitive in practice. PQKryvos [44], a recent extension of the Kryvos remote voting system to the PQ setting, relies on the generic Ligero proof system [4] to demonstrate the validity of elections in a tally-hiding way: the Ligero approach leads to faster computation at a cost of slower verification and proofs with a size in $O(\sqrt{|C|})$, where $|C|$ is the size of the proof circuit.

Scheme	Setting	Ballot primitive	Ballot representation	Ballot well-formedness proof mechanism	Proof size per voter	Assumptions
Cramer et al. [31]	Unspecified	Pedersen + PKE	$O(n \cdot t)\|\mathbb{G}\| + O(n \cdot t)\|\mathbb{C}\|$	CDS [30]	$O(n)\|\mathbb{F}\|$	DLOG, ROM
Helios [1]	Remote	Exponential ElGamal	$O(n)\|\mathbb{G}\|$	CDS	$O(n)\|\mathbb{F}\|$	DDH, ROM
EVOLVE [35]	Remote	BDOP [11] + PKE	$O(n \cdot t)\|\mathbb{R}\| + O(n \cdot t)\|\mathbb{C}\|$	CDS-style Lattice OR-proof	$O(n^2)\|\mathbb{R}\|$	MSIS, MLWE, ROM
Belenios [29]	Remote	Exponential ElGamal	$O(n)\|\mathbb{G}\|$	CDS	$O(n)\|\mathbb{F}\|$	DDH, ROM
Kryvos* [43]	Remote	Pedersen VC + PKE	$O(p \cdot t)\|\mathbb{G}\| + O(t)\|\mathbb{C}\|$	Groth16	$O(1)\|2\mathbb{G}_1 + 1\mathbb{G}_2\|$	DLOG
PQKryvos* [44]	Remote	BDLOP [10] + ML-KEM	$O(p \cdot t)\|\mathbb{R}\| + O(t)\|\mathbb{C}\|$	Ligero	Circuit-dependent	MSIS, MLWE, QROM
Prêt à Voter† [26]	In-person	Onion-style PKE	$O(n \cdot s)\|\mathbb{C}\|$	-	-	DDH, ROM
Moran-Naor† [52]	In-person	Pedersen	$O(n \cdot s)\|\mathbb{G}\|$	-	-	DLOG, ROM
Bingo† [23]	In-person	Pedersen	$O(n \cdot s)\|\mathbb{G}\|$	-	-	DLOG
Scantegrity II† [25]	In-person	Hash	$O(n \cdot s)\|\mathbb{C}\|$	-	$O(n)$	ROM
Wombat† [13]	In-person	ElGamal	$O(s)\|\mathbb{G}\|$	-	-	DDH, ROM
STAR-Vote [12]	In-person	Exponential ElGamal	$O(n)\|\mathbb{G}\|$	CDS	$O(n)\|\mathbb{F}\|$	DDH, ROM
DRE-i [40], DRE-ip [53]	In-person & remote	Pedersen-variant	$O(n)\|\mathbb{G}\|$	CDS	$O(n)\|\mathbb{F}\|$	DLOG, DDH, RO
ElectionGuard [18]	In-person	Exponential ElGamal	$O(n)\|\mathbb{G}\|$	CDS	$O(n)\|\mathbb{F}\|$	DDH, ROM
Haechi	In-person	Pedersen VC	$O(1)\|\mathbb{G}\|$	(Compressed) Σ-protocol	$O(\log n)\|\mathbb{G}\|, O(n)\|\mathbb{F}\|$	DLOG, ROM
Amort-Haechi	In-person	Pedersen VC	$O(1)\|\mathbb{G}\|$	(Compressed) Σ-protocol with amortization	$O(1)\|\mathbb{G}\|, O(1)\|\mathbb{F}\|$	DLOG, ROM
PQ-Haechi (§9)	In-person	BDLOP	$O(n)\|\mathbb{R}\|$	Σ-protocol	$O(n)\|\mathbb{R}\|$	MSIS, MLWE, QROM

Mixnet-based systems (†). These systems obtain public assurance of ballots and tally from verifiable shuffles. Since they reveal the shuffled raw selections, they do not require publicly posted per-ballot well-formedness proofs. The bulletin board initially includes the ballot data from the cast record only. A mixnet additionally publishes one output shuffled ballot list and a proof of correct shuffling at every stage. Thus, for s shuffle stages, the election record transcript contains $O(s)$ times additional ballot-sized records per ballot.
Kryvos and PQKryvos(∗). If an encoded vote has dimension m and one vector commitment accommodates N entries, it requires $p = \lceil m/N \rceil$ packed commitments per tallier.

Table 1: Asymptotic comparison between prior work and this work. Haechi denotes the base protocol of this work. Amort-Haechi is Haechi with amortized proof scheme and PQ-Haechi is post-quantum instantiation of Haechi. A *ballot* consists of commitments, ciphertexts, and/or other choice-hiding objects published per voter. Here, n is the number of selectable options, t is the number of trustees, s denotes the number of shuffle stages of mixnet. We use $\mathbb{G}, \mathbb{F}, \mathbb{R}$ for group elements, field elements, and ring elements respectively; $\mathbb{C}$ denotes the size of a ciphertext or cryptographic objects for schemes that do not specify the instantiation.

Very few works applied logarithmic-size proof techniques to voting protocol in a way that does not require any trusted setup. Bulletproofs [24] outlines a strategy for a proof of shuffle of committed values, which Hoffman et al. [42] adapt to the shuffle of ciphertexts. Recently, Harrison and Haines [41] applied STARKS to produce proofs that a reported election result is consistent with the homomorphic aggregation of a set of encrypted votes that are only provided in a perfectly hiding committed form.

While these works focus primarily on proofs computed by talliers, our work focuses primarily on proofs of ballot well-formedness computed by vote-collection devices while interacting with voters.

3 Preliminaries

Let λ denote the security parameter. For a positive integer $n \in \mathbb{N}$, let $[n] = \{1, 2, \ldots, n\}$. We denote by $\mathbb{Z}_q$ the integers modulo q for $q \in \mathbb{Z}$. For a set S, $x \leftarrow S$ denotes sampling x uniformly at random from S.

Pedersen vector commitments. Let G be a cyclic group of prime order q. For a message vector $\mathbf{m}$ of length n and $r \in \{0,1\}^\lambda$ sampled uniformly at random,

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: generates $h, g_1, \cdots, g_n \in G$ independently such that discrete logarithm relations are verifiably unknown and outputs $\mathsf{pp} = (G, q, h, g_1, \ldots, g_n)$.
- $\mathsf{Com}(\mathsf{pp}, \mathbf{m}; r) \to c$: computes and outputs $c = h^r \prod_{i=1}^{n} g_i^{m_i}$
- $\mathsf{Open}(\mathsf{pp}, c, \mathbf{m}, r) \to \{0,1\}$ outputs 1 if $(\mathbf{m}, r)$ is a valid opening of c, and 0 otherwise.

Pedersen vector commitments are perfectly hiding and are computationally binding under the discrete-logarithm assumption. They are also additively homomorphic.

Sigma protocol. A Sigma protocol for a relation $\mathcal{R}$ defined over a statement x and witness w is a three message, public-coin interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$:

1. Given $(x, w) \in \mathcal{R}$, the prover samples private randomness ρ and sends a first message $a \leftarrow \mathcal{P}_1(x, w; \rho)$.
2. The verifier samples a challenge $c \leftarrow_\$ \mathcal{C}$ from the challenge space $\mathcal{C}$ and sends c to the prover.
3. The prover returns a response $z \leftarrow \mathcal{P}_2(x, w; \rho, c)$ and the verifier outputs $\mathcal{V}(x, a, c, z) \in \{0,1\}$.

A transcript (a, c, z) and is accepting if $\mathcal{V}(x, a, c, z) = 1$. A Sigma protocol satisfies:

- *Completeness:* for every $(x, w) \in \mathcal{R}$, a transcript from honest execution of Sigma protocol is accepting.
- *2-Special soundness:* from two accepting transcripts (a, c, z) and (a, c', z') with distinct challenges $c \neq c'$, an efficient extractor can recover a witness w such that $(x, w) \in \mathcal{R}$.
- *Special honest-verifier zero knowledge:* there exists an efficient simulator $\mathcal{S}$ that, given x and a challenge $c \in \mathcal{C}$, outputs an accepting transcript distributed identically to, or computationally indistinguishable from, an honest transcript.

4 Overview

Haechi considers an in-person device-centric election, where voters cast their ballots at designated official voting sites using dedicated voting devices such as ballot scanners. We assume that an election can have several contests where each contest may have a distinct set of candidates and election rules (e.g., vote for one option, vote for up to three options, or approval voting). Our scheme focuses on traditional non-preferential voting schemes, where voters select one (or more) candidates instead of ranking them. Our work focuses on the voting process itself; voter eligibility and limiting voters to one cast ballot each are achieved by publishing a list of voters who cast ballots—as is typical in U.S. jurisdictions.

We consider three groups of participants: voters, voting devices, and verifiers. A *voter* casts a ballot using a voting device. A *device* records each voter's ballot (not the plaintext votes) on a public bulletin board called the *election record*. *Verifiers* can download the election record and verify the election locally. Haechi is a publicly verifiable election scheme; any person—candidates, press, voters, and even non-voters—can be a verifier.

4.1 Threat Model & Goals

Voters may attempt to violate the election rules. Such behavior includes casting more votes than allowed for preferred candidates or assigning negative values to other candidates. Verifiers may also be malicious and try to learn voters' choices.

We assume that voting devices preserve ballot privacy: they neither store nor send voters' selections. Although it seems a strong assumption, it is standard in virtually all in-person voting systems. In systems that encrypt ballots, the device first learns the voters' selections and then encrypts them. Even under conventional device-centric elections without cryptography, voters typically reveal their votes to a scanner

or DRE. We aim to ensure ballot privacy after the (committed) ballots are revealed to the public bulletin board, but cannot protect the selections from a fully compromised device. Preventing such a malicious device from recording or leaking selections—e.g. through spyware, cameras, or covert communication channels—is outside the scope of this work. The voting device, however, may attempt to incorrectly record votes or report inaccurate tallies.

We define our security and integrity goals informally using the terminology in [19] and defer the formalism to §7.

- **Ballot Privacy** (against verifiers): No one can learn about the selections from the ballot in the bulletin board.
- **Receipt-freeness**: Voters must not be able to convince a third party (i.e. coercer) of the contents of their ballots—even if they want to.
- **Cast-as-intended verifiability**: At the time of voting, voters must be able to verify whether a device has provided commitments to their ballot that match their selections.
- **Recorded-as-cast verifiability**: Voters must be able to verify whether the provided ballot commitments have been altered in the election record.
- **Tallied-as-recorded verifiability**: Verifiers must be able to verify whether every recorded ballot is correctly counted in the tally.

Additionally, we ensure ballots follow the election rule via proof of ballot well-formedness.

4.2 Overview of Procedure

Before a voting procedure begins, the device is loaded with the election setting from a manifest file and an empty election record. When voters make selections on a device (or scan their paper-ballots on the device), the device first prepares a *ballot*, a *proof of ballot well-formedness*, and a *confirmation code*. If a voter's selections do not satisfy the election rules, the device will fail to generate the well-formedness proof. If the proof is successfully generated, the device returns a confirmation code with the ballot to the voter.

Now, the voter has two options, either to *cast* or to *challenge* the ballot. If the voter chooses to cast, the device adds the ballot along with the proof and confirmation code to the election record and adds the selections to the running aggregated tallies. If the voter chooses to challenge, then the device needs to show a proof that the provided confirmation code has been generated honestly with the voter's choices; it also publishes this proof in the election record. The voter may verify the proof immediately or refer to the election record when published. The challenged ballot is marked as "spoiled", and the voter needs to restart the voting process. Looking ahead, this is in order to preserve receipt-freeness; if the challenged ballot is included in the tally, the device's proof becomes a receipt so that a coercer can check the voter's cast selections. Challenged must be clearly identified as such in the election record, and their contents are not added to the tallies.

After voting is completed, the tally procedure begins. The ballot collecting device reveals the aggregated selections and a proof of correct tallying along with the election record. Voters can use their confirmation codes to verify that their votes have been properly recorded and included in the tallies, and verifiers can confirm that the tallies are consistent with the published election record.

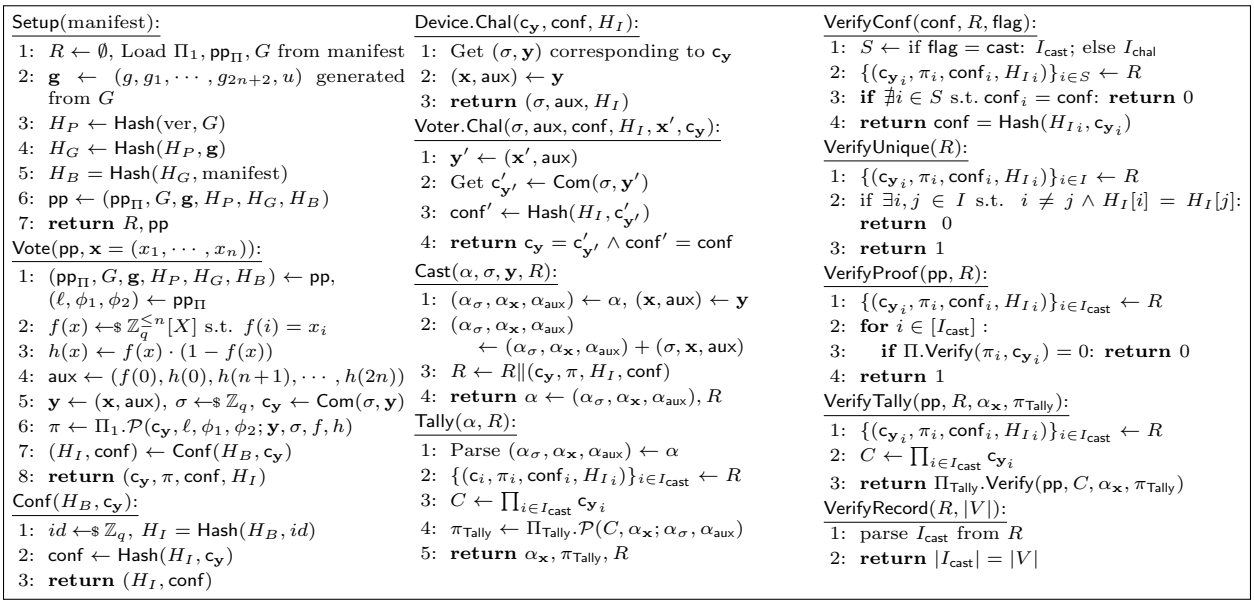
5 Commitment-Based Verifiable Elections

Haechi follows in spirit the homomorphic tallying approach. However, instead of an additively homomorphic encryption, it uses an additively homomorphic commitment. This removes the need for threshold encryption and complicated distributed key agreement.

The protocols in this section are described to use a cyclic group G with a generator g of prime order q in which the discrete logarithm problem is assumed to be hard. Also, we describe Pedersen vector commitment as our commitment scheme. However, the Haechi framework can be fitted to any homomorphic

vector commitment. In § 9, we formulate a post-quantum instantiation of Haechi that utilizing the BDLOP commitment scheme.

Figure 1 provides the pseudocode of protocols we describe here with the proof of ballot well-formedness in §6.



$\underline{\mathsf{Setup}(\text{manifest})}$:
1: $R \leftarrow \emptyset$, Load $\Pi_1, \mathsf{pp}_\Pi, G$ from manifest
2: $\mathbf{g} \leftarrow (g, g_1, \cdots, g_{2n+2}, u)$ generated from G
3: $H_P \leftarrow \mathsf{Hash}(\text{ver}, G)$
4: $H_G \leftarrow \mathsf{Hash}(H_P, \mathbf{g})$
5: $H_B = \mathsf{Hash}(H_G, \text{manifest})$
6: $\mathsf{pp} \leftarrow (\mathsf{pp}_\Pi, G, \mathbf{g}, H_P, H_G, H_B)$
7: **return** R, pp

$\underline{\mathsf{Vote}(\mathsf{pp}, \mathbf{x} = (x_1, \cdots, x_n))}$:
1: $(\mathsf{pp}_\Pi, G, \mathbf{g}, H_P, H_G, H_B) \leftarrow \mathsf{pp}$, $(\ell, \phi_1, \phi_2) \leftarrow \mathsf{pp}_\Pi$
2: $f(x) \leftarrow\$\ \mathbb{Z}_q^{\leq n}[X]$ s.t. $f(i) = x_i$
3: $h(x) \leftarrow f(x) \cdot (1 - f(x))$
4: $\mathsf{aux} \leftarrow (f(0), h(0), h(n+1), \cdots, h(2n))$
5: $\mathbf{y} \leftarrow (\mathbf{x}, \mathsf{aux})$, $\sigma \leftarrow\$\ \mathbb{Z}_q$, $\mathsf{c}_\mathbf{y} \leftarrow \mathsf{Com}(\sigma, \mathbf{y})$
6: $\pi \leftarrow \Pi_1.\mathcal{P}(\mathsf{c}_\mathbf{y}, \ell, \phi_1, \phi_2; \mathbf{y}, \sigma, f, h)$
7: $(H_I, \mathsf{conf}) \leftarrow \mathsf{Conf}(H_B, \mathsf{c}_\mathbf{y})$
8: **return** $(\mathsf{c}_\mathbf{y}, \pi, \mathsf{conf}, H_I)$

$\underline{\mathsf{Conf}(H_B, \mathsf{c}_\mathbf{y})}$:
1: $id \leftarrow\$\ \mathbb{Z}_q$, $H_I = \mathsf{Hash}(H_B, id)$
2: $\mathsf{conf} \leftarrow \mathsf{Hash}(H_I, \mathsf{c}_\mathbf{y})$
3: **return** (H_I, conf)

$\underline{\mathsf{Device.Chal}(\mathsf{c}_\mathbf{y}, \mathsf{conf}, H_I)}$:
1: Get $(\sigma, \mathbf{y})$ corresponding to $\mathsf{c}_\mathbf{y}$
2: $(\mathbf{x}, \mathsf{aux}) \leftarrow \mathbf{y}$
3: **return** $(\sigma, \mathsf{aux}, H_I)$

$\underline{\mathsf{Voter.Chal}(\sigma, \mathsf{aux}, \mathsf{conf}, H_I, \mathbf{x}', \mathsf{c}_\mathbf{y})}$:
1: $\mathbf{y}' \leftarrow (\mathbf{x}', \mathsf{aux})$
2: Get $\mathsf{c}'_{\mathbf{y}'} \leftarrow \mathsf{Com}(\sigma, \mathbf{y}')$
3: $\mathsf{conf}' \leftarrow \mathsf{Hash}(H_I, \mathsf{c}'_{\mathbf{y}'})$
4: **return** $\mathsf{c}_\mathbf{y} = \mathsf{c}'_{\mathbf{y}'} \wedge \mathsf{conf}' = \mathsf{conf}$

$\underline{\mathsf{Cast}(\alpha, \sigma, \mathbf{y}, R)}$:
1: $(\alpha_\sigma, \alpha_\mathbf{x}, \alpha_{\mathsf{aux}}) \leftarrow \alpha$, $(\mathbf{x}, \mathsf{aux}) \leftarrow \mathbf{y}$
2: $(\alpha_\sigma, \alpha_\mathbf{x}, \alpha_{\mathsf{aux}}) \leftarrow (\alpha_\sigma, \alpha_\mathbf{x}, \alpha_{\mathsf{aux}}) + (\sigma, \mathbf{x}, \mathsf{aux})$
3: $R \leftarrow R \| (\mathsf{c}_\mathbf{y}, \pi, H_I, \mathsf{conf})$
4: **return** $\alpha \leftarrow (\alpha_\sigma, \alpha_\mathbf{x}, \alpha_{\mathsf{aux}}), R$

$\underline{\mathsf{Tally}(\alpha, R)}$:
1: Parse $(\alpha_\sigma, \alpha_\mathbf{x}, \alpha_{\mathsf{aux}}) \leftarrow \alpha$
2: $\{(\mathsf{c}_i, \pi_i, \mathsf{conf}_i, H_{I\,i})\}_{i \in I_{\mathsf{cast}}} \leftarrow R$
3: $C \leftarrow \prod_{i \in I_{\mathsf{cast}}} \mathsf{c}_{\mathbf{y}\,i}$
4: $\pi_{\mathsf{Tally}} \leftarrow \Pi_{\mathsf{Tally}}.\mathcal{P}(C, \alpha_\mathbf{x}; \alpha_\sigma, \alpha_{\mathsf{aux}})$
5: **return** $\alpha_\mathbf{x}, \pi_{\mathsf{Tally}}, R$

$\underline{\mathsf{VerifyConf}(\mathsf{conf}, R, \mathsf{flag})}$:
1: $S \leftarrow$ if $\mathsf{flag} = \mathsf{cast}$: I_{cast}; else I_{chal}
2: $\{(\mathsf{c}_{\mathbf{y}\,i}, \pi_i, \mathsf{conf}_i, H_{I\,i})\}_{i \in S} \leftarrow R$
3: **if** $\nexists i \in S$ s.t. $\mathsf{conf}_i = \mathsf{conf}$: **return** 0
4: **return** $\mathsf{conf} = \mathsf{Hash}(H_{I\,i}, \mathsf{c}_{\mathbf{y}\,i})$

$\underline{\mathsf{VerifyUnique}(R)}$:
1: $\{(\mathsf{c}_{\mathbf{y}\,i}, \pi_i, \mathsf{conf}_i, H_{I\,i})\}_{i \in I} \leftarrow R$
2: if $\exists i, j \in I$ s.t. $i \neq j \wedge H_I[i] = H_I[j]$: **return** 0
3: **return** 1

$\underline{\mathsf{VerifyProof}(\mathsf{pp}, R)}$:
1: $\{(\mathsf{c}_{\mathbf{y}\,i}, \pi_i, \mathsf{conf}_i, H_{I\,i})\}_{i \in I_{\mathsf{cast}}} \leftarrow R$
2: **for** $i \in [I_{\mathsf{cast}}]$:
3: **if** $\Pi.\mathsf{Verify}(\pi_i, \mathsf{c}_{\mathbf{y}\,i}) = 0$: **return** 0
4: **return** 1

$\underline{\mathsf{VerifyTally}(\mathsf{pp}, R, \alpha_\mathbf{x}, \pi_{\mathsf{Tally}})}$:
1: $\{(\mathsf{c}_{\mathbf{y}\,i}, \pi_i, \mathsf{conf}_i, H_{I\,i})\}_{i \in I_{\mathsf{cast}}} \leftarrow R$
2: $C \leftarrow \prod_{i \in I_{\mathsf{cast}}} \mathsf{c}_{\mathbf{y}\,i}$
3: **return** $\Pi_{\mathsf{Tally}}.\mathsf{Verify}(\mathsf{pp}, C, \alpha_\mathbf{x}, \pi_{\mathsf{Tally}})$

$\underline{\mathsf{VerifyRecord}(R, |V|)}$:
1: parse I_{cast} from R
2: **return** $|I_{\mathsf{cast}}| = |V|$

Figure 1: The full construction of Haechi instantiated with finite field integer groups for ballot well-formedness proof Π_1 for the simplest election rule (1), where each selection must be either 0 or 1, and their sum must be 1. I denotes all indices and I_{cast}, I_{chal} denotes the index set of cast or challenged ballots in the election record R. $|V|$ is the number of voters who cast ballots.

Election definition. The election is defined by an election manifest file that describes the election, its contests, the selectable options, and the voting modalities for each contest, and may contain other information about the election such as details on voting equipment, the election jurisdiction, and time information.

Setup Phase: Before an election begins, the device runs Setup (in Figure 1). It first initializes the election record R from the election manifest file. The initialized R contains election information but the ballot record fields are empty.

Setup then prepares the public parameters pp including the public parameters for proof scheme and group G information.

All cryptographic objects and operations are tied to these cryptographic parameters and to the specific election and are kept in a specific context through a chain of hashes using the hash function Hash. This prevents the reuse of cryptographic objects such as confirmation codes or proofs of well-formedness for other purposes within the protocol or in a different election.

First, the cryptographic parameters are captured in the parameter hash H_P: H_P hashes to the parameters defining G.

Since the commitment-based approach avoids the need for generating and managing cryptographic keys, only the vector $\mathbf{g} = (g, g_1, g_2, \ldots, g_n)$ of generators needs to be computed in a verifiable way as outlined in Appendix F. The number $n \in \mathbb{N}$ that determines the required number of generators is an election-specific parameter. The generator information is bound to the parameter hash by the generator hash $H_G \leftarrow \mathsf{Hash}(H_P; 0\text{x}01, n, \mathbf{g})$.

Finally the base hash $H_B \leftarrow \mathsf{Hash}(H_G; 0\text{x}02, \text{manifest})$ binds the manifest file and captures the cryptographic and election parameters in a single value. Setup outputs the public parameters $\mathsf{pp} = (G, \mathsf{pp}_\Pi, \mathbf{g}, H_P, H_G, H_B)$.

Voting Phase: A voting phase consists of three steps: vote, challenge, and cast. For each voter V, Vote protocol begins when V enters their selections $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ to the voting device, where x_i denotes the voter's choice for the i-th option. The selections $x_i \in \mathbb{Z}$ are integers, typically very small, and in the simplest, most common case $x_i \in \{0, 1\}$. The device generates a ballot record for $\mathbf{x}$ as follows:

(1) The device picks a uniform random value σ and computes the commitment $\mathsf{c}_{\mathbf{x}} = \mathsf{Com}(\sigma, \mathbf{x})$ that commits to all of the votes in a single Pedersen vector commitment.
(2) It now computes a proof of ballot well-formedness $\pi = \Pi(\mathsf{c}_{\mathbf{x}}; \mathbf{x}, \sigma)$. The details of this proof are deferred to §6.
(3) Finally, the device computes the confirmation code for the ballot. It first computes an identifier hash from the base hash and a random identifier: $H_I = \mathsf{Hash}(H_B; 0\text{x}03, id)$ for $id \leftarrow_\$ \mathbb{Z}_q$. Then it derives the confirmation code $\mathsf{conf} = \mathsf{Hash}(H_I; 0\text{x}04, \mathsf{c}_{\mathbf{x}})$ and gives it to the voter (by printing, displaying, etc.).

After the ballot record is prepared, the device returns $(\mathsf{c}_{\mathbf{x}}, \mathsf{conf})$ to V. V has two options: either to cast or challenge this ballot.

If V chooses to *challenge* the ballot, the device needs to prove that $(\mathsf{c}_{\mathbf{x}}, \mathsf{conf})$ are correctly computed from $\mathbf{x}$. It outputs (σ, H_I) so that the commitment can be opened and verified for correctness. Using (σ, H_I) and the plaintext selection vector $\mathbf{x}$, V can reconstruct the commitment and the confirmation code and compare them to the values obtained from the device. The challenged ballot is marked as "spoiled" in R and the voter restarts the ballot preparation step. (See $\mathsf{Device.Chal}$ and $\mathsf{Voter.Chal}$ in Figure 1.)

If V *casts* the ballot, the device stores $(\mathsf{c}_{\mathbf{x}}, \pi, \mathsf{conf}, H_I)$ in the election record R. The device updates the plaintext tally and randomness aggregator α by adding the plaintext voter selections and randomness $(\mathbf{x}, \sigma)$ to α. (Cast in Figure 1.)

Tallying Phase: The device parses the aggregated randomness and selections $(\alpha_\sigma, \alpha_{\mathbf{x}})$ from α. $\alpha_{\mathbf{x}}$ is essentially the election result announced to the public.

The aggregate commitments C are computed by multiplying the cast commitments together. Due to the additively homomorphic Pedersen vector commitments, C commits to the tally values $\alpha_{\mathbf{x}}$ using the aggregate randomness α_σ. Thus, the device constructs the proof of correct tally π_{Tally} that it knows α_σ such that C opens to $(\alpha_\sigma, \alpha_{\mathbf{x}})$. We use the standard Sigma proof for the tally proof scheme. Looking ahead, the aggregator also contains auxiliary values due to the proof of ballot well-formedness. (We discuss them later and defer the detailed protocol in §A.) The device reports $\alpha_{\mathbf{x}}, \pi_{\mathsf{Tally}}$ to the public with the election record R.

Verification Phase: Now, voters and verifiers can verify the election. Haechi consists of three verification types.

The first is the *cast-or-challenge* paradigm of [14] during the voting phase, as we described above. Giving voters the opportunity to challenge their ballots enables them to hold the voting device accountable and verify correct recording of their votes by verifying the opening of the commitments associated with their confirmation codes. It is important that the device commits to the votes and computes each confirmation code *before* the voter chooses to challenge or cast. A challenged ballot cannot be cast, which ensures the privacy of cast votes as their commitments are never opened. Only voters can verify that their votes are cast as intended.

The second type is record verification, after R is published. Using pp in R, any verifier can verify that the generators used in the protocol have been generated properly. $\mathsf{VerifyRecord}$ checks if the number of cast ballots corresponds to the number of attended voters who cast; verifiers can verify if the device has appended adversarial ballots to R. Second, $\mathsf{VerifyUnique}$ allows verifiers to check if all ballot records in R are unique. If not, verifiers can detect a cast ballot has been duplicated or a same ballot is cast and challenged at the same time, which is contradictory.

Through the confirmation code verification ($\mathsf{VerifyConf}$), voters can verify that the device has not tampered with their cast ballot records in R. Recall that the confirmation codes conf are given to the voters after ballot records are prepared. A voter V first checks if conf is in R and specified either cast or challenged correctly. If it does not exist, V can detect the elimination of V's ballot record from the election record or is marked incorrectly. If conf exists, V recomputes the confirmation code from the the corresponding identifier hash and commitment and checks that it matches conf. A mismatch indicates that the device tampered with the ballot data.

Finally, the election record allows *tally verification* by anyone with access to the record. Verifiers first verify the proofs of well-formedness (VerifyProof). This ensures that each individual commitment was made to a valid set of selections. Verifiers then check if the tally result $\alpha_{\mathbf{x}}$ is correct by verifying π_{Tally} with the product of all commitments. The homomorphic property of Pedersen vector commitments allows verifiers to confirm that $\alpha_{\mathbf{x}}$ is the correct message opening of the commitment aggregation. (VerifyTally).

6 Proof of Ballot Well-Formedness

Verifiable election systems compute non-interactive zero-knowledge proofs (NIZKs) to prove that the vote data follows the election rules defined in the election manifest file. In this section, we discuss the simplest case where the ballot has only a single contest with n selectable options, and a voter must select exactly one option. However, our proof scheme also supports more complicated election rules efficiently; we defer them to Appendix D.

Haechi follows the Compressed Σ-Protocol Theory approach outlined by Attema and Cramer in [5]. We specialize and modify it to the verifiable election setting.

It is worth mentioning that our scheme fundamentally differs from the compressed k-out-of-n partial knowledge proofs in [6], a follow-up work of [5]. Although it first seems that voting could be encoded using a 1-out-of-n proof, it has two critical limitations: (1) it requires n commitments since it essentially proves the knowledge of the opening of 1 commitment out of n commitments; (2) thus, more significantly, it does not constrain the remaining openings—the choices to other candidates—which may open to any arbitrary values.

On the other hand, our proof scheme instead uses a single vector commitment to prove about every option while enforcing stronger constraints. We first show our proof scheme without compression and describe the compression technique. The proofs are described as interactive protocols between a prover and a verifier as in [5] and are made non-interactive by the Fiat-Shamir methodology [38].

A simple base case. Given the vector of voter selections $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ on a ballot, the election rules state requirements for the values of the x_i. Consider the simplest rule where a voter must select exactly one candidate. This translates into the logical statement

$$\forall i \in [n], x_i \in \{0,1\} \wedge \sum_{i=1}^{n} x_i = 1. \tag{1}$$

We adopt the language used in [5]. Let $\mathcal{C} : \mathbb{Z}_q^n \to \mathbb{Z}_q^n$ be an arithmetic circuit over $\mathbb{Z}_q$ with $\mathcal{C}(\mathbf{x}) = \mathbf{x} * (1 - \mathbf{x})$ and $*$ denotes component-wise multiplication. Also, define the linear equation $\ell(\mathbf{x}) = \sum_{i=1}^{n} x_i$. Validity of the above statement for the voter selections $\mathbf{x}$ and the commitment $\mathsf{c}_{\mathbf{x}}$ to $\mathbf{x}$ with randomness σ can be expressed by the relation

$$\mathcal{R}_0 = \{(\mathcal{C}, \ell, \mathsf{c}_{\mathbf{x}}; \mathbf{x}, \sigma) : \mathcal{C}(\mathbf{x}) = 0 \wedge \ell(\mathbf{x}) = 1 \wedge \mathsf{Com}(\sigma, \mathbf{x}) = \mathsf{c}_{\mathbf{x}}\}.$$

To prove $\mathcal{C}(\mathbf{x}) = 0$, [5] proceeds as follows. The circuit $\mathcal{C}$ is comprised of n multiplicative gates whose left input, right input, and output we denote by $(\alpha_i, \beta_i, \gamma_i)$ for $i \in [n]$. In our particular case, we assign $\alpha_i = x_i, \beta_i = 1 - x_i$, and $\gamma_i = 0$ and select three polynomials $f(x), g(x)$ and $h(x)$ that are a multiplicative packed secret sharing of the vectors of the α_i, β_i, and γ_i. Specifically, the prover randomly selects polynomials f, g of degree $\leq n$ such that $f(i) = \alpha_i, g(i) = \beta_i$ and computes $h(x) = f(x)g(x)$ of degree $\leq 2n < q$. Both $f(x)$ and $g(x)$ can be computed with Lagrange interpolation by using the additional degree of freedom to assign $f(0)$ and $g(0)$ uniformly at random.

Verifying $\mathcal{C}(\mathbf{x}) = 0$ is then reduced to checking $f(x) \cdot g(x) = h(x)$. The verifier picks a random challenge $c \leftarrow\!\!\$\, \mathbb{Z}_q \setminus [n]$ and sends it to the prover who responds with $z_1 = f(c)$, $z_2 = g(c)$, and $z_3 = h(c)$. The verifier accepts if $z_1 \cdot z_2 = z_3$. A malicious prover will be caught with probability $> 1 - 2n/(q - n)$.

For this to work, we need to additionally commit to f, g, h. This can be done by committing to a vector of polynomial evaluations $(f(0), \ldots, f(n), g(0), \ldots, g(n), h(0), \ldots, h(2n)) \in \mathbb{Z}_q^{4n+3}$. However, rather than computing an additional vector commitment, we instead extend the vector $\mathbf{x}$ and could commit to the vector $(\mathbf{x}, f(0), \ldots, h(2n)) \in \mathbb{Z}_q^{5n+3}$ all at once.

In our specific case, we can shorten this vector. Observe that we can omit $(f(1),\ldots,f(n)) = (\alpha_1,\ldots,\alpha_n)$ and $(g(1),\ldots,g(n)) = (\beta_1,\ldots,\beta_n)$ because $\alpha_i = x_i$ and $\beta_i = 1 - x_i$ and the x_i are already contained in $\mathbf{x}$. We can also remove $h(1),\ldots,h(n)$ from $\mathbf{y}$ since they are all enforced to be 0.

Furthermore, there is no need to include the polynomial $g(x)$ at all because we can use $1-f(x)$ as pointed out in [5]. Accordingly, the prover only sends $z_1 = f(c)$ and $z_2 = h(c)$ (instead of three values) and the verifier simply checks $z_2 = z_1 \cdot (1 - z_1)$. Thus, we can also remove $g(0)$. Overall, the final vector that is committed to is

$$\mathbf{y} = (x_1,\ldots,x_n, f(0), h(0), h(n+1),\ldots,h(2n)) \in \mathbb{Z}_q^{2n+2}$$

and the prover commits to $\mathbf{y}$ instead of $\mathbf{x}$, computing $\mathsf{c}_\mathbf{y} = \mathsf{Com}(\sigma, \mathbf{y})$. Naturally, the setup for the Pedersen vector commitment scheme now needs to generate $2n + 2$ generators.

Beyond verifying $z_1 \cdot (1 - z_1) = z_2$ the prover must also convince the verifier that $z_1 = f(c)$ and $z_2 = h(c)$ because a malicious prover could send an arbitrary pair (z_1, z_2) that satisfies $z_2 = z_1 \cdot (1 - z_1)$.

To enable the verification that z_1 and z_2 are the values of the polynomials f and h at c, we show that the polynomials can be written as affine maps $\phi_1, \phi_2 : \mathbb{Z}_q^{2n+2} \mapsto \mathbb{Z}_q$ that depend on c such that $f(c) = \phi_1(\mathbf{y})$ and $h(c) = \phi_2(\mathbf{y})$. This allows to verify $z_1 = \phi_1(\mathbf{y})$ and $z_2 = \phi_2(\mathbf{y})$ in terms of $\mathbf{y}$.

The maps ϕ_1 and ϕ_2 are obtained by Lagrange interpolation. As usually, define the Lagrange-coefficients $\ell_{i,m}(x) = \prod_{j=0, j\neq i}^{m}(x - j)/(i - j)$. Using these coefficients, $f(x) = \sum_{i=0}^{n} \ell_{i,n}(x) f(i)$ and $h(x) = \sum_{i=0}^{2n} \ell_{i,2n}(x) h(i)$ and we use

$$\phi_1(\mathbf{y}) = \sum_{i=1}^{n} \ell_{i,n}(c)\mathbf{y}_i + \ell_{0,n}(c)\mathbf{y}_{n+1},$$

$$\phi_2(\mathbf{y}) = \ell_{0,2n}(c)\mathbf{y}_{n+2} + \sum_{i=n+1}^{2n} \ell_{i,2n}(c)\mathbf{y}_{i+2}.$$

Since the Lagrange coefficients only depend on the value c, ϕ_1 and ϕ_2 are public functions.

Finally, we combine $\mathcal{C}(\mathbf{x}) = 0$ and $\ell(\mathbf{x}) = 1$ into a single linear equation, where we interpret ℓ as a function of $\mathbf{y}$: $\ell(\mathbf{y}) = \sum_{i=1}^{n} \mathbf{y}_i$. The verifier provides an additional random value ρ and both prover and verifier can derive the function $L(\mathbf{y}) = \ell(\mathbf{y}) \cdot \rho^0 + \phi_1(\mathbf{y}) \cdot \rho^1 + \phi_2(\mathbf{y}) \cdot \rho^2$. The condition that needs to be verified then becomes $L(\mathbf{y}) = 1 + z_1 \cdot \rho + z_2 \cdot \rho^2$. This means that instead of providing a proof for $\mathcal{R}_0$, we provide a proof for the relation

$$\begin{aligned}\mathcal{R}_1 = \{(\mathsf{c}_\mathbf{y}, L, z_1, z_2, \rho; \mathbf{y}, \sigma) : z_1(1 - z_1) = z_2 \\ \wedge\ L(\mathbf{y}) = 1 + z_1\rho + z_2\rho^2 \wedge\ \mathsf{Com}(\sigma, \mathbf{y}) = \mathsf{c}_\mathbf{y}\},\end{aligned}$$

where the coefficients of L can be computed from ℓ, ϕ_1, and ϕ_2, which in turn depend on c. In conclusion, applying [5] to this situation, we arrive at the proof protocol Π_1 for the validity of (1) as shown in Figure 2. Note that this protocol needs an additional generator $u \leftarrow\!\!\$\ G$, i.e., setup should generate $2n + 3$ generators right away.

We note another change in the election protocol due to the proof scheme. Now, the device keeps the aggregated vector $\mathbf{y}$, instead of $\mathbf{x}$, along with the aggregated randomness. Let $\mathsf{aux} = (f(0), h(0), h(n + 1), \cdots, h(2n)) \in \mathbb{Z}_q^{n+2}$. Thus, the tally proof now proves that it knows $(\alpha_\sigma, \alpha_{\mathsf{aux}})$ with respect to $\alpha_\mathbf{x}$ and the product of all commitments.

The following theorem sums up the completeness, soundness and zero-knowledge properties of protocol Π_1. We defer its proof to Appendix B.

Theorem 1. Π_1 *is perfectly complete, special honest-verifier zero-knowledge, and computationally* $(2n + 1, 3, 2)$*-special-sound under the discrete logarithm assumption.*

Compressing the proof. We now describe a compressed proof scheme Π_1^{comp} that also proves statement (1) but uses the compression technique introduced in [5]. Π_1^{comp} is identical to Figure 2 except for steps 7-9, which are modified as follows. Rather than explicitly sending all $2n + 2$ third-round response values, the prover in Π_1^{comp} proves the knowledge of this response using a compressed-Σ protocol.

Protocol $\Pi_1(\mathsf{c}_{\mathbf{y}}, \ell, \phi_1, \phi_2; \mathbf{y}, \sigma, f, h)$:

1: V: sends a challenge $c \leftarrow\!\!\$\ \mathbb{Z}_q \setminus [n]$ to P.
2: P: computes $z_1 \leftarrow f(c)$, $z_2 \leftarrow h(c)$ and sends z_1, z_2 to V.
3: V: verifies $z_2 = z_1(1 - z_1)$.
4: V: sends $\rho \leftarrow\!\!\$\ \mathbb{Z}_q$ to P.
5: P, V: compute $L(\cdot) \leftarrow \ell(\cdot) \cdot \rho^0 + \phi_1(\cdot) \cdot \rho^1 + \phi_2(\cdot) \cdot \rho^2$.
6: P: samples $\mathbf{r} \leftarrow\!\!\$\ \mathbb{Z}_q^{2n+2}, \gamma \leftarrow\!\!\$\ \mathbb{Z}_q$ and sends $t \leftarrow L(\mathbf{r}), A \leftarrow g^{\gamma} \Pi_{i=1}^{2n+2} g_i^{\mathbf{r}_i}$
7: V: sends $c' \leftarrow\!\!\$\ \mathbb{Z}_q$
8: P: sends $\mathbf{z} \leftarrow c' \cdot \mathbf{y} + \mathbf{r}, \zeta \leftarrow c' \cdot \sigma + \gamma$.
9: V: verifies $g^{\zeta} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{z}_i} \cdot u^{L(\mathbf{z})} \stackrel{?}{=} \mathsf{c}_{\mathbf{y}}^{c'} \cdot A \cdot u^{c' \cdot (1 + z_1 \cdot \rho + z_2 \cdot \rho^2) + t}$

Figure 2: Zero-knowledge proof protocol Π_1 proving (1).

The intuition is that the third round response vector $(\mathbf{z}, \zeta)$—and the vector of generators and the linear equation L—can be recursively folded using random challenges, while maintaining the ability to verify due to linearity.

In each recursive step, the prover splits the generators, responses, and the linear equation into left and right halves. (The linear equation is decomposed into two linear equations: the left (resp., right) equation is obtained from the original equation by setting the right (resp., left) variables to zero.) The prover then commits to each half of the response together with the corresponding linear equation evaluation on them. Upon receiving a random challenge from the verifier, the prover folds the left and right halves using the challenge, thereby dividing the dimension of the response vector, generators, and the linear equation by two. After repeating this procedure for logarithmically many steps, only two response messages remain. The prover sends them as final messages and the verifier runs the verification. We make Π_1^{comp} non-interactive using the Fiat-Shamir heuristics. The full (interactive) protocol is given in Appendix C.

If the length of $\mathbf{y}$ is not a power of 2, the prover pads it with zeroes. Observe that the prover sends 2 commitments every step. There are $\lceil \log(|\mathbf{y}| + 1) \rceil - 1$ recursive steps, thus a proof from Π_1^{comp} contains $(2\lceil \log(2n + 3) \rceil - 1)$ group elements and 5 scalars.

We remark that the compression is especially advantageous for elliptic curve groups where the size of a group element and that of a scalar have little difference. On the other hand, for a group that has a huge difference between the size of a group element and a scalar—e.g., for the finite-field group from [18]—the regular protocol Π_1 may yield smaller proofs.

Amortizing the proofs. We can further reduce the bulletin board size by proving about $\mathcal{R}_1$ for all voters' ballots with the similar cost of proving about one ballot. Let's denote m as the number of voters and $\theta = 1 + z_1\rho + z_2\rho^2$. The idea is that rather than checking $L(\mathbf{y}_i) = \theta_i$ separately per i-th ballot, the verifier checks the random linear combination with the challenge c':

$$\text{for } \tilde{\mathbf{y}} = \mathbf{r} + \sum_{i=1}^{m} \mathbf{y}_i^{c'^i}, L(\tilde{\mathbf{y}}) = L(\mathbf{r}) + \sum_{i=1}^{m} \theta_i^{c'^i}.$$

In more detail, given the verifier's challenge c, the prover sends the pair $(z_1, z_2)_i$ corresponding to the commitment $\mathsf{c}_{\mathbf{y}_i}$ for all $i \in [m]$ to the verifier. The verifier checks if $z_{2i} = z_{1i} \cdot (1 - z_{1i})$ for all i. The verifier now sends another challenge ρ, and both parties derive L and $\theta_1, \cdots, \theta_m$ from c, ρ, and $\mathsf{c}_{\mathbf{y}_i}$.

Now, the prover samples $\mathbf{r}$ and γ and sends t and A computed the same as Π_1 to the verifier. Given the challenge c', the prover computes the third round message differently:

$$\mathbf{z} \leftarrow \sum_{i=1}^{m} c'^i \mathbf{y}_i + \mathbf{r} \text{ and } \zeta \leftarrow \sum_{i=1}^{m} c'^i \sigma_i + \gamma.$$

The verifier checks

$$g^{\zeta} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{z}_i} \cdot u^{L(\mathbf{z})} \stackrel{?}{=} \prod_{i=1}^{m} \mathsf{c}_{\mathbf{y}_i}^{c'^i} \cdot A \cdot u^{\sum_{i=1}^{m} c'^i \theta_i + t}.$$

Notice that the challenges are derived after the polynomials f and h for all ballots are already committed. Let $P(c') = L(\mathbf{z}) - (\sum_{i=1}^{m} {c'}^i \theta_i + t) = \sum_{i=1}^{m} {c'}^i (L(\mathbf{y}_i) - \theta_i)$. If all ballots are valid, $P(c') = 0$. If at least one ballot does not satisfy the linear relation, the probability that the malicious prover can convince the verifier is bounded by $m/|\mathbb{Z}_q|$ by Schwartz-Zippel Lemma. We defer the full analysis to Appendix E.

Without compression, the proof size of Π_1^{Am} is $1\mathbb{G} + (2m + 2n + 4)\mathbb{Z}_q$, whereas that of Π_1 for m ballots is $m\mathbb{G} + m(2n + 6)\mathbb{Z}_q$. With compression, the proof size is $(2\lceil \log(2n + 3) \rceil - 1)\mathbb{G} + (2m + 3)\mathbb{Z}_q$.

We briefly remark the trade-off: the device needs to retain all witnesses until all voters vote to construct the amortized proof, whereas using Π_1, the device can erase each voter's ballot data right after creating each ballot record. We thus leave Π_1^{Am} as optional.

7 Security and Integrity

In this section, we give formal game-based definitions of ballot privacy and the integrity properties: cast-as-intended, recorded-as-cast, and tallied-as-recorded verifiability. We give high-level proof sketches for these properties as well as receipt-freeness; the full proofs are in Appendices H, I, J, K.

7.1 Ballot Privacy

Although the game-based definition BPRIV (ballot privacy) proposed in [22] has been widely used to analyze various systems including widely used Helios [1] and Belenios [29], it is not well suited for Haechi, or more generally, to in-person voting systems. BPRIV models remote voting and allows an adversary to cast ballots as a function of those that appear on the bulletin board in order to break privacy. In in-person voting, ballots are cast by devices that are are trusted for privacy and are used by many voters. Instead, privacy in in-person voting focuses on the content of the published bulletin board. Also, accordingly, verifiable in-person voting schemes usually contain cast-as-intended verification mechanisms, such as challenge in Haechi, which BPRIV does not capture. The transcripts of the verification are revealed in the election record, to which the adversary has additionally access.

Game $\mathsf{PRIV}^{\mathcal{A},\beta}(\lambda)$:
1: $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$,
2: $BB_0, BB_1, V_0, V_1, \Pi_0, \Pi_1 \leftarrow \emptyset$
3: $\beta' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Vote},\mathcal{O}\mathsf{Proof},\mathcal{O}\mathsf{Tally}}(\mathsf{pp})$
4: **return** β'

$\mathcal{O}\mathsf{Proof}(i)$:
1: if $V_0[i] \neq V_1[i]$: **abort**
2: mark $BB_\beta[i]$ as challenged.
3: $\pi_\beta \leftarrow \Pi_\beta[i]$
4: for $\delta \in \{0, 1\}$:
5: $V_\delta \leftarrow V_\delta \setminus V_\delta[i]$
6: $\Pi_\delta \leftarrow \Pi_\delta \setminus \Pi_\delta[i]$
7: **return** π_β

$\mathcal{O}\mathsf{Vote}(v_0, v_1)$:
1: for $\delta \in \{0, 1\}$:
2: $(b_\delta, \pi_\delta) \leftarrow \mathsf{Vote}(v_\delta)$
3: $BB_\delta \leftarrow BB_\delta \| b_\delta$
4: $V_\delta \leftarrow V_\delta \| v_\delta$, $\Pi_\delta \leftarrow \Pi_\delta \| \pi_\delta$
5: **return** BB_β

$\mathcal{O}\mathsf{Tally}()$:
1: $(\alpha_{\mathbf{x}}, \pi_{\mathsf{Tally}}, BB_0)$
$\leftarrow \mathsf{Tally}(V_0, BB_0)$
2: if $\beta = 0$:
3: **return** $(\alpha_{\mathbf{x}}, \pi_{\mathsf{Tally}}, BB_0)$
4: if $\beta = 1$:
5: $C \leftarrow \prod_{i \in I_{\mathsf{cast}}(BB_1)} \mathsf{c}_{\mathbf{y}_i}$
6: $\pi'_{\mathsf{Tally}} \leftarrow \mathcal{S}(C, \alpha_{\mathbf{x}})$
7: **return** $(\alpha_{\mathbf{x}}, \pi'_{\mathsf{Tally}}, BB_1)$

Figure 3: The ballot privacy game PRIV. The adversary $\mathcal{A}$ wins if it can successfully guess β. $I_{\mathsf{cast}}(BB_1)$ denotes the index of cast ballot records in BB_1, whose cast-as-intended proofs are not revealed through $\mathcal{O}\mathsf{Proof}$. $\mathcal{O}\mathsf{Tally}$ is queried by the adversary once at the end of the experiment.

Thus, we formally define ballot privacy for in-person voting using the game PRIV in Figure 3. Similar to BPRIV, PRIV models the real and fake world, and the adversary $\mathcal{A}$ wins if it can successfully guess which world it is in. In more detail, PRIV is initialized with empty bulletin board, vote set, and cast-as-intended

proof set (BB_b, V_b, Π_b) for each world $b \in \{0,1\}$. (In Haechi, the proof is essentially the commitment opening.) The challenger randomly selects a bit β. $\mathcal{A}$ can add different votes to V_0 and V_1 through the oracle $\mathcal{O}\mathsf{Vote}$. $\mathcal{O}\mathsf{Vote}$ generates corresponding ballot records and cast-as-intended proofs and updates the bulletin boards and the proof sets. When the adversary queries $\mathcal{O}\mathsf{Tally}$, the result is obtained by BB_0 in both worlds. When $\beta = 0$, π_{Tally} is generated by the prover, whereas when $\beta = 1$, π_{Tally} is simulated.

$\mathcal{A}$ can also query $\mathcal{O}\mathsf{Proof}$ an index i and the oracle returns $\Pi_\beta[i]$. We remark that $V_0[i]$ must be the same as $V_1[i]$. If $V_0[i] \neq V_1[i]$, since $\mathcal{A}$ knows the vote sequence, $\mathcal{A}$ can easily find out which world it is in by checking if which value makes $\Pi_\beta[i]$ accepted. For an adversary $\mathcal{A}$ and an election scheme E, we define the PRIV advantage of $\mathcal{A}$ against E as

$$\mathrm{Adv}^{\mathsf{PRIV}}_{E}(\mathcal{A}) := |\Pr[\mathsf{PRIV}^{\mathcal{A},0}_{E}(\lambda) = 1] - \Pr[\mathsf{PRIV}^{\mathcal{A},1}_{E}(\lambda) = 1]|.$$

We say that E satisfies ballot privacy if $\mathrm{Adv}^{\mathsf{PRIV}}_{E}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

Theorem 2. *For all adversaries $\mathcal{A}$, the ballot privacy game* PRIV *in Figure 3 with up to $Q_{\mathcal{O}\mathsf{Vote}}$ queries from $\mathcal{A}$ to $\mathcal{O}\mathsf{Vote}$, $Adv^{\mathsf{PRIV}}_{\mathit{Haechi}}(\mathcal{A}) \leq Adv^{\mathsf{ZK}}_{\Pi_{\mathsf{Tally}}}(\lambda) + 2Q_{\mathcal{O}\mathsf{Vote}} \cdot Adv^{\mathsf{ZK}}_{\Pi_1}(\lambda)$.*

Proof Sketch. We start from $\mathsf{PRIV}^{\mathcal{A},0}$ and proceed with a sequence of game hops to $\mathsf{PRIV}^{\mathcal{A},1}$. We first proceed with a game hop to a game where the tally correctness proof π_{Tally} is always generated by a simulator $\mathcal{S}_{\mathsf{Tally}}$ regardless of β. This game hop is bounded by the zero-knowledge property of Π_{Tally}. The second game hop generates the ballot well-formedness proof π using a simulator $\mathcal{S}$ of Π_1. This hop is also bounded by the zero-knowledge property of Π_1.

At this point, the adversary sees BB_0 with simulated ballot well-formedness proofs and Π_0 for the cast-as-intended proofs. We proceed with a series of hybrids for the number of queries $Q_{\mathcal{O}\mathsf{Vote}}$ that replaces each ballot record with that in BB_1 one by one. $\mathcal{O}\mathsf{Proof}$ and $\mathcal{O}\mathsf{Tally}$ responses change accordingly. We prove that the adversary's views in each adjacent hybrids are identical due to the perfect hiding property of our commitment scheme.

After the hybrids, $\mathcal{A}$ always sees BB_1, Π_1, and the tally proof with respect to BB_1 (but X from V_0). The ballot well-formedness proofs in $\mathcal{A}$'s view are still simulated. Thus, we proceed with a final game hop where the ballot well-formedness proofs are generated by the prover. This game hop is bounded by zero-knowledge property of Π_1. Now, the adversary's view is exactly that in $\mathsf{PRIV}^{\mathcal{A},1}$. The full proof is in Appendix H. □

7.2 Cast-as-intended Verifiability

Cast-as-intended verifiability implies that at the time of voting, the voters should be able to verify if the device generates ballot records honestly with their selections. We first define a single-ballot notion of cast-as-intended verifiability and show how to derive an election-wide cast-as-intended guarantee.

Game $\mathsf{sCAI}^{\mathcal{A}}_{\mathsf{VerifyCAI}}(\lambda)$:

1: $\mathsf{pp} \leftarrow \mathsf{Setup}(\cdot)$
2: $(v, v', b', \sigma') \leftarrow \mathcal{A}(\mathsf{pp})$
3: **if** $\mathsf{extract}(b', v', \sigma') = 0$: **abort**
4: **return** $v' \neq v \wedge \mathsf{VerifyCAI}(b', \sigma', v) = 1$

Figure 4: A single-ballot cast-as-intended verifiability game sCAI. The game is parameterized by a verification algorithm VerifyCAI. This allows the experiment to capture different mechanisms by which a voter associates a ballot record with the voter's intended vote. $\mathcal{V}$ is the space of (valid) votes.

We formalize single-ballot cast-as-intended verifiability as sCAI in Figure 4. $\mathcal{A}$ wins if it can output a ballot record b' for an adversarial vote $v' \neq v$, but VerifyCAI still considers that b' is associated with the intended vote v. For an adversary $\mathcal{A}$, we define sCAI advantage of $\mathcal{A}$ against a verifiable election scheme E as

$$\mathrm{Adv}^{\mathsf{sCAI}}_{E,\mathsf{VerifyCAI}}(\mathcal{A}) := \Pr[\mathsf{sCAI}^{\mathcal{A}}_{E,\mathsf{VerifyCAI}}(\lambda) = 1].$$

We say that E satisfies single-ballot cast-as-intended verifiability if $\mathrm{Adv}^{\mathsf{sCAI}}_{E,\mathsf{VerifyCAI}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

We remark that extract check is to prevent trivial win. It verifies if b' is constructed with respect to the selection v'. Without this check, $\mathcal{A}$ can output (v', b') such that $v' \neq v$ but b' is produced from v and trivially win the game. The game is parameterized by a verification algorithm VerifyCAI. In Haechi, the challenge verification algorithm Voter.Chal corresponds to VerifyCAI. The values b', v, σ' in the game corresponds to $(\mathsf{c_y}, H_I, \mathsf{conf}), \mathbf{x}, (\sigma, \mathsf{aux})$ in Haechi respectively.

Theorem 3. *For all PPT adversaries and the single-ballot cast-as-intended verifiability game* sCAI *in Figure 4,* $Adv^{\mathsf{sCAI}}_{\mathit{Haechi},\mathsf{Voter.Chal}}(\mathcal{A}) \leq Adv^{\mathsf{CB}}_{\mathsf{Com}}(\lambda)$.

Proof sketch. We first instantiate sCAI tailored for Haechi. A vote v corresponds to a vote vector $\mathbf{x}$. $\mathcal{A}$ outputs a ballot record $b' = (\mathsf{c}'_{\mathbf{y}}, \pi', H'_I, \mathsf{conf}')$ together with σ' and $\mathbf{y}'$. The extraction algorithm runs Open of the commitment scheme.

Suppose $\mathcal{A}$ wins sCAI. Then $\mathbf{x}' \neq \mathbf{x}$ but Voter.Chal accepts. Thus, $\mathsf{c}'_{\mathbf{y}} = \mathsf{Com}((\mathbf{x}', \mathsf{aux}'), \sigma') = \mathsf{Com}((\mathbf{x}, \mathsf{aux}'), \sigma')$. We can then construct an adversary $\mathcal{A}'$ against binding property. However, $\mathcal{A}'$'s advantage is bounded by negligible probability. The full proof is in Appendix I. □

Corollary 1. *Consider an election that embeds Haechi. Suppose that a ballot record is independently challenged with probability at least p (after the ballot record has been fixed). Also suppose that the device generates t number of ballot records different from the intended votes. The probability that none of these t misbehavior is detected in the election is bounded by* $(1-p)^t + \mathsf{negl}(\lambda) < e^{-tp} + \mathsf{negl}(\lambda)$.

Proof. Let's denote T as the set of all adversarial ballot records and $C \subseteq T$ as the set of challenged ballots among T. Also, let W_i be the event that the device wins sCAI game on the i-th ballot record in C. Since the challenge-or-cast decision is made after a ballot record is fixed, C is independent of the events W_i. Let BAD event be an event that an adversary $\mathcal{A}$ can pass all challenge verifications.

$$\begin{aligned}
\Pr[\mathsf{BAD}] &= \Pr[\forall i \in |C| : W_i] \\
&= \Pr[\forall i \in |C| : W_i \wedge C = \emptyset] + \Pr[\forall i \in |C| : W_i \wedge C \neq \emptyset] \\
&= \Pr[C = \emptyset] \cdot \Pr[\forall i \in |C| : W_i \mid C = \emptyset] \\
&+ \Pr[C \neq \emptyset] \cdot \Pr[\forall i \in |C| : W_i \mid C \neq \emptyset] \\
&\leq (1-p)^t + (1-(1-p)^t) \cdot \Pr[\forall i \in |C| : W_i \mid C \neq \emptyset] \\
&\leq (1-p)^t + (1-(1-p)^t) \cdot \mathrm{Adv}^{\mathsf{sCAI}}_{\mathsf{Haechi}}(\lambda).
\end{aligned}$$

The fourth equality holds from follows. If $C \neq \emptyset$, for a fixed i^*, $\Pr[\forall i \in |C|, W_i] \leq \Pr[W_{i^*}]$. And by independence of C from W_{i^*}, $\Pr[W_{i^*}] \leq \mathrm{Adv}^{\mathsf{sCAI}}_{\mathsf{Haechi}}(\mathcal{A}) = \mathsf{negl}(\lambda)$. □

We note that challenges in practice are not independent with a fixed probability; the value p is simply a lower bound on the challenge probability—akin to a min-entropy argument. Even with a small p, once the number of t exceeds $1/p$, the adversarial behavior is detected with probability well over $1/2$. We further discuss voter challenge strategies including a setting where each voter may issue only a bounded number of challenges in Remark 1 in Appendix I.

7.3 Recorded-as-cast Verifiability

Recorded-as-cast verifiability implies that the voters should be able to detect if the bulletin board has modified a commitment to a cast ballot after the voter left. Similarly to §7.2, we define single-ballot recorded-as-cast verifiability game in Figure 5 and analyze the election-level recorded-as-cast verifiability.

In sRAC, given pp, $\mathcal{A}$ should output two bulletin board BB, BB' where BB contains a ballot b but BB' has modified (or deleted) b. $\mathcal{A}$ wins if the recorded-as-cast verification on b and BB' still succeeds. That is, the verifier is convinced that BB' still contains b, even after $\mathcal{A}$ tampered with b.

Observe that the game allows changes to the ballot well-formedness proofs. This is because proofs are randomized: an honest bulletin board may produce different proofs for the same statement and witness. For an adversary $\mathcal{A}$, we define the sRAC advantage of $\mathcal{A}$ against a verifiable election scheme E as

$$\mathrm{Adv}^{\mathsf{sRAC}}_{E,\mathsf{VerifyRAC}}(\mathcal{A}) := \Pr[\mathsf{sRAC}^{\mathcal{A}}_{E,\mathsf{VerifyRAC}}(\lambda) = 1].$$

Game $\mathsf{sRAC}^{\mathcal{A}}_{\mathsf{VerifyRAC}}(\lambda)$:

1: $\mathsf{pp} \leftarrow \mathsf{Setup}(\cdot)$
2: $(BB, BB', b) \leftarrow \mathcal{A}(\mathsf{pp})$
3: **return** $b \in BB \wedge b \setminus \pi \notin BB' \wedge \mathsf{VerifyRAC}(BB', b) = 1$

Figure 5: Single-ballot recorded-as-cast verifiability game sRAC. We abuse the notation and use $b \setminus \pi$ to denote a ballot record except the ballot well-formedness proof. For simplicity, we consider only the cast ballot records on a BB.

We say that an election scheme satisfies single-ballot recorded-as-cast verifiability if $\mathrm{Adv}^{\mathsf{sRAC}}_{E,\mathsf{VerifyRAC}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

In Haechi, VerifyRAC corresponds to VerifyConf, VerifyUnique, and VerifyRecord.

Theorem 4. *For any PPT adversary $\mathcal{A}$ and the single-ballot recorded-as-cast verifiability game* sRAC *in Figure 5, $Adv^{\mathsf{sRAC}}_{\mathit{Haechi},\mathsf{VerifyConf},\mathsf{VerifyUnique},\mathsf{VerifyRecord}}(\mathcal{A}) \leq Adv^{CR}_{\mathsf{Hash}}(\lambda)$.*

Proof Sketch. Let $b = (\mathsf{c}_{\mathbf{y}}, \pi, H_I, \mathsf{conf})$. Our strategy is to show that if $\mathcal{A}$ wins the game, then it contradicts the assumption that $b \setminus \pi \notin BB'$.

First, given $\mathcal{A}$'s output BB', the challenger runs VerifyRecord, with BB' and the number of cast ballots in BB. Thus, if $\mathcal{A}$ adds more cast ballot records to BB' than BB, $\mathcal{A}$ loses. Also, due to VerifyUnique, if $\mathcal{A}$ wins, there are no duplicate ballot records—neither duplicate cast records nor a ballot both cast and challenged. VerifyUnique together with cast-or-challenge in Haechi neutralizes clash attacks, where a device issues one conf to multiple voters while adding a single ballot to the tally.

By the definition of VerifyConf, if $\mathcal{A}$ wins, then there exists a ballot record $b_i = (\mathsf{c}_{\mathbf{y}i}, \pi_i, H_{Ii}, \mathsf{conf}_i)$ where the confirmation code is exactly conf, i.e. $\mathsf{conf}_i = \mathsf{conf}$. Then, VerifyConf checks if $\mathsf{conf}_i = \mathsf{Hash}(H_{Ii}, \mathsf{c}_{\mathbf{y}i})$. Suppose that $(H_{Ii}, \mathsf{c}_{\mathbf{y}i}) \neq (H_I, \mathsf{c}_{\mathbf{y}})$. This means that $\mathsf{Hash}(H_{Ii}, \mathsf{c}_{\mathbf{y}i}) = \mathsf{Hash}(H_I, \mathsf{c}_{\mathbf{y}})$. However, the probability of this event is bounded by collision resistant property of our underlying hash function. Therefore $(H_{Ii}, \mathsf{c}_{\mathbf{y}i}) = (H_I, \mathsf{c}_{\mathbf{y}})$ with overwhelming probability. This contradicts that $b \setminus \pi \notin BB'$. □

Corollary 2. *Consider an election that embeds Haechi. Suppose voter each independently verify their ballot confirmations against the published bulletin board with a probability at least q. If the published bulletin board tampers with t distinct cast ballots, the probability that none of t tamperings is detected is bounded by $(1-q)^t + \mathsf{negl}(\lambda)$.*

We omit the proof as it is exactly the same as that of Corollary 1 where C is not the set of challenged ballots among the adversarial ballots but the set of ballots that the voters run VerifyConf with.

7.4 Tallied-as-recorded Verifiability

Tallied-as-recorded verifiability definition is well understood in literature [15, 19, 27, 28, 46, 47]. Using the game-based definition, the adversary wins if it can produce a dishonest tally result from a bulletin board. As the definition is clear, we provide a definition tailored for Haechi in Figure 6.

The adversary $\mathcal{A}$ outputs an election record BB, a claimed aggregate selection vector $\alpha'_{\mathbf{x}}$, and tally proof π'_{Tally}. It additionally provides an honest opening $(\alpha_{\mathbf{x}}, \alpha_\sigma, \alpha_{\mathsf{aux}})$ of the aggregate commitment derived honestly from the cast ballot records in BB. Notice that this opening is only used for the experiment and is not included in the election record BB.

The challenger gets the product of all cast commitments, C. It then verifies that $(\alpha_{\mathbf{x}}, \alpha_\sigma, \alpha_{\mathsf{aux}})$ is a valid opening of C. $\mathcal{A}$ wins if the claimed aggregate $\alpha'_{\mathbf{x}}$ and the proof π'_{Tally} passes tally verification with BB but $\alpha'_{\mathbf{x}} \neq \alpha_{\mathbf{x}}$. For an adversary $\mathcal{A}$, we define the TAR advantage of $\mathcal{A}$ against an election scheme E as

$$\mathrm{Adv}^{\mathsf{TAR}}_{E}(\mathcal{A}) := \Pr[\mathsf{TAR}^{\mathcal{A}}_{E}(\lambda) = 1].$$

We say that a scheme satisfies tallied-as-recorded verifiability if $\mathrm{Adv}^{\mathsf{TAR}}_{E}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

Although it may seem counterintuitive that $\mathcal{A}$ outputs both BB and $\alpha'_{\mathbf{x}}$, this reflects the real-world setting in which the device produces both the record and the tally result. Tallied-as-recorded verifiability

Game $\mathsf{TAR}^{\mathcal{A}}(\lambda)$:

1: $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2: $(BB, \alpha'_{\mathbf{x}}, \pi'_{\mathsf{Tally}}, \alpha_{\mathbf{x}}, \alpha_\sigma, \alpha_{\mathsf{aux}}) \leftarrow \mathcal{A}(\mathsf{pp})$
3: Abort if $\mathsf{VerifyProof}(\mathsf{pp}, BB) = 0$
4: $C \leftarrow \prod_{i \in I_{\mathsf{cast}}(BB)} \mathsf{c}_{\mathbf{y}i}$
5: Abort if $C \neq \mathsf{Com}\big(\alpha_\sigma, (\alpha_{\mathbf{x}}, \alpha_{\mathsf{aux}})\big)$
6: **return** $\rho(\alpha'_{\mathbf{x}}) \neq \rho(\alpha_{\mathbf{x}}) \ \wedge \ \mathsf{VerifyTally}(\mathsf{pp}, BB, \alpha'_{\mathbf{x}}, \pi'_{\mathsf{Tally}}) = 1$

Figure 6: Tallied-as-recorded Verifiability Game TAR. ρ is the result function that outputs the tally result from the aggregates. In Haechi, ρ outputs the input itself. $I_{\mathsf{cast}}(BB)$ is the set of indices of cast ballots in BB.

does not address whether the record reflects the voters' intended choices; instead, it captures the internal consistency between the record and the tally outcome.

Theorem 5. *For any PPT adversary $\mathcal{A}$, a polynomial bound $R_{\max}$ of the length of election record BB, and the tallied-as-recorded verifiability game TAR in Figure 6,* $Adv^{\mathsf{TAR}}_{\mathit{Haechi}}(\mathcal{A}) \leq R_{\max} \cdot Adv^{\mathsf{KS}}_{\Pi_1}(\lambda) + 2 \cdot Adv^{CB}_{\mathsf{Com}}(\lambda) + Adv^{\mathsf{KS}}_{\Pi_{\mathsf{Tally}}}(\lambda)$.

Proof Sketch. Without loss of generality, we suppose that BB only contains the cast ballot records. We first show that Π_1 and Π_{Tally} satisfy knowledge soundness (Corollary 3, 4). Then, we proceed with a game hop to a game that additionally extracts a valid witness from every accepting ballot well-formedness proof in BB. Each witness contains the committed message $\mathbf{y}_i = (\mathbf{x}_i, \mathsf{aux}_i)$ and the randomness σ_i. We bound this game hop using the knowledge soundness of our proof scheme.

From the extracted witness, we can derive the sum of $(\mathbf{x}_i, \mathsf{aux}_i, \sigma_i)$ for all i. Let's denote the aggregated values as $(\alpha^*_{\mathbf{x}}, \alpha^*_{\mathsf{aux}}, \alpha^*_\sigma)$. We hop to another game where we replace $\alpha_{\mathbf{x}}$ in the winning condition with $\alpha^*_{\mathbf{x}}$. By the homomorphism of Com, $(\alpha^*_\sigma, (\alpha^*_{\mathbf{x}}, \alpha^*_{\mathsf{aux}}))$ also opens C. If $\rho(\alpha_{\mathbf{x}}) \neq \rho(\alpha^*_{\mathbf{x}})$, the two openings contain distinct messages. This probability is bounded by the binding property of our commitment scheme.

In the final game hop, the challenger additionally extracts a witness pair $(\alpha'_\sigma, \alpha'_{\mathsf{aux}})$ from the accepting tally correctness proof π'_{Tally} for the statement $(C, \alpha'_{\mathbf{x}})$ and verifies that it satisfies $\mathcal{R}_{\mathsf{Tally}}$. This game hop is bounded by the knowledge soundness of Π_{Tally}.

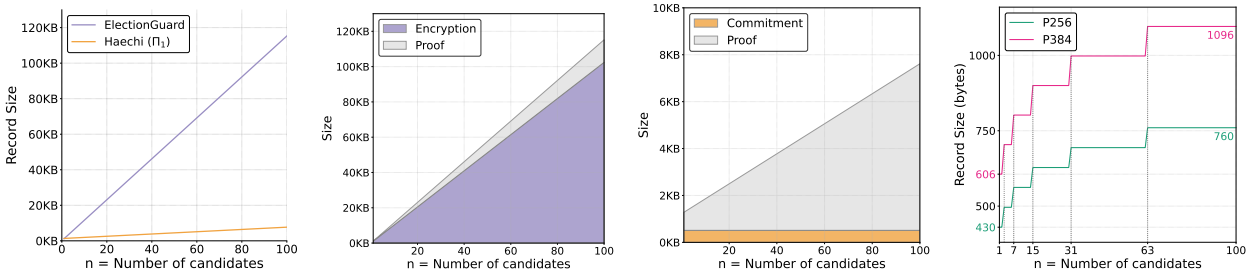
If $\mathcal{A}$ wins the final game, the extracted tally witness gives $C = \mathsf{Com}\big(\alpha'_\sigma, (\alpha'_{\mathbf{x}}, \alpha'_{\mathsf{aux}})\big)$, whereas the aggregated ballot witnesses give $C = \mathsf{Com}\big(\alpha^*_\sigma, (\alpha^*_{\mathbf{x}}, \alpha^*_{\mathsf{aux}})\big)$. Moreover, winning implies $\rho(\alpha'_{\mathbf{x}}) \neq \rho(\alpha^*_{\mathbf{x}})$ and therefore $\alpha'_{\mathbf{x}} \neq \alpha^*_{\mathbf{x}}$. We show that $\mathcal{A}$'s advantage is bounded by the binding assumption of Com. The full proof is in Appendix J. □

We conclude this section with a brief discussion of the receipt-freeness of Haechi. Resisting coerced voters can cast their real vote while claiming to the coercer that they used the given fake vote. Cast ballot records provide no transferable evidence of individual vote and the voters never receives the randomness used to construct the commitment for cast ballots. We give a detailed discussion in Appendix K.

8 Implementation and Evaluation

We have implemented Haechi in Rust. The election scheme is instantiated with two NIST standard elliptic curve groups, P-256 and P-384, as well as the finite field group introduced by ElectionGuard [18]. The implementation uses the hash function Hash based on HMAC-SHA-256, and the random oracle is instantiated with $\mathsf{HashToField}$ as in RFC9380 [37] (see Appendix F). All experiments were evaluated on an Apple M2 Pro CPU with 10 physical cores and 32GB RAM.

Our empirical analysis measures proof size, election record size, and prover and verifier running times. Among these, the election record size matters most, since it determines whether verifiers can feasibly download and audit an election.



(a) Ballot record size (b) The cost breakdown of ElectionGuard (c) The cost breakdown of Haechi (d) Ballot record size for elliptic curve groups

Figure 7: The comparison between ElectionGuard vs. Haechi in terms of the size of a ballot record per voter for different numbers of candidates. Figure 7b and 7c shows the breakdown of the cost in 7a. Figure 7d shows the ballot record size of Haechi when instantiated with P-256 and P-384 elliptic curve groups.

Election record size: Haechi vs. ElectionGuard We first instantiate Haechi with the ElectionGuard finite field group [18] and Π_1 and compare the election record size (Figure 7). Note that a group element has 512 bytes and a scalar 32 bytes.

An election record contains a large amount of information about the election and the cryptographic artifacts that allow election verification. For simplicity, we focus only on the cryptographic objects as they constitute the majority of the election record data. These are: (1) cast ballot records, (2) challenged ballot records, and (3) the final tally proof. Challenged ballot records additionally contain the commitment openings, which are marginal; they can even be represented with bits. Thus, we focus on (1) and (3) in this section.

For ElectionGuard, the data for a cast ballot record consists of a ciphertext for each selection in each contest, a range proof for each ciphertext and range proofs for the selection limits of each contest that prove that the ballot is well-formed. The Haechi election record consists of a commitment to the voter's choices and a proof of well-formedness for the commitment. Both types of election records each contain an identifier, an identifier hash, and the ballot confirmation code among other small data items. Since they are contained in both records and are comparatively small, we do not take them into account in our comparison. The bulk of the data consists of ciphertexts and proofs, or commitments and proofs, respectively.

Figure 7 displays the size of the election record per ballot depending on the number n of selectable options on the ballot. Figure 7a shows that the size of an election record in Haechi is always smaller than that of ElectionGuard, and the gap becomes wider as the number of candidates grows. If the number of candidates is $n = 100$, the election record size is 115,264 bytes per ballot in ElectionGuard. In Haechi, the election record size is 7,688, when using Π_1 which gives approximately a factor $\times 15$ improvement from ElectionGuard in this setting and increases further as n grows.

As Figure 7b shows, the dominating factor of a record size in ElectionGuard are the ciphertexts encrypting the voter selections. ElectionGuard encrypts each selection using ElGamal encryption; thus when $n = 100$, the ciphertexts amount to 2 group elements $\times 100 = 102{,}400$ bytes. On the other hand, the corresponding data in Haechi consists of a single vector commitment, i.e., a single group element. The commitment size is always 512 bytes regardless of the number of candidates (Figure 7c). This yields a factor $\times 200$ improvement in data size for $n = 100$, or a factor $\times 2n$ in general.

Although Figure 7c indicates that the proof contributes a much larger fraction of the record size for Haechi than for ElectionGuard, the absolute size of the proof component in Haechi is smaller than that in ElectionGuard when $n > 11$. ElectionGuard generates a CDS [30] proof consisting of 4 scalars for each selection encryption; thus, the proof size grows linearly with n. Concretely, when $n = 100$, it is 12,800 bytes. On the other hand, Haechi proves well-formedness for all selections jointly, making the proof size 7,172 bytes for Π_1, a reduction by a factor $\times 1.8$.

An election record also contains the transcript from tally. ElectionGuard publishes the tally decryption proof, which is 2 scalars per candidate, yielding 6.4KB with 100 candidates. The equivalent in Haechi is the tally proof from Π_{Tally}, which is 3.8KB. Compared to encryptions/commitments and proofs, these values are

minor in both schemes.

Election record size under elliptic curve groups Our implementation also supports the elliptic curve groups P-256 and P-384. Figure 7d shows the size of the election record per ballot with elliptic curve groups and Π_1. The first observation is that communication complexity is much smaller when using elliptic curve groups than the finite field group. With $n = 100$ candidates, the record size per voter is 760 bytes with P-256 and 1,096 bytes with P-384 vs. $7,688$ bytes with the finite field group, $\sim 10\times, \sim 7\times$ smaller. The tally proof size (one for the whole election) is 502 bytes and 742 bytes respectively.

Another observation is that the proof size of Π_1 in the elliptic curve group setting grows in a stepwise manner, whereas in the finite field group setting it grows linearly. This is essentially the result of using the compressed proof as it includes only a logarithmic number of group elements in the number of committed values.

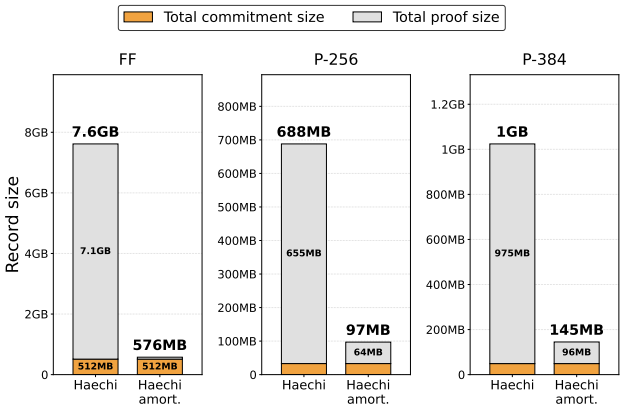


Figure 8: The election record size for an election with 1M voters and 100 candidates for Haechi and Amort-Haechi.

Further size reduction with Amort-Haechi Recall that the amortized proof Π_1^{Am} (introduced in §6) uses a single proof to prove about all voters' ballots. As shown in Figure 8, in a finite field group instantiation with $m = 1M$ voters and $n = 100$ candidates, the total proof size reduces from 7.1GB with regular Haechi to 64MB with amortization. Including commitments, the total election record size a verifier has to download becomes 576MB in Amort-Haechi vs. 115GB in ElectionGuard, an $\sim 200\times$ improvement. Amort-Haechi under P-256 and P-384 instantiation also show $7\times$ improvement than Haechi, resulting in only 97MB and 145MB of total election record size.

	Record Generation Time Per Ballot (ms)					Verifier Time Per Ballot (ms)				
Number of Candidates	10	30	50	100	150	10	30	50	100	150
P256	18	33	59	115	221	7	14	27	54	108
P384	53	96	172	325	623	26	48	89	175	339
Finite Field	73	185	278	517	769	45	80	117	204	287
ElectionGuard*	174	526	874	1742	2632	291	877	1471	2908	4381

Table 2: The election record generation (prover) and verification time per ballot with Π_1 for varying number of candidates.

Time complexity Table 2 shows the time metrics of Haechi: election record generation time (commitment plus proof generation) and the proof verification time instantiated with the basic election rule Π_1 for varying

numbers of candidates. It takes 115ms and 325ms for generating an election record for $n = 100$ candidates with P-256 and P-384 respectively. The corresponding proof verification times are 54ms and 175ms respectively. The scheme instantiated with the ElectionGuard's finite field group gives 517ms for generating an election record and 204ms for verifying the proof under the same setting. These are approximately $\times 3$ and $\times 14$ improvements from ElectionGuard. The post-quantum instantiation of §9 is priced at the same $n = 100$ in §9.6, where its record sizes are set against the figures above.

9 Post-Quantum Instantiation of Haechi

Here, we present a post-quantum instantiation of Haechi. The voter-facing election flow does not change from the classical scheme: each voter still publishes a commitment to a selection vector together with a proof that the committed ballot is well formed, and the record is later aggregated into a public tally. The underlying cryptography, however, is different. The construction replaces the Pedersen commitment with the BDLOP lattice commitment [10, 50], the discrete-logarithm Σ-proof of ballot well-formedness with a lattice counterpart, and the classical Π_{Tally} proof of knowledge of an opening of the aggregate commitment with a tree-based aggregation of the commitments and proofs. As the flow is unchanged, the formal guarantees of §7 follow by the same arguments as in the classical scheme once the new components supply the corresponding hiding, binding, soundness and zero-knowledge properties, with computational hiding standing in for perfect hiding in the privacy hybrids. Therefore, we present each of these components, sketch their security, and then compute the parameters and record sizes of the resulting scheme.

9.1 Background

Notation. We follow the conventions of the lattice literature, which differ from those of the preceding sections in four places. In this section σ is a Gaussian width, so the commitment randomness is $\mathbf{r}$ (the letter Figure 2 uses for its mask), the mask is now γ, and the selection entries are x_i. The symbol k is the rank of the randomness rather than a voting limit, and κ is the weight of a challenge, not a bit-decomposition length. Finally, $\mathsf{Com}(\mathbf{y}; \mathbf{r})$ lists the message before the randomness. We write N_v for the number of ballots cast, D for the aggregation tree's maximum arity, d for a given node's arity and e for a Σ-protocol challenge.

The ring. Fix a power of two N, and let $R = \mathbb{Z}[X]/(X^N + 1)$ and $R_q = R/qR$ for a prime q. Elements are identified with their coefficient vectors, whose entries are represented in $(-q/2, q/2]$, and we write $S_\eta = \{y \in R : \|y\|_\infty \le \eta\}$ for the set of elements with small coefficients. Two standard norm inequalities are used without comment:

$$\|ab\|_2 \le \|a\|_1 \|b\|_2, \qquad \|a\|_2 \le \sqrt{N}\, \|a\|_\infty.$$

We will need $X^N + 1$ to split modulo q. Let N_{c} be a power of two with $2 \le N_{\mathsf{c}} \mid N$. If $q \equiv 2N_{\mathsf{c}} + 1 \pmod{4N_{\mathsf{c}}}$, then $X^N + 1$ factors modulo q into exactly N_{c} irreducibles of degree N/N_{c}, so that $R_q \cong \prod_{j \le N_{\mathsf{c}}} \mathbb{F}_{q^{N/N_{\mathsf{c}}}}$. Moreover, every y with $0 < \|y\|_\infty < q^{1/N_{\mathsf{c}}}/\sqrt{N_{\mathsf{c}}}$ is invertible [51, Cor. 1.2]. We write $y \mapsto (y^{(1)}, \ldots, y^{(N_{\mathsf{c}})})$ for this isomorphism and call $y^{(j)}$ the jth *slot*. Addition and multiplication are slotwise. Note that R_q is never a field at the degrees we use, because $X^N + 1$ is reducible modulo every prime once $N \ge 4$. Instead of working around this, §9.3 fixes $N_{\mathsf{c}} = 2$ and builds the ballot encoding directly on the two slots.

Challenges. Fix a weight $\kappa \le N$, and let $\mathcal{C} = \{c \in R_q : \|c\|_\infty = 1,\ \|c\|_1 = \kappa\}$ be the challenge set, so that $|\mathcal{C}| = \binom{N}{\kappa} 2^\kappa$. We also write $\bar{\mathcal{C}} = \{c - c' : c \ne c' \in \mathcal{C}\}$ for the set of challenge differences. Every element of $\bar{\mathcal{C}}$ has ℓ_∞ norm at most 2 and ℓ_1 norm at most 2κ, so as soon as $q^{1/N_{\mathsf{c}}} > 2\sqrt{N_{\mathsf{c}}}$ the whole of $\bar{\mathcal{C}}$ consists of units. A set whose pairwise differences are units is called *exceptional*, and this property is what makes interpolation work over a ring with zero divisors, where mere distinctness would not suffice. For an exceptional set, the Vandermonde determinant $\prod_{i<j}(a_j - a_i)$ is a unit, so a polynomial of degree at most d is determined by its values at $d + 1$ such points, and vanishes identically if all of them are zero. We use this twice in the extraction argument for Theorem 6.

Masking. Let $\mathcal{N}_\sigma^k$ be the discrete Gaussian of width σ on R^k. Here, the Gaussian plays the part of the uniform mask in a classical Σ-protocol. However, a Gaussian shifted by the witness is not the same as an unshifted one, so the prover applies rejection sampling [49, Thm. 4.6], aborting with a prescribed probability so that what it does send is statistically close to a distribution that does not depend on the witness at all.

For a witness of ℓ_2 norm at most T and width $\sigma = \alpha T$, rejection sampling outputs with probability at least $(1 - 2^{-100})/M$, where $M = \exp(12/\alpha + 1/(2\alpha^2))$. The expected number of attempts is essentially M, and the output lies within $2^{-100}/M$ of an ideal sampler that returns a draw from $\mathcal{N}_\sigma^k$ with probability $1/M$ and $\perp$ otherwise. We take $\alpha = 11$ throughout, which gives $M \approx 3$. We record three further facts, used silently in the proofs of Theorems 6 and 7:

- A sample from $\mathcal{N}_\sigma^1$ exceeds $2\sigma\sqrt{N}$ in ℓ_2 norm with probability below $2^{-1.16N}$.
- Any given coefficient of such a sample exceeds 6σ with probability below $2e^{-18} < 2^{-24.96}$ [49, Lem. 4.4].
- Conditioning two distributions at distance δ on an event of probability at least p leaves them within $2\delta/p$. This is the source of every 2^{-98} in the bounds of those theorems.

Hardness. Two problems take the place of the discrete logarithm. Informally, $\mathrm{MSIS}_{\nu,k,B}$ asks for a short nonzero combination of the columns of a public $\mathbf{A} \in R_q^{\nu \times k}$ that vanishes, with ℓ_2 bound B. The problem $\mathrm{MLWE}_{\mu,m,\chi}$ asks whether μ noisy linear equations in a secret of rank m, with secret and noise both drawn from χ, can be told apart from uniform data. One unimodular row operation puts our stacked matrix into Hermite normal form, with a leading identity block, so the instances we actually rely on are the corresponding *knapsack* ones. For M-LWE, the knapsack form is equivalent to the uniform one at the same parameters [10]. For M-SIS, binding gives a systematic rather than uniform instance, and it is that form whose hardness we assume and estimate. The worst-case reductions for module lattices [48] justify the general family, but they do not cover the narrow ternary distribution $\beta = 1$ that we instantiate. We treat these instances as assumptions in their own right, as is standard for practical lattice commitment schemes, and set parameters by direct cryptanalysis (§9.6).

9.2 The commitment scheme

The commitment scheme we need here is one in which *only the randomness* must be short, while the message may range over the whole ring. The reason is that the aggregate of all N_v ballots cast is nowhere near short, so a compressing, Ajtai-style commitment, whose binding also requires a short message, would cease to bind long before the tally. We commit to ballots with the scheme of Baum, Damgård, Lyubashevsky, Oechsner and Peikert [10], which plays the role of the Pedersen vector commitment of §3.

Setup and commitment. The public parameters are $\mathbf{A}_1 = [\,\mathbf{I}_\nu \mid \mathbf{A}_1'\,] \in R_q^{\nu \times k}$ and $\mathbf{A}_2 = [\,\mathbf{0} \mid \mathbf{I}_{2n+2} \mid \mathbf{A}_2'\,] \in R_q^{(2n+2)\times k}$ with the free blocks uniform and $k > \nu + 2n + 2$, generated from a public seed in the manner of the Pedersen bases of Appendix F. To commit to $\mathbf{y} \in R_q^{2n+2}$, one samples $\mathbf{r} \leftarrow\!\!\$\, S_\beta^k$ and outputs

$$\mathsf{Com}(\mathbf{y}; \mathbf{r}) = \begin{pmatrix} \mathsf{c}_1 \\ \mathsf{c}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix} \mathbf{r} + \begin{pmatrix} \mathbf{0}^\nu \\ \mathbf{y} \end{pmatrix}. \tag{2}$$

The scheme is additively homomorphic, and we take $\beta = 1$, so the randomness is ternary. The message enters unmultiplied, and only $\mathbf{r}$ is constrained in size, which is the property we asked for above.

Relaxed openings. Extraction from the proofs of §9.4 will be *relaxed*, meaning the extractor recovers an opening only of a scaled copy of the commitment, multiplied by a small invertible ring element. The notion of opening has to be relaxed to match, and the definition below additionally carries a *width*. Looking ahead, the extractor of the aggregation proof of §9.5 will recover openings of one commitment from two different proofs, whose responses are masked at two different Gaussian widths.

Definition 1 (Relaxed opening)**.** *A* relaxed opening at width w *of* $\mathsf{c} = (\mathsf{c}_1, \mathsf{c}_2)$ *is a triple* $(\boldsymbol{y}, \boldsymbol{r}, \bar{e})$ *with* $\boldsymbol{y} \in R_q^{2n+2}$, $\boldsymbol{r} \in R_q^k$ *and* $\bar{e} \in \bar{\mathcal{C}}$ *such that* $\bar{e}\,(\mathsf{c}_1; \mathsf{c}_2) = (\mathbf{A}_1; \mathbf{A}_2)\boldsymbol{r} + \bar{e}\,(\mathbf{0}^\nu; \boldsymbol{y})$ *and* $\|\boldsymbol{r}_i\|_2 \le 4w\sqrt{N}$ *for every* $i \in [k]$*, hence* $\|\boldsymbol{r}\|_2 \le 4w\sqrt{N}\sqrt{k}$.

In this section the width always takes one of two values, namely σ for openings extracted from ballot proofs and the larger $\sigma_D = \sigma\sqrt{D+1}$ for openings extracted from node proofs (§9.5). An honest commitment yields a relaxed opening at width σ for any $\bar{e} \in \bar{\mathcal{C}}$, since $\|\bar{e}\mathbf{r}_i\|_2 \le 2\kappa\beta\sqrt{N}$, which is at most $4\sigma\sqrt{N}$ whenever $\sigma \ge \kappa\beta/2$, as every width in this section is. Definition 1 is therefore a genuine relaxation of exact opening.

Binding and hiding. Both security properties survive the relaxation. For binding, two relaxed openings of one commitment to different messages, at widths w and w', combine into a nonzero element of the

kernel of $\mathbf{A}_1$ of ℓ_2 norm at most $8\kappa(w+w')\sqrt{N}\sqrt{k}$. To form it, multiply each opening by the other's relaxation factor and subtract. The difference is nonzero because applying $\mathbf{A}_2$ to it gives $\bar{e}\bar{e}'(\mathbf{y}-\mathbf{y}') \neq \mathbf{0}$, the factors being units. Binding at those widths therefore rests on M-SIS at that bound. Hiding follows from $\mathrm{MLWE}_{\nu+2n+2,\,k-\nu-2n-2,\,S_\beta}$, since a unimodular row operation puts $(\mathbf{A}_1;\mathbf{A}_2)$ into knapsack form and exhibits (2) as an M-LWE sample shifted by $\mathbf{y}$, making a commitment indistinguishable from uniform whatever it commits to. Note that hiding is *computational*, unlike the perfectly hiding Pedersen commitment used elsewhere in the paper. As §1 discusses, the instantiation trades everlasting ballot privacy for the post-quantum binding it buys.

9.3 Encoding ballots

Throughout this section we treat a single contest in which the voter selects exactly one of n options. This is the rule Π_1, which §8 prices classically at the same n, so the sizes in §9.6 can be compared with the classical ones directly. Several contests on one ballot, and cumulative voting, are accommodated by the same linear-equation mechanism as in the classical scheme, with bit-decomposition blocks for the cumulative rules.

Ring slots. Classically, well-formedness of $\mathbf{x} \in \mathbb{Z}_q^n$ is the requirement that $x_i(1-x_i)=0$ for all i together with $\sum_i x_i = 1$. Over R_q, these same equations admit more solutions. We fix $N_{\mathsf{c}} = 2$, the smallest value the ring permits, and let $\varepsilon \in R_q$ be the idempotent that is 1 in slot 1 and 0 in slot 2. The vector $(\varepsilon, 1-\varepsilon, 0, \ldots, 0)$ then satisfies both constraints while selecting option 1 in slot 1 and option 2 in slot 2, and since $N_{\mathsf{c}} \geq 2$ is forced, no choice of q avoids this. Both conditions are ring identities, so they hold if and only if they hold slotwise, and each Chinese-remainder slot carries its own copy of the constraint system. Since no ring identity admitting both 0 and 1 can force the x_i to be constants, all that the two conditions certify is that the ballot is one-hot in *every* slot, possibly at a different option in each, and the certified vectors form a strictly larger set than the honest encodings.

Pinning the encoding. We rule the extra solutions out by pinning the encoding to slot 1. With ε fixed at setup, the encoding is

$$\mathsf{Encode}(\mathsf{sel}) = \mathbf{x} \in R_q^n, \qquad x_i = \begin{cases} \varepsilon & \mathsf{sel} = i, \\ 0 & \text{otherwise}, \end{cases}$$

and Decode recovers sel by reading slot 1. The choice of ε is public and checkable. The idempotents at $N_{\mathsf{c}} = 2$ are exactly $0, 1, \varepsilon, 1-\varepsilon$, so checking $\varepsilon^2 = \varepsilon$ and $\varepsilon \notin \{0,1\}$ pins ε to one of the two slots, and which of the two is a labelling convention.

Definition 2 (Well-formed ballot)**.** $\boldsymbol{x} \in R_q^n$ *is* well formed *if* $x_i(1-x_i)=0$ *for all* $i \in [n]$ *and* $\sum_{i=1}^n x_i = \varepsilon$.

Provided $n < q$, the well-formed vectors are exactly the honest encodings. To see why, read the conditions slotwise. The first says that each $x_i^{(j)}$ is a bit (a slot is a field). The second says that those bits sum to 1 in slot 1 and to 0 in slot 2. Since a slot has characteristic $q > n$, a sum of n bits cannot wrap, so slot 1 is one-hot and slot 2 vanishes identically. Only the sum condition differs from the classical one, and an honest ballot still satisfies the quadratic, as $\varepsilon(1-\varepsilon) = 0$.

Arithmetization. Fix $2n+1$ interpolation nodes $\mathcal{X} = \{s_0, \ldots, s_{2n}\} \subseteq \mathcal{C}$, which is exceptional. The prover samples a mask $\tau \leftarrow\!\!\$\; R_q$, takes $f \in R_q[T]$ of degree at most n with $f(s_0) = \tau$ and $f(s_i) = x_i$ for $i \in [n]$, and sets $h(T) = f(T)(1-f(T))$, of degree at most $2n$. Then $h(s_i) = x_i(1-x_i)$, which vanishes on $[n]$ exactly when $\mathbf{x}$ satisfies the first half of Definition 2. The committed message is

$$\mathbf{y} = \big(x_1, \ldots, x_n,\ \tau,\ h(s_0), h(s_{n+1}), \ldots, h(s_{2n})\big), \tag{3}$$

where $h(s_1), \ldots, h(s_n)$ are omitted because they are hardwired to 0. For a first challenge c, public R_q-linear forms ℓ, ϕ_1, ϕ_2 read $\sum_i x_i$, $f(c)$ and $h(c)$ off $\mathbf{y}$ by Lagrange interpolation. Given claimed evaluations z_1, z_2 and a batching challenge ρ, both parties then form

$$L = \ell + \rho\,\phi_1 + \rho^2\phi_2, \qquad v = \varepsilon + z_1\rho + z_2\rho^2. \tag{4}$$

All three forms, and hence L, are *homogeneous*, i.e. $L(a\mathbf{w}) = aL(\mathbf{w})$, with the constant kept separate in v, just as in Figure 2. This split is essential here, since the relaxation factor cancels in extraction only when L is homogeneous. The relation the protocol establishes is

$$\mathcal{R}^{\Lambda} = \big\{(\mathsf{c}; \mathbf{y}, \mathbf{r}, \bar{e}) \;\big|\; (\mathbf{y}, \mathbf{r}, \bar{e}) \text{ opens } \mathsf{c} \text{ relaxed at width } \sigma \text{ and } (y_1, \ldots, y_n) \text{ is well formed}\big\}. \tag{5}$$

Note that the identity $L(\mathbf{y}) = v$ is not part of $\mathcal{R}^{\Lambda}$, as it is a consequence of a single transcript, not of the whole tree.

9.4 Proving well-formedness

Protocol Π_1^{Λ} (Figure 9) is the lattice counterpart of Π_1 (Figure 2). Its first four steps mirror the classical ones (the same two challenges, the same quadratic check $z_2 = z_1(1 - z_1)$, the same batching by powers of ρ), with the challenges now drawn from $\mathcal{C}$ rather than $\mathbb{Z}_q$ and the classical constant 1 replaced by ε. The last four steps replace the Schnorr argument of a Pedersen opening with the BDLOP Σ-protocol, adding norm checks and rejection sampling. The verification equation $L(\mathbf{A}_2\mathbf{z}) = u + e\,(L(\mathsf{c}_2) - v)$ plays the same role as the classical check. Since $\mathsf{c}_2 = \mathbf{A}_2\mathbf{r} + \mathbf{y}$ and L is linear, $L(\mathsf{c}_2) - v = L(\mathbf{A}_2\mathbf{r})$ holds exactly when $L(\mathbf{y}) = v$.

Protocol $\Pi_1^{\Lambda}\big(\mathsf{c}, \ell, \phi_1, \phi_2; \mathbf{y}, \mathbf{r}, f, h\big)$:

1: V: sends $c \leftarrow\!\!\$\; \mathcal{C} \setminus \{s_1, \ldots, s_n\}$ to P.
2: P: sends $z_1 \leftarrow f(c)$ and $z_2 \leftarrow h(c)$ to V.
3: V: checks $z_2 = z_1(1 - z_1)$ and sends $\rho \leftarrow\!\!\$\; \mathcal{C}$ to P.
4: P, V: form $L = \ell + \rho\phi_1 + \rho^2\phi_2$ and $v = \varepsilon + z_1\rho + z_2\rho^2$.
5: P: samples $\gamma \leftarrow\!\!\$\; \mathcal{N}_{\sigma}^{k}$ and sends $t \leftarrow \mathbf{A}_1\gamma$ and $u \leftarrow L(\mathbf{A}_2\gamma)$.
6: V: sends $e \leftarrow\!\!\$\; \mathcal{C}$ to P.
7: P: sets $\mathbf{z} \leftarrow \gamma + e\mathbf{r}$ and aborts with probability $1 - \min\big(1, \mathcal{N}_{\sigma}^{k}(\mathbf{z})/(M \cdot \mathcal{N}_{e\mathbf{r},\sigma}^{k}(\mathbf{z}))\big)$, and also if $\|\mathbf{z}\|_{\infty} > 6\sigma$. Otherwise it sends $\mathbf{z}$ to V.
8: V: accepts iff $\|\mathbf{z}_i\|_2 \le 2\sigma\sqrt{N}$ for all $i \in [k]$ and $\|\mathbf{z}\|_{\infty} \le 6\sigma$, $\mathbf{A}_1\mathbf{z} = t + e\,\mathsf{c}_1$, and $L(\mathbf{A}_2\mathbf{z}) = u + e\,(L(\mathsf{c}_2) - v)$.

Figure 9: The lattice proof of ballot well-formedness Π_1^{Λ}, establishing that c commits to a well-formed ballot in the sense of Definition 2. Compare Figure 2.

Theorem 6 (Ballot well-formedness)**.** *Let* $B_{\mathsf{bind}} = 16\kappa\sigma\sqrt{N}\sqrt{k}$ *with* $\sigma = 11\kappa\beta\sqrt{kN}$*, and assume* $n < q$*, which every parameter set of Table 3 satisfies. Protocol* Π_1^{Λ} *is*

(i) complete, an honest prover aborting with probability at most $1 - (1 - 2^{-100})/M + 2^{-98} + kN\,2^{-24.96}$ *and the verifier then rejecting with probability at most* $2^{-98} + k\,2^{-1.16N}$*.*

(ii) computationally $(2n+1, 3, 2)$*-special-sound for* $\mathcal{R}^{\Lambda}$ *of* (5)*, meaning that from* $2n+1$*,* 3 *and* 2 *differing challenges in the three respective rounds one can extract either a witness or a solution to* $\mathrm{MSIS}_{\nu,k,B_{\mathsf{bind}}}$*, assuming that problem is hard over* R_q*.*

(iii) computationally non-abort special honest-verifier zero-knowledge for honestly generated commitments, meaning that $(\mathsf{c}, \mathsf{tr})$ *is indistinguishable from* $(\mathsf{c}, \mathcal{S}(\mathsf{c}, c, \rho, e))$*, assuming* $\mathrm{MLWE}_{\nu+2n+2,\; k-\nu-2n-2,\; S_\beta}$ *is hard.*

The vector $\bar{\boldsymbol{x}}$ *extracted in (ii) is well formed in the sense of Definition 2, hence is the honest encoding of exactly one selection.*

Proof sketch. Completeness. Observe that $L(\mathbf{y}) = v$ and the homogeneity of L together give the third verification equation. The abort bound of (i) collects the rejection-sampling term $1 - (1 - 2^{-100})/M$ and the per-coefficient ℓ_∞ tail $kN\,2^{-24.96}$, while the rejection bound is the per-block ℓ_2 tail $k\,2^{-1.16N}$. Both tails are computed for the ideal $\mathcal{N}_{\sigma}^{k}$, and the 2^{-98} in each bound pays for the distance to the response the prover

actually sends. The prover pre-checks only ℓ_∞, so the ℓ_2 check can cause a rejection but never an abort. We keep the ℓ_∞ check because it lets us encode $\mathbf{z}$ in $\lceil \log_2(12\sigma+1) \rceil$ bits per coefficient, whereas the ℓ_2 check alone would admit coefficients as large as $2\sigma\sqrt{N}$. Aborts are harmless, since the prover simply restarts with a fresh mask, and the restarts leave no trace once Fiat–Shamir is applied.

Special soundness. We follow the classical special-soundness proof of Appendix B. Three things change: the extracted opening is relaxed rather than exact, uniqueness of the extracted message rests on M-SIS rather than on the discrete logarithm, and the interpolation steps need the relevant differences to be units of R_q rather than merely nonzero. Consider a node of the transcript tree at which (c, z_1, z_2, ρ) (and therefore L and v) are fixed, and take its two transcripts with challenges $e \neq e'$. Subtracting the $\mathbf{A}_1$ verification equations gives $\mathbf{A}_1\bar{\mathbf{r}} = \bar{e}\,\mathsf{c}_1$ for $\bar{e} = e - e' \in \bar{\mathcal{C}}$ and $\bar{\mathbf{r}} = \mathbf{z} - \mathbf{z}'$. Both responses pass the norm check, so $\bar{\mathbf{y}} = \mathsf{c}_2 - \bar{e}^{-1}\mathbf{A}_2\bar{\mathbf{r}}$ makes $(\bar{\mathbf{y}}, \bar{\mathbf{r}}, \bar{e})$ a relaxed opening at width σ. Subtracting the L equations and using homogeneity gives $L(\bar{\mathbf{y}}) = v$ with no relaxation factor attached, so the linear relation is recovered exactly even though the opening is relaxed. This is what homogeneity was needed for, and it is what lets a relaxed commitment sit inside an otherwise unchanged protocol. Each of the $3(2n+1)$ nodes yields such a triple, all of them openings of the same c, and two of them with different messages would give an M-SIS solution instead. Up to that loss, there is a single $\bar{\mathbf{y}}$ satisfying $L_{c,\rho}(\bar{\mathbf{y}}) = v_{c,\rho}$ everywhere.

From here the argument is the classical one, with exceptional sets in place of a field. Fix c. The degree-two polynomial $(\ell(\bar{\mathbf{y}}) - \varepsilon) + (\phi_1(\bar{\mathbf{y}}) - z_1)T + (\phi_2(\bar{\mathbf{y}}) - z_2)T^2$ vanishes at the three batching challenges, whose pairwise differences are units, so it is identically zero, giving $\ell(\bar{\mathbf{y}}) = \varepsilon$, $\phi_1(\bar{\mathbf{y}}) = z_1$ and $\phi_2(\bar{\mathbf{y}}) = z_2$. Write $\bar{\mathbf{y}} = (\bar{\mathbf{x}}, \bar{\mathbf{y}}_{n+1}, \ldots, \bar{\mathbf{y}}_{2n+2})$, and let $\hat{f}$, of degree at most n, and $\hat{h}$, of degree at most $2n$, interpolate the entries of $\bar{\mathbf{y}}$ over $\mathcal{X}$ in the same pattern as the honest f and h, with $\hat{h}(s_i) = 0$ hardwired for $i \in [n]$. Neither depends on c, so $\phi_1(\bar{\mathbf{y}}) = \hat{f}(c)$ and $\phi_2(\bar{\mathbf{y}}) = \hat{h}(c)$ identically in c. Step 3 then forces $z_2 = z_1(1 - z_1)$ at each of the $2n+1$ branches, so $\hat{h} - \hat{f}(1 - \hat{f})$, of degree at most $2n$, vanishes at $2n+1$ exceptional points and is therefore zero. Evaluating at s_i gives $\bar{x}_i(1 - \bar{x}_i) = 0$, and $\ell(\bar{\mathbf{y}}) = \varepsilon$ reads $\sum_i \bar{x}_i = \varepsilon$. So $\bar{\mathbf{x}}$ is well formed, with slot 1 one-hot and slot 2 vanishing.

Zero knowledge. The simulator samples z_1 uniformly, sets $z_2 = z_1(1 - z_1)$, draws $\mathbf{z}$ from $\mathcal{N}_\sigma^k$ conditioned on the verifier's two norm checks, and back-computes t and u. Rejection sampling puts the real accepted response within 2^{-98} of that distribution and makes it independent of $\mathbf{r}$, after which t and u are determined in both worlds. Conditioning both worlds on the ℓ_2 check contributes the $2k\,2^{-1.16N}$. The remaining term is $z_1 = f(c)$. Its coefficient of the mask τ is $\prod_{j \in [n]} (c - s_j)/(s_0 - s_j)$, a product of unit differences of elements of $\mathcal{C}$, so z_1 is a bijective function of τ. This is why Step 1 excludes $s_1, \ldots, s_n$. At $c = s_j$ the coefficient vanishes, and $z_1 = x_j$ would reveal a selection. However, the bijection does not suffice on its own, because the published commitment already fixes τ. We therefore replace c by a uniform element, which costs one M-LWE step. The response z_1 is then uniform and the transcript is exactly the simulator's. Switching the commitment back costs a second step. Every PPT distinguisher accordingly has advantage at most $2\,\mathrm{Adv}^{\mathrm{MLWE}} + 2^{-98} + 2k\,2^{-1.16N}$. □

By [7,8], an $(a_1, \ldots, a_r)$-special-sound protocol has knowledge error at most $\sum_i (a_i - 1)/|\mathcal{C}_i|$, with $\mathcal{C}_i$ the ith challenge set. Here this gives

$$\kappa_{\mathsf{err}} \leq \frac{2n}{|\mathcal{C}| - n} + \frac{3}{|\mathcal{C}|}, \tag{6}$$

up to the advantage of a PPT adversary against $\mathrm{MSIS}_{\nu,k,B_{\mathsf{bind}}}$. The cited bounds are stated for unconditional special soundness. The standard argument nevertheless covers our computational variant, since truncating the tree extractor at twice its expected running time turns any extraction failure beyond (6) into a strict-time M-SIS solver at a constant-factor loss. This is the direct analogue of Corollary 3 (the classical protocol has the same $(2n+1, 3, 2)$ structure). The denominator $|\mathcal{C}| - n$ reflects the points Step 1 excludes, and the second term collects the contributions $2/|\mathcal{C}|$ and $1/|\mathcal{C}|$ of the later rounds.

9.5 Aggregation: proving the tally

It remains to convince a verifier that the committed ballots sum to a public tally $\mathbf{T}$. Every well-formed ballot is a multiple of ε, so $\mathbf{T}_i = T_i\varepsilon$, where T_i is the count for option i and the second slot is zero. Since $N_v < q$ the counts are recovered unambiguously, and the public conditions $0 \leq T_i \leq N_v$, $\sum_{i \leq n} T_i = N_v$ and $\mathbf{T}_i^{(2)} = 0$ are checked directly.

Why the classical argument does not carry over. Classically, the device announces the tally and runs Π_{Tally} on the aggregate $\prod_i \mathsf{c}_i$, proving knowledge of the aggregate randomness and auxiliary values that open it to the announced result (Appendix A). BDLOP is equally homomorphic, and since only the randomness must be short, the tally entries are unconstrained in size, so one might hope to run the same one-shot proof over the aggregate. However, three obstacles block this direct route.

(i) *A preimage of the aggregate is not by itself an opening.* Classically, any pair satisfying the commitment equation opens the aggregate, so knowledge of one is already worth proving. Over BDLOP the kernel of $\mathbf{A}_1$ is public, so a preimage can be shifted by a kernel element into one that satisfies the equation against any tally one likes. A value counts as an opening only if it is short, so shortness has to become part of the proved statement.

(ii) *Proving the aggregate randomness short does not yield binding.* The difficulty is not in the masking. A sum of N_v independent ternary vectors concentrates, so the width needed is $\sqrt{2N_v/3}\,\sigma$, some 816 times a ballot's, and the responses stay far below $q/2$. The problem appears instead on the extraction side. An extractor for that proof bounds the randomness it recovers only at that width, so binding would rest on M-SIS at $\sqrt{2N_v/3}\,B_{\mathsf{bind}} \approx 2^{40.9}$ against a 34-bit modulus already at $n = 10$, and a bound above q is vacuous because the lattice contains $q\mathbb{Z}^{kN}$. Widening q would restore non-triviality ($B_{\mathsf{bind}} = 176\kappa^2\beta kN$ does not involve q), at the price of inflating every object in the election.

(iii) *Relaxed openings of distinct ballots do not merge.* Whatever the parameters, ballot i carries its own relaxation factor $\bar{e}_i$, read off its own transcript tree. Clearing denominators multiplies term i by $\prod_{l\neq i}\bar{e}_l$, which grows the witness by $(2\kappa)^{N_v-1}$ and attaches the factor $\prod_i \bar{e}_i$, in general no element of $\bar{\mathcal{C}}$. The merged equation is then no relaxed opening at any width.

The construction. We aggregate along a tree, refreshing the randomness at every node and tying each node to its children by a *single batched opening proof* that treats the node and all of its children as one witness (Figure 10). This addresses obstacles (iii) and (ii) at once. A single proof yields a single challenge, and a single relaxation factor shared by every term it must cancel against, so factors never have to merge inside a node. Moreover, the fresh witness is ternary again, so the $\sqrt{2N_v/3}$ that made the direct route vacuous becomes the $\sqrt{D+1}$ (under nine) that batching D children costs. This factor does not compound with depth, because each node proof is its own M-SIS instance, not one instance spanning the election.

Let $\mathbf{J}$ be the diagonal projector onto the first n coordinates, fix a maximum arity D, and let $\mathcal{T}$ be a tree of arity at most D whose leaves are the N_v accepted ballot commitments, each appearing once. Concretely, $\mathcal{T}$ is the complete D-ary tree over the accepted ballots sorted by identifier and filled left to right. Its shape is therefore a deterministic function of public data, so the tallier cannot signal through the grouping, and the verifier recomputes $\mathcal{T}$ and checks its leaf multiset against the ballots that passed Π_1^{Λ}.

For an internal node u with children $v_1, \ldots, v_d$, the tallier samples fresh $\mathbf{s}_u \leftarrow\!\!\$\ S_\beta^k$ and publishes $\mathsf{c}_u = \mathsf{Com}(\mathbf{J}\sum_j \mathbf{Y}_{v_j}; \mathbf{s}_u)$, where $\mathbf{Y}_v$ is the message of node v. The projector discards the $n+2$ auxiliary rows of (3), which are not part of the tally, and without it the root would also open to their aggregate, which is not simulatable from the result. The classical scheme must hide that aggregate behind the proof Π_{Tally}, whereas here the projector simply removes it. At the root, the tallier publishes $\mathbf{s}_{\mathsf{root}}$ outright, and the verifier checks the exact opening against $(\mathbf{T}, \mathbf{0}^{n+2})$ together with $\|\mathbf{s}_{\mathsf{root}}\|_\infty \le \beta$. The norm check is necessary here, since without it the public kernel of $\mathbf{A}_1$ would again admit any tally.

The node proof $\Pi^{\Lambda}_{\mathsf{agg},d}$ masks all $d+1$ blocks jointly at the single width $\sigma_D = \sigma\sqrt{D+1}$, fixed at the maximum arity. This is legitimate, because a shorter witness only improves the rejection-sampling acceptance rate, and the single width gives one binding bound throughout. Masking the blocks separately at width σ would instead accept with probability $M^{-(d+1)}$. Although masking and rejection sampling are joint, the verifier checks each child's shortness and the BDLOP relation separately. Throughout this subsection, a single subscript on a commitment indexes a node and never a component, so c_j is the jth child, with blocks $\mathsf{c}_{j,1}$ and $\mathsf{c}_{j,2}$.

Theorem 7 (Aggregation)**.** *Let $\sigma_D = \sigma\sqrt{D+1}$, let $B_{\mathsf{agg}} = 16\kappa\sigma_D\sqrt{N}\sqrt{k}$ and $B_{\mathsf{leaf}} = 8\kappa(\sigma+\sigma_D)\sqrt{N}\sqrt{k}$, let I be the number of internal nodes of $\mathcal{T}$, and assume $N_v < q$.*

(i) $\Pi^{\Lambda}_{\mathsf{agg},d}$ is complete, the prover aborting with probability at most $1-(1-2^{-100})/M + 2^{-98} + (d+1)kN\,2^{-24.96}$ and the verifier then rejecting with probability at most $2^{-98} + (d+1)k\,2^{-1.16N}$. It is 2-special-sound, the extractor returning relaxed openings of every c_j at width σ_D that share the same

Protocol $\Pi^{\Lambda}_{\mathsf{agg},d}\big(\mathsf{c}_0,\mathsf{c}_1,\dots,\mathsf{c}_d;\mathbf{r}_0,\mathbf{r}_1,\dots,\mathbf{r}_d\big)$:

1: P: samples $\gamma_0,\dots,\gamma_d \leftarrow\!\!\$\, \mathcal{N}^k_{\sigma_D}$ and sends $t_j \leftarrow \mathbf{A}_1\gamma_j$ for $j=0,\dots,d$ and $u \leftarrow \mathbf{A}_2\gamma_0 - \mathbf{J}\mathbf{A}_2\sum_{j\ge1}\gamma_j$.

2: V: sends $e \leftarrow\!\!\$\, \mathcal{C}$ to P.

3: P: sets $\mathbf{z}_j \leftarrow \gamma_j + e\mathbf{r}_j$ for $j=0,\dots,d$ and aborts with probability $1-\min\big(1,\mathcal{N}^{(d+1)k}_{\sigma_D}(\mathbf{z})/(M\cdot\mathcal{N}^{(d+1)k}_{e\mathbf{r},\sigma_D}(\mathbf{z}))\big)$ on the concatenation $\mathbf{z}=(\mathbf{z}_0,\dots,\mathbf{z}_d)$, and also if $\|\mathbf{z}\|_\infty > 6\sigma_D$. Otherwise it sends $\mathbf{z}_0,\dots,\mathbf{z}_d$ to V.

4: V: accepts iff (a) $\|\mathbf{z}_{j,i}\|_2 \le 2\sigma_D\sqrt{N}$ and $\|\mathbf{z}_j\|_\infty \le 6\sigma_D$ for all $j\in\{0,\dots,d\}$ and $i\in[k]$, (b) $\mathbf{A}_1\mathbf{z}_j = t_j + e\,\mathsf{c}_{j,1}$ for every $j\in\{0,\dots,d\}$, and (c) $\mathbf{A}_2\mathbf{z}_0 - \mathbf{J}\mathbf{A}_2\sum_{j\ge1}\mathbf{z}_j = u + e\big(\mathsf{c}_{0,2} - \mathbf{J}\sum_{j\ge1}\mathsf{c}_{j,2}\big)$.

Figure 10: The batched node proof $\Pi^{\Lambda}_{\mathsf{agg},d}$, run at every internal node of $\mathcal{T}$ with c_0 the node and $\mathsf{c}_1,\dots,\mathsf{c}_d$ its $d\le D$ children. The width $\sigma_D=\sigma\sqrt{D+1}$ is the same for every node. The checks are *blockwise*: in a norm check on a random linear combination of the blocks, large blocks could cancel, and a combined $\mathbf{A}_1$ check would extract no individual child's opening. The ℓ_∞ check is what makes the response encodable in $\lceil\log_2(12\sigma_D+1)\rceil$ bits per coefficient.

$\bar e$ and whose extracted messages satisfy $\boldsymbol{y}^\dagger_0 = \mathbf{J}\sum_{j\ge1}\boldsymbol{y}^\dagger_j$ exactly. Finally, it is statistically non-abort special honest-verifier zero-knowledge with simulation error at most $2^{-98}+2(d+1)k\,2^{-1.16N}$.

(ii) If every ballot proof, every node proof and the root opening verify, then $\boldsymbol{T}=\sum_{i\le N_v}\boldsymbol{x}_i$, with $\boldsymbol{x}_i$ the selection extracted from ballot i by Theorem 6(ii), and the counts it decodes to are correct over $\mathbb{Z}$, except with probability at most $N_v\kappa_{\mathsf{err}} + I/|\mathcal{C}| + N_v\mathrm{Adv}^{\mathrm{MSIS}}_{B_{\mathsf{bind}}} + N_v\mathrm{Adv}^{\mathrm{MSIS}}_{B_{\mathsf{leaf}}} + I\,\mathrm{Adv}^{\mathrm{MSIS}}_{B_{\mathsf{agg}}}$.

(iii) Given the tally $\boldsymbol{T}$, the tree $\mathcal{T}$ and the ballot commitments, the aggregation transcript is simulatable with advantage at most $(I-1)\mathrm{Adv}^{\mathrm{MLWE}} + I(2^{-98}+2(D+1)k2^{-1.16N})$. Extending the hybrid to honestly generated ballots makes the transcript of the whole election simulatable from $\boldsymbol{T}$ and the tree, at the further cost of $N_v(\mathrm{Adv}^{\mathrm{MLWE}}+2^{-98}+2k2^{-1.16N})$.

Proof sketch. For (i), honest inputs satisfy $\mathbf{y}_0=\mathbf{J}\sum_{j\ge1}\mathbf{y}_j$, so the message terms cancel between a node and its children, and the third verification equation concerns randomness alone. The concatenated witness lies in $S^{(d+1)k}_\beta$, so the shift $e\cdot(\mathbf{r}_0,\dots,\mathbf{r}_d)$ has norm at most $\kappa\beta\sqrt{(D+1)kN}=\sigma_D/11$, and rejection sampling applies in dimension $(d+1)kN$ at the same $\alpha=11$. For special soundness, two accepting transcripts with $e\ne e'$ give, writing $\bar e = e-e'\in\bar{\mathcal{C}}$, a vector $\mathbf{r}^\dagger_j=\mathbf{z}_j-\mathbf{z}'_j$ for each $j\in\{0,\dots,d\}$ satisfying $\mathbf{A}_1\mathbf{r}^\dagger_j=\bar e\,\mathsf{c}_{j,1}$, together with one further equation relating the node to its children. Setting $\mathbf{y}^\dagger_j=\mathsf{c}_{j,2}-\bar e^{-1}\mathbf{A}_2\mathbf{r}^\dagger_j$ and multiplying through by $\bar e^{-1}$ gives the claimed identity. The relaxation factor cancels because $\mathbf{J}$ and $\mathbf{A}_2$ are R_q-linear and the same $\bar e$ appears in every term, which is what batching was designed to ensure. Note that only each $\mathbf{r}^\dagger_j$, and never their sum, needs the norm bound, since the argument reconciles each c_j individually. Finally, zero knowledge is statistical here because every response in a transcript is a masked randomness term, whereas Π^{Λ}_1 had the message-dependent $z_1=f(c)$ and needed an M-LWE hop.

For (ii), we first explain where the five terms in the bound come from. The first, $N_v\kappa_{\mathsf{err}}$, is the union over ballots of the chance that a malicious one is accepted. The second, $I/|\mathcal{C}|$, is the union over node proofs of their knowledge error, as each is 2-special-sound and so contributes $1/|\mathcal{C}|$. The remaining three are M-SIS events. Two of them account for openings that fail to agree. Every commitment is opened twice, in its own proof and in its parent's, and a disagreement is an M-SIS solution at the corresponding bound. The third, $N_v\mathrm{Adv}^{\mathrm{MSIS}}_{B_{\mathsf{bind}}}$, is the collision a ballot extractor may output instead of a witness (Theorem 6(ii)).

In more detail, run the two extractors on all N_v+I proofs of Π^{Λ}_1 and $\Pi^{\Lambda}_{\mathsf{agg},d}$ under a sequential schedule, and condition on their success. Each proof is extracted as soon as its statement is fixed, so rewinding one session never alters another's statement, and the union bound over sessions is legitimate. Call a message *bound* at a commitment if every extracted relaxed opening of it, each taken at the width its own proof masks at, carries that message, and write $\bar{\mathbf{Y}}_u$ for that message. Each leaf has two extracted openings, one at width σ from its own proof and one at width σ_D from its parent's. A mismatch is an $\mathrm{MSIS}_{\nu,k,B_{\mathsf{leaf}}}$ solution. Each

non-root internal node likewise has two, both at width σ_D. A mismatch gives an $\mathrm{MSIS}_{\nu,k,B_{\mathsf{agg}}}$ solution. The root has its extracted opening together with the exact one the tallier published, and an exact opening is a relaxed one after scaling. Excluding all of these events, the message is well defined on every node and satisfies $\bar{\mathbf{Y}}_u = \mathbf{J}\sum_j \bar{\mathbf{Y}}_{v_j}$ at every internal node. Induction up the tree gives $\bar{\mathbf{Y}}_{\mathsf{root}} = (\sum_i \mathbf{x}_i, \mathbf{0}^{n+2})$, using that $\mathbf{J}$ is idempotent and that each leaf appears exactly once, which the verifier checks. Each entry is then $T_i\varepsilon$, with T_i the number of ballots selecting option i, so matching it against the published $\mathbf{T}_i$ determines T_i modulo q, hence exactly, since $T_i \le N_v < q$. The tally must then be correct over $\mathbb{Z}$, and not merely modulo q.

For (iii), two hybrid sequences are needed. A non-root node commitment appears in two proofs, its own and its parent's, so replacing it in a single step would leave the other proof unsimulated. The first sequence replaces all I node proofs by the simulator of (i), at the accumulated statistical cost. After it, no node proof depends on any $\mathbf{s}_u$ or on any ballot's randomness. The second sequence replaces the $I-1$ non-root node commitments by uniform elements, one at a time. Each hop is indistinguishable by hiding, since $\mathbf{s}_u$ is fresh and now used nowhere else, and the two proofs a commitment appears in are regenerated on the new statement. It is this second sequence that hides the internal subtotals. The root, whose message $(\mathbf{T}, \mathbf{0})$ the simulator forms directly, needs no replacement, which is why the count is $I-1$. The last statement extends the second sequence to the ballot commitments, at one M-LWE step per ballot. □

Deployment. Fiat–Shamir [38] is applied per proof. At a node, the hash must bind the election and node identifiers, the arity, and the child and node commitments, with the children in order as a length-prefixed list. Without this, the leaf-multiset check of Theorem 7(ii) would carry no weight. In a ballot proof, each of the three rounds needs its own tagged hash, or a prover could choose which round it is answering. Separately, completeness of $\Pi^{\Lambda}_{\mathsf{agg},d}$ needs *exact* ternary openings of every tallied ballot, not the relaxed openings the ballot proofs certify. In Haechi, the ballot-collecting device prepares each ballot itself (§4), so it has them, but it must retain each $\mathbf{r}_i$ until the node proof of that ballot's parent is produced. A per-device PRF seed, erased once the whole first level of node proofs is published, keeps this requirement constant-size. A remote-voting adaptation would have to enforce delivery of the exact opening at admission time, for instance by verifiable encryption of the opening to the tallier, since encryption alone would not suffice.

From interactive to compiled proofs. Theorems 6 and 7 are statements about the *interactive* protocols. Extracting from a compiled record, whose prover fixes every statement adaptively, not one proof at a time, requires a further assumption. This is standard for Fiat–Shamir records of this kind, but we do not establish it here. Classically, the compiled loss is benign, amounting to $N_v Q\kappa_{\mathsf{err}} + IQ/|\mathcal{C}|$ for Q random-oracle queries [8], which at $Q = 2^{64}$ and the parameters of Table 3 still leaves about a hundred bits of margin. In the quantum random oracle model, we can bound the three-move node proof, though the margin is small, provided it is a *quantum* proof of knowledge, which classical special soundness does not give. For the three-round ballot proof, the only generic route iterates measure-and-reprogram [36] once per round, which degrades as Q^6 and is vacuous at any usable Q. Enlarging $\mathcal{C}$ does not close the gap either, as the ternary weight-κ sets reach at most $\log_2 |\mathcal{C}| = 401.5$, leaving under 18 bits while growing σ fourfold. Since we know of no attack exploiting this gap, we regard the compiled ballot proof as conjecturally secure rather than as proved.

9.6 Parameters and sizes

We fix $\beta = 1$ and $N = 256$, and choose q, the M-SIS rank ν, the randomness rank k, the challenge weight κ and the arity D subject to the constraints collected above:

- *Well-posedness*: $k > \nu + 2n + 2$, so that the hiding secret has positive rank, and $n < q$ and $N_v < q$, which Theorems 6 and 7 assume.
- *Splitting and invertibility*: the conditions on q of §9.1.
- *Non-triviality*: $B_{\mathsf{bind}} < q$ and $B_{\mathsf{agg}} < q$, without which the M-SIS bounds are vacuous. The mixed bound B_{leaf} lies between the two and needs no constraint of its own.
- *Representability*: $12\sigma_D < q$, so that a node response does not wrap.
- *Knowledge error*: at most 2^{-190} in (6).

- *Security*: at least 128 bits of classical core-SVP security for the two binding instances and the hiding instance.

At $N_{\mathsf{c}} = 2$, the splitting condition reads $q \equiv 5 \pmod 8$ and invertibility $q > 8$, so neither costs anything. The knowledge error then fixes κ on its own. At $n = 10$ the least admissible weight is $\kappa = 40$, with $\log_2 |\mathcal{C}| = 196.20$, and it stays admissible only through $n = 35$, so the two wider columns take $\kappa = 41$. We target 2^{-190} rather than 2^{-128} because Fiat–Shamir and the union bound over $N_v + I$ proofs both reduce the margin.

Aggregation forces the modulus up and couples the parameters. Substituting $\sigma_D = \sigma\sqrt{D+1}$ into $B_{\mathsf{agg}} < q$ gives $D + 1 < \left(q/B_{\mathsf{bind}}\right)^2$, so whatever slack the binding constraint leaves below q is squared into the arity, and a parameter set tuned for cast-time cost alone leaves almost none. At $n = 100$, the smallest ballot satisfying the remaining constraints takes $\nu = 7$, $k = 216$ and a 34-bit q, where $q/B_{\mathsf{bind}} = 1.05$, so $D + 1 < 1.11$ and not even a binary tree fits. To account for aggregation, we raise the modulus by one to three bits, which enlarges the ballot itself by 5 to 7%.

	$n = 10$	$n = 50$	$n = 100$
ν	7	8	8
k	36	117	218
κ	40	41	41
$\lceil \log_2 q \rceil$	34	36	37
σ	$4.2 \cdot 10^4$	$7.8 \cdot 10^4$	$1.1 \cdot 10^5$
σ_D	$2.8 \cdot 10^5$	$6.0 \cdot 10^5$	$8.9 \cdot 10^5$
M-SIS, binding	192.7	215.2	207.6
M-SIS, aggregated	153.3	170.2	163.8
M-LWE, hiding	141.6	132.0	151.9
commitment	30.8 KiB	123.8 KiB	242.8 KiB
proof	32.0 KiB	85.5 KiB	155.8 KiB
ballot	**62.8 KiB**	**209.2 KiB**	**398.6 KiB**
arity D / depth	42 / 4	59 / 4	68 / 4
internal nodes	24,392	17,244	14,928
aggregation, per ballot	35.1 KiB	100.4 KiB	185.0 KiB
total per cast ballot	**97.9 KiB**	**309.7 KiB**	**583.6 KiB**

Table 3: Parameters and sizes at $N = 256$, $N_{\mathsf{c}} = 2$, $\beta = 1$ and $N_v = 10^6$ voters, for a single contest with n selectable options. Security rows are classical core-SVP bit costs from the `lattice-estimator` in its core-SVP mode. The moduli are the largest primes $q \equiv 5 \pmod 8$ below $2^{34}, 2^{36}, 2^{37}$. The horizontal rules separate parameters, cast-time cost, and the one-off aggregation of §9.5. The bold total row adds the ballot and its per-ballot aggregation share. Both layers count the whole Fiat–Shamir transcript, first-move messages included.

The security rows of Table 3 report classical core-SVP bit costs $0.292\,b$ [2, 3], the conventional way of quoting lattice security. We take them from the `lattice-estimator` [2] in its core-SVP mode, minimizing over the attack families that mode covers (lattice reduction for M-SIS, and primal-uSVP together with the dual hybrid for M-LWE), and we treat each module instance as an unstructured one. Every entry exceeds 128 bits, and the estimator's full cost model places each at least twenty-three bits higher, so the quoted figures are conservative. The two M-SIS rows differ by the $\sqrt{D+1}$ that the batched opening adds to the bound. A factor of six to nine in the norm is worth thirty-nine to forty-five bits of security here, which is why the modulus must accommodate the arity and not the ballot. The smallest estimate is M-LWE at $n = 50$, where the best attack is the dual hybrid at 132.0. This is above Kyber-512's published 118 in the same convention, and we read our target as the NIST level 1 that Kyber-512 meets.

The size rows count what a verifier downloads. Every R_q coefficient is packed in $\lceil \log_2 q \rceil$ bits, and every Gaussian response coefficient in $\lceil \log_2(12\sigma + 1) \rceil$ or $\lceil \log_2(12\sigma_D + 1) \rceil$ bits, the widths the ℓ_∞ checks permit, with no compression beyond this packing. Each layer is charged its whole Fiat–Shamir transcript, first-move messages included, and the aggregation rows spread the one-off cost of the tree, node commitments and proofs alike, evenly over the N_v ballots. The figures cover cast ballots and their proofs, which dominate the

record. As in the classical scheme, the full election record additionally holds the challenged ballots and their openings, in numbers set by how often voters challenge. The scripts that derive the parameters and carry out this size accounting are included in our open-science artifact.

A ballot costs 63 to 399 KiB to cast, and 98 to 584 KiB once its share of the aggregation is included, an overhead of 46 to 56%. The overhead is dominated by roughly one block of kN Gaussian coefficients per ballot, which the construction cannot avoid, since every ballot's randomness has to be proved known once inside some batch.

The evaluation of §8 prices the same rule Π_1 at the same $n = 100$, where a Haechi ballot record is 760 bytes with P-256 and 7,688 bytes in ElectionGuard's finite-field group, while ElectionGuard's own record is 115,264 bytes. Here the ballot costs 597,619 bytes, of which 408,160 is cast at the booth and 189,459 is the ballot's share of the aggregation. That is some 786 times the elliptic-curve record, 78 times Haechi in the same finite-field group, and about five times ElectionGuard itself. Only the commitment grows asymptotically rather than by a constant factor, because a Pedersen vector commitment holds the whole selection vector in one group element whatever n is, whereas BDLOP does not compress the message. Our commitment alone is 243 KiB, which is the price of the unbounded message space §9.5 needs to commit to a tally with entries as large as N_v. The aggregation material has no classical counterpart at all.

A more meaningful comparison, however, is against other post-quantum systems. PQKryvos [44] also commits with BDLOP and proves ballot well-formedness with Ligero. In our measurements with their artifact at $n = 100$, the commitment costs about 25 KiB and the proof 652 KiB, some 677 KiB in all, against 584 KiB here. That said, the figures are not like for like. PQKryvos additionally hides the tally from the public, a stronger goal that Haechi does not attempt, and it commits to far less, whereas our commitment carries the whole $(2n + 2)$-element arithmetization and our figure includes aggregation material for which PQKryvos has no counterpart. Nevertheless, the two systems arrive at the same order of magnitude by very different routes.

10 Discussion & Future Work

Haechi offers a departure from recent efforts at E2E-verifiable voting. By focusing on in-person voting and the device-centric model currently used in practice, Haechi eliminates the need for keys and election trustees, two persistent sources of difficulty in other systems. Replacing encryption with commitments also allows modern zero-knowledge techniques to shrink the election record substantially, removing yet another practical obstacle to verification. The post-quantum instantiation of §9 shows that this keyless, commitment-based workflow does not depend on the discrete-logarithm setting: it carries over to lattice commitments, although with substantially larger election records.

Haechi can also be used with VAULT-style risk-limiting audits (see [20] and [16]) to greatly simplify the process for privacy-preserving risk-limiting audits and make the size of the audit records far more manageable.

Future work will include an exploration of extending the approach to other voting methods such as ranked-choice voting. Methods to incorporate votes from other channels such as mail-in voting will also be explored.

Dispute resolution is another topic of future work. While verifiable election protocols are valuable even without dispute resolution (there is value in knowledge even without the ability to provide proof to third parties), good dispute resolution is very desirable. Recent work on dispute resolution protocols such as [17] offer hope that more practical and effective means of addressing this difficult problem can be found and incorporated within future versions of Haechi.

References

[1] Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In T. Moran D. Jefferson, J.L. Hall, editor, *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. Usenix, August 2009.

[2] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[3] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—a new hope. In *25th USENIX Security Symposium*, pages 327–343, 2016.

[4] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of ACM CCS 2017*, pages 2087–2104. ACM, 2017.

[5] Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In *Advances in Cryptology - CRYPTO 2020*, volume 12172 of *Lecture Notes in Computer Science*, pages 513–543. Springer, 2020.

[6] Thomas Attema, Ronald Cramer, and Serge Fehr. Compressing proofs of k-out-of-n partial knowledge. In *Annual International Cryptology Conference*, pages 65–91. Springer, 2021.

[7] Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed σ-protocol theory for lattices. In *Annual International Cryptology Conference*, pages 549–579. Springer, 2021.

[8] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs. In *Theory of Cryptography (TCC)*, pages 113–142, 2022.

[9] Jordi Barrat, Andrew Ellis, Graeme Orr, Vasil Vashchanka, and Peter Wolf. *Special Voting Arrangements*. International Institute for Democracy and Electoral Assistance, November 2023. `https://doi.org/10.31752/idea.2023.84`.

[10] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In *International Conference on Security and Cryptography for Networks*, pages 368–385. Springer, 2018.

[11] Carsten Baum, Ivan Damgård, Sabine Oechsner, and Chris Peikert. Efficient commitments and zero-knowledge protocols from ring-sis with applications to lattice-based threshold cryptosystems. *IACR Cryptol. ePrint Arch.*, 2016:997, 2016.

[12] Susan Bell, Josh Benaloh, Michael D Byrne, Dana DeBeauvoir, Bryce Eakin, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B Stark, Dan S Wallach, et al. STAR-Vote: A secure, transparent, auditable, and reliable voting system. In *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13)*, 2013.

[13] Jonathan Ben-Nun, Niko Fahri, Morgan Llewellyn, Ben Riva, Alon Rosen, Amnon Ta-Shma, and Douglas Wikström. A new implementation of a dual (paper and cryptographic) voting system. In *5th International Conference on Electronic Voting 2012 (EVOTE2012)*, pages 315–329. Gesellschaft für Informatik eV, 2012.

[14] Josh Benaloh. Simple verifiable elections. In *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'06)*, Vancouver, BC, Canada, 2006. USENIX Association. PDF: `https://www.usenix.org/legacy/event/evt06/tech/full_papers/benaloh/benaloh.pdf`.

[15] Josh Benaloh. Simple verifiable elections. *EVT*, 6:5–5, 2006.

[16] Josh Benaloh, Kammi Foote, Philip B. Stark, Vanessa Teague, and Dan S. Wallach. VAULT-style risk-limiting audits and the inyo county pilot. *IEEE Secur. Priv.*, 19(4):8–18, 2021.

[17] Josh Benaloh, Michael Naehrig, and Olivier Pereira. The DROP protocol: Dispute resolution via observation in public for verifiable, in-person voting. *Cryptology ePrint Archive*, 2025.

[18] Josh Benaloh, Michael Naehrig, Olivier Pereira, and Dan S. Wallach. ElectionGuard: a cryptographic toolkit to enable verifiable elections. In Davide Balzarotti and Wenyuan Xu, editors, *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. USENIX Association, 2024.

[19] Josh Benaloh, Ronald Rivest, Peter Y. A. Ryan, Philip Stark, Vanessa Teague, and Poorvi Vora. End-to-end verifiability. *arXiv preprint arXiv:1504.03778*, 2015.

[20] Josh Benaloh, Philip B. Stark, and Vanessa Teague. VAULT: Verifiable audits using limited transparency. In *E-Vote-ID: International Joint Conference on Electronic Voting*, October 2019. Also available as a preprint: `https://www.stat.berkeley.edu/~stark/Preprints/vault19.pdf`.

[21] Josh Daniel Cohen Benaloh. *Verifiable secret-ballot elections*. Yale University, 1987.

[22] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. SoK: A comprehensive analysis of game-based ballot privacy definitions. In *2015 IEEE Symposium on Security and Privacy, SP 2015*, pages 499–516. IEEE Computer Society, 2015.

[23] Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. Bingo voting: Secure and coercion-free voting using a trusted random number generator. In *International Conference on E-Voting and Identity*, pages 111–124. Springer, 2007.

[24] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.

[25] David Chaum, Richard T Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L Rivest, Peter Y. A. Ryan, Emily Shen, Alan T Sherman, and Poorvi L Vora. Scantegrity II: End-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE transactions on information forensics and security*, 4(4):611–627, 2009.

[26] David Chaum, Peter YA Ryan, and Steve Schneider. A practical voter-verifiable election scheme. In *European Symposium on Research in Computer Security*, pages 118–139. Springer, 2005.

[27] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachene. Election verifiability for helios under weaker trust assumptions. In *European Symposium on Research in Computer Security*, pages 327–344. Springer, 2014.

[28] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. Sok: Verifiability notions for e-voting protocols. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 779–798. IEEE, 2016.

[29] Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. Belenios: A simple private and verifiable electronic voting system. In *Foundations of Security, Protocols, and Equational Reasoning - Essays Dedicated to Catherine A. Meadows*, volume 11565 of *LNCS*, pages 214–238. Springer, 2019.

[30] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*, pages 174–187. Springer, 1994.

[31] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-autority secret-ballot elections with linear work. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 1996.

[32] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 1997.

[33] Chris Culnane, Peter Y. A. Ryan, Steve A. Schneider, and Vanessa Teague. vvote: A verifiable voting system. *ACM Trans. Inf. Syst. Secur.*, 18(1):3:1–3:30, 2015.

[34] Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In *Computer Security - ESORICS 2013*, volume 8134 of *Lecture Notes in Computer Science*, pages 481–498. Springer, 2013.

[35] Rafael del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical quantum-safe voting from lattices. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1565–1581. ACM, 2017.

[36] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat–Shamir transformation in the quantum random-oracle model. In *Advances in Cryptology – CRYPTO 2019*, pages 356–383. Springer, 2019.

[37] A. Faz-Hernandez, S. Scott, N. Sullivan, R. S. Wahby, and C. A. Wood. RFC 9380: Hashing to Elliptic Curves. RFC 9380, Internet Research Task Force (IRTF), August 2023. Informational RFC.

[38] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[39] Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016.

[40] Feng Hao, Matthew N. Kreeger, Brian Randell, Dylan Clarke, Siamak F. Shahandashti, and Peter Hyun-Jeen Lee. Every vote counts: Ensuring integrity in large-scale electronic voting. *Journal of Election Technology and Systems (JETS)*, 2(3):1–25, aug 2014.

[41] Max Harrison and Thomas Haines. On the applicability of STARKs to counted-as-collected verification in existing homomorphic e-voting systems. In *Financial Cryptography and Data Security. FC 2024 International Workshops - Voting, DeFI, WTSC, CoDecFin*, volume 14746 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2024.

[42] Max Hoffmann, Michael Klooß, and Andy Rupp. Efficient zero-knowledge arguments in the discrete log setting, revisited. In *Proceedings of the 2019 ACM CCS 2019*, pages 2093–2110. ACM, 2019.

[43] Nicolas Huber, Ralf Küsters, Toomas Krips, Julian Liedtke, Johannes Müller, Daniel Rausch, Pascal Reisert, and Andreas Vogt. Kryvos: Publicly tally-hiding verifiable e-voting. In *Proceedings of ACM CCS 2022*, pages 1443–1457. ACM, 2022.

[44] Nicolas Huber, Ralf Küsters, and Pascal Reisert. PQKryvos: Post-quantum secure e-voting with flexible ballot formats and public tally-hiding. *Proceedings on Privacy Enhancing Technologies*, 2026.

[45] IACR News: Election 2025 update. `https://www.iacr.org/news/item/27138`, 2025.

[46] Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *European Symposium on Research in Computer Security*, pages 389–404. Springer, 2010.

[47] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: definition and relationship to verifiability. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 526–535, 2010.

[48] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[49] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 738–755. Springer, 2012.

[50] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In *IACR International Conference on Public-Key Cryptography*, pages 215–241. Springer, 2021.

[51] Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. Cryptology ePrint Archive, Paper 2017/523, 2017.

[52] Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer, 2006.

[53] Siamak F Shahandashti and Feng Hao. Dre-ip: a verifiable e-voting scheme without tallying authorities. In *European Symposium on Research in Computer Security*, pages 223–240. Springer, 2016.

[54] The New York Times. Cryptography group lost election results, November 2025.

A Tally Correctness Proof

In this section, we provide the proof scheme that proves the tally result is correct. Given the aggregation of commitments of all cast ballots

$$C = \prod_{i=1}^{m} \mathsf{c}_{\mathbf{y}_i}$$

where m is the number of cast ballots, the proof scheme Π_{Tally} is defined over the following relation,

$$\mathcal{R}_{\mathsf{Tally}} = \{(C, \alpha_{\mathbf{x}}; \alpha_\sigma, \alpha_{\mathsf{aux}}) : C = \mathsf{Com}(\alpha_\sigma, (\alpha_{\mathbf{x}}, \alpha_{\mathsf{aux}}))\}, \tag{7}$$

where $\alpha_{\mathbf{x}}, \alpha_\sigma, \alpha_{\mathsf{aux}}$ is the aggregated vote values, aggregated randomness, and aggregated auxiliary values $(f(0), h(0), h(n+1), \cdots, h(2n))$.

By additive homomorphic property of the Pedersen commitment scheme, if the prover (device) has computed the tally correctly, then

$$C = g^{\alpha_\sigma} \cdot \prod_{i=1}^{n} g_i^{\alpha_{\mathbf{x},i}} \cdot \prod_{i=n+1}^{2n+2} g_i^{\alpha_{\mathsf{aux},i}}.$$

Rather than revealing $(\alpha_\sigma, \alpha_{\mathsf{aux}})$, we prove the knowledge of $(\alpha_\sigma, \alpha_{\mathsf{aux}})$ that satisfies 7. Similar to the proof of ballot well-formedness, we use standard Compressed-Σ proof. The protocol is described in Figure 11.

Protocol $\Pi_{\mathsf{Tally}}(C, \alpha_{\mathbf{x}}; \alpha_\sigma, \alpha_{\mathsf{aux}})$:

1: P: samples $(\gamma, a_{n+1}, \cdots, a_{2n+2}) \leftarrow\!\!\$\ \mathbb{Z}_q^{n+3}$ and sends $A \leftarrow g^\gamma \prod_{i=n+1}^{2n+2} g_i^{a_i}$

2: V: samples the challenge and sends $e \leftarrow\!\!\$\ \mathbb{Z}_q$

3: P: computes $\zeta = \gamma + e \cdot \alpha_\sigma$, $z_k \leftarrow \tilde{a_k} + e \cdot \alpha_{\mathsf{aux},k}$ for $k \in [n+1, 2n+2]$ and sends $(\zeta, z_{n+1}, \cdots, z_{2n+2})$

4: V: verifies $g^\zeta \cdot \prod_{k=n+1}^{2n+2} g_k^{z_k} \stackrel{?}{=} A \cdot (C / \prod_{i=n+1}^{2n+2} g_i^{\alpha_{\mathbf{x},i}})^e$

Figure 11: Zero-knowledge proof protocol Π_{Tally} proving (7).

The proof size of Π_{Tally} is $1\mathbb{G} + (n+3)\mathbb{Z}_q$. Notice that a single tally proof is attached to the bulletin board during the tally procedure. Using the same compression technique as Π_1, we achieve the proof size with $2(\lceil \log(n+3) \rceil - 1)\mathbb{G} + 2\mathbb{Z}_q$.

Theorem 8. Π_{Tally} *is perfectly complete, special honest-verifier zero-knowledge, and computationally* 2*-special-sound under the discrete logarithm assumption.*

Proof. **(Perfect completeness.)** Suppose that

$$(C, \alpha_{\mathbf{x}}; \alpha_\sigma, \alpha_{\mathsf{aux}}) \in \mathcal{R}_{\mathsf{Tally}}.$$

Then, an honestly generated response satisfies

$$\begin{aligned} g^\zeta \prod_{k=n+1}^{2n+2} g_k^{z_k} &= g^{\gamma + e\alpha_\sigma} \prod_{k=n+1}^{2n+2} g_k^{a_k + e\alpha_{\mathsf{aux},k}} \\ &= \left(g^\gamma \prod_{k=n+1}^{2n+2} g_k^{a_k} \right) \left(g^{\alpha_\sigma} \prod_{k=n+1}^{2n+2} g_k^{\alpha_{\mathsf{aux},k}} \right)^e \\ &= A \cdot \left(\frac{C}{\prod_{i=1}^{n} g_i^{\alpha_{\mathbf{x},i}}} \right)^e. \end{aligned}$$

Hence, the verifier always accepts an honest prover.

(2-special soundness.) Suppose that there are two accepting transcripts with the same first message A and distinct challenges $e \neq e'$:

$$(A, e, \zeta, z_{n+1}, \ldots, z_{2n+2})$$

and

$$\left(A, e', \zeta', z'_{n+1}, \ldots, z'_{2n+2}\right).$$

Let's denote

$$\widehat{C} = g^{\alpha_\sigma} \prod_{k=n+1}^{2n+2} g_k^{\alpha_{\mathsf{aux},k}}.$$

It is easy to see that $\widehat{C} = C / \prod_{i=1}^{n} g_i^{\alpha_{\mathbf{x},i}}$.

Since both transcripts are accepting,

$$g^{\zeta} \prod_{k=n+1}^{2n+2} g_k^{z_k} = A\widehat{C}^{e}, \tag{8}$$

$$g^{\zeta'} \prod_{k=n+1}^{2n+2} g_k^{z'_k} = A\widehat{C}^{e'}. \tag{9}$$

Dividing (8) by (9) gives

$$g^{\zeta-\zeta'} \prod_{k=n+1}^{2n+2} g_k^{z_k - z'_k} = \widehat{C}^{e-e'}.$$

Because q is prime and $e \neq e'$, the value $e - e'$ is invertible in $\mathbb{Z}_q$. The extractor computes

$$\alpha^*_\sigma := \frac{\zeta - \zeta'}{e - e'} \pmod{q} \tag{10}$$

and, for every $k \in \{n+1, \ldots, 2n+2\}$,

$$\alpha^*_{\mathsf{aux},k} := \frac{z_k - z'_k}{e - e'} \pmod{q}. \tag{11}$$

Thus,

$$g^{\alpha^*_\sigma} \prod_{k=n+1}^{2n+2} g_k^{\alpha^*_{\mathsf{aux},k}} = \widehat{C}.$$

Using the definition of $\widehat{C}$, we obtain

$$C = \widehat{C} \cdot \prod_{i=1}^{n} g_i^{\alpha_{\mathbf{x},i}} = g^{\alpha^*_\sigma} \prod_{i=1}^{n} g_i^{\alpha_{\mathbf{x},i}} \prod_{k=n+1}^{2n+2} g_k^{\alpha^*_{\mathsf{aux},k}}.$$

Thus,

$$(\alpha^*_\sigma, \alpha^*_{\mathsf{aux}})$$

is a valid witness for the statement $(C, \alpha_{\mathbf{x}})$ in $\mathcal{R}_{\mathsf{Tally}}$.

We now show that a prover cannot produce an accepting proof for an incorrect $\alpha_{\mathbf{x}}$ without finding an alternative opening of C. Let

$$C = \mathsf{Com}\left(\alpha^\star_\sigma, (\alpha^\star_{\mathbf{x}}, \alpha^\star_{\mathsf{aux}})\right)$$

be the opening of C obtained by homomorphically aggregating the openings of the cast-ballot commitments. Suppose that a probabilistic polynomial-time prover produces an accepting proof for a claimed tally $\alpha_{\mathbf{x}} \neq \alpha^\star_{\mathbf{x}}$ with probability non-negligibly greater than the knowledge error. By the knowledge soundness of Π_{Tally}, an extractor obtains values $(\alpha_\sigma, \alpha_{\mathsf{aux}})$ such that

$$C = \mathsf{Com}\left(\alpha_\sigma, (\alpha_{\mathbf{x}}, \alpha_{\mathsf{aux}})\right).$$

Consequently, C admits the two valid openings

$$(\alpha_\sigma^\star, (\alpha_{\mathbf{x}}^\star, \alpha_{\mathsf{aux}}^\star)) \quad \text{and} \quad (\alpha_\sigma, (\alpha_{\mathbf{x}}, \alpha_{\mathsf{aux}}))\,.$$

These openings are distinct because $\alpha_{\mathbf{x}} \neq \alpha_{\mathbf{x}}^\star$. Therefore, the prover, together with the knowledge extractor, yields a violation of the computational binding property of the Pedersen vector commitment. It follows that, assuming the commitment scheme is computationally binding, no probabilistic polynomial-time prover can convince the verifier of a tally $\alpha_{\mathbf{x}} \neq \alpha_{\mathbf{x}}^\star$ with probability non-negligibly greater than the knowledge error of Π_{Tally}.

(Special honest-verifier zero-knowledge.) We construct a simulator $\mathcal{S}_{\mathsf{Tally}}$ that receives only the public statement $(C, \alpha_{\mathbf{x}})$ and does not know $(\alpha_\sigma, \alpha_{\mathsf{aux}})$.

The simulator first computes

$$\widehat{C} = \frac{C}{\prod_{i=1}^{n} g_i^{\alpha_{\mathbf{x},i}}}.$$

It then samples

$$e, \zeta, z_{n+1}, \ldots, z_{2n+2} \leftarrow_{\$} \mathbb{Z}_q$$

uniformly and independently, and defines

$$A := g^\zeta \prod_{k=n+1}^{2n+2} g_k^{z_k} \cdot \widehat{C}^{-e}. \tag{12}$$

Finally, it outputs

$$(A, e, \zeta, z_{n+1}, \ldots, z_{2n+2})\,.$$

By construction,

$$g^\zeta \prod_{k=n+1}^{2n+2} g_k^{z_k} = A\widehat{C}^e,$$

so the simulated transcript is always accepting.

It remains to show that the simulated and real transcripts have the same distribution. In a real execution, for every fixed challenge e,

$$\zeta = \gamma + e\alpha_\sigma$$

and

$$z_k = a_k + e\alpha_{\mathsf{aux},k}.$$

Because γ and the a_k's are sampled uniformly from $\mathbb{Z}_q$, and addition by a fixed value is a permutation of $\mathbb{Z}_q$, the values

$$\zeta, z_{n+1}, \ldots, z_{2n+2}$$

are uniformly and independently distributed, even conditioned on e.

Furthermore, the verification equation uniquely determines the first message as

$$A = g^\zeta \prod_{k=n+1}^{2n+2} g_k^{z_k} \cdot \widehat{C}^{-e}.$$

Therefore, the simulated transcript is identically distributed to a transcript from an honest execution. Hence, Π_{Tally} is perfect honest-verifier zero knowledge. □

B Proof of Theorem 1

Proof. **(Completeness).** Completeness follows directly from the protocol.

(Special Soundness). Recall that a public-coin protocol is $(k_1, \cdots, k_\mu)$-special sound, if there exists a polynomial time algorithm that on input a statement and a $(k_1, \cdots, k_\mu)$-tree of accepting transcripts, outputs a witness for the statement. In our protocol, the statement is $\mathsf{c}_{\mathbf{y}}, \ell, \phi_1, \phi_2$. We divide our protocol

of Π_1 in Figure 2 into two parts: step 1-3, step 4-5, and step 6-9 and show that each part satisfies $2n+1$, 3, and 2-special sound.

(i) We start from the last part. For fixed $(c, (z_1, z_2), \rho)$, we show that there exists an efficient extract $\mathcal{E}_3$ that given two accepting transcripts $((t, A), c', (\mathbf{z}', \zeta'))$ and $((t, A), c^*, (\mathbf{z}^*, \zeta^*))$ extracts the witness $(\mathbf{y}, \sigma)$. From the accepting transcripts, $\mathcal{E}_3$ has the following equations.

$$\begin{aligned} g^{\zeta'} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{z}'_i} \cdot h^{L(\mathbf{z}')} &= \mathsf{c}_{\mathbf{y}}^{c'} \cdot A \cdot h^{c' \cdot (1 + z_1\rho + z_2\rho^2) + t} \\ &= (g^{\sigma} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{y}_i})^{c'} \cdot A \cdot h^{c' \cdot (1 + z_1\rho + z_2\rho^2) + t} \\ g^{\zeta^*} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{z}^*_i} \cdot h^{L(\mathbf{z}^*)} &= \mathsf{c}_{\mathbf{y}}^{c^*} \cdot A \cdot h^{c^* \cdot (1 + z_1\rho + z_2\rho^2) + t} \\ &= (g^{\sigma} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{y}_i})^{c^*} \cdot A \cdot h^{c^* \cdot (1 + z_1\rho + z_2\rho^2) + t}. \end{aligned}$$

Thus, by dividing the first equation by the second equation, $\mathcal{E}_3$ gets

$$g^{\zeta' - \zeta^*} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{z}'_i - \mathbf{z}^*_i} = (g^{\sigma} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{y}_i})^{c' - c^*}. \tag{13}$$

We show that given this, $\mathcal{E}_3$ either finds non-trivial discrete logarithm relations between generators or from

$$\zeta' - \zeta^* = (c' - c^*) \cdot \sigma \text{ and } \mathbf{z}'_i - \mathbf{z}^*_i = (c' - c^*)\mathbf{y}_i,$$

outputs

$$\sigma \leftarrow (\zeta' - \zeta^*)/(c' - c^*) \text{ and } \mathbf{y}_i \leftarrow (\mathbf{z}'_i - \mathbf{z}^*_i)/(c' - c^*).$$

In other words, we show that if an extractor $\mathcal{E}_3$ outputs $(\sigma', \mathbf{y}') \neq (\sigma, \mathbf{y})$, then it knows the discrete logarithm relation between generators. Consider an adversarial extractor $\mathcal{E}_3^*$ that, given two accepting transcripts, tries to output a set of witnesses $(\sigma', \mathbf{y}')$ such that $(\sigma', \mathbf{y}') \neq (\sigma, \mathbf{y})$ under the discrete logarithm assumption.

First, suppose $\mathcal{E}_3^*$ outputs $(\sigma', \mathbf{y}')$ such that only one element is different from $(\sigma, \mathbf{y})$. Without loss of generality, suppose that $\sigma' \neq \sigma$ but $\mathbf{y}' = \mathbf{y}$. Then,

$$\begin{aligned} g^{\zeta' - \zeta^*} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{z}'_i - \mathbf{z}^*_i} &= (g^{\sigma} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{y}_i})^{c' - c^*} \\ &= (g^{\sigma} \cdot \prod_{i=1}^{2n+2} g_i^{\frac{\mathbf{z}'_i - \mathbf{z}^*_i}{c' - c^*}})^{c' - c^*} \\ &= g^{\sigma \cdot (c' - c^*)} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{z}'_i - \mathbf{z}^*_i}. \end{aligned}$$

Since the term $\prod_{i=1}^{2n+2} g_i^{\mathbf{z}'_i - \mathbf{z}^*_i}$ cancels out, we get $\sigma = (\zeta' - \zeta^*)/(c' - c^*)$, which contradicts the assumption. This means that at least two elements of witnesses need to be different.

We show that $\mathcal{E}_3^*$ cannot output such $(\sigma', \mathbf{y}')$ with more than negligible probability. Consider a game $\mathsf{ExtDiff}$ in Figure 12 where an adversarial extractor $\mathcal{E}_3^*$ wins if it can output witnesses $(\sigma', \mathbf{y}')$ that have two or more different elements than $(\sigma, \mathbf{y})$. We show a reduction between $\mathsf{ExtDiff}$ and the discrete log game DL and prove that the advantage of $\mathcal{E}_3^*$ against $\mathsf{ExtDiff}$ is upper-bounded by the advantage of $\mathcal{A}$ against DL. The reduction is described in Figure 13. The adversary $\mathcal{A}$ is given the set of generators $(g, g_1, \cdots, g_{2n+2})$ where $g_i = g^{d_i}$ and wins if it outputs at least one d_i correctly. $\mathcal{A}$ uses $\mathcal{E}_3^*$ as a subroutine and acts as the challenger

Game $\mathsf{ExtDiff}^{\mathcal{E}_3^*}(\lambda)$

1: $((z_1, z_2), c, (t, A), c', (\mathbf{z}', \zeta')) \leftarrow \Pi_1(\mathsf{c}_\mathbf{y}, \ell, \phi_1, \phi_2; \sigma, \mathbf{y}, f, h)$
2: $((z_1, z_2), c, (t, A), c^*, (\mathbf{z}^*, \zeta^*)) \leftarrow \Pi_1(\mathsf{c}_\mathbf{y}, \ell, \phi_1, \phi_2; \sigma, \mathbf{y}, f, h)$
3: $\mathsf{tr}_1 \leftarrow ((t, A), c', (\mathbf{z}', \zeta'))$, $\mathsf{tr}_2 \leftarrow ((t, A), c^*, (\mathbf{z}^*, \zeta^*))$
4: $(\sigma', \mathbf{y}') \leftarrow \mathcal{E}_3^*(\mathsf{tr}_1, \mathsf{tr}_2)$
5: **return** $(\sigma', \mathbf{y}') \neq (\sigma, \mathbf{y})$

Figure 12: The game ExtDiff where the adversarial extractor $\mathcal{E}_3^*$ wins if it outputs a set of witnesses that is different from the witnesses used for the transcripts. Since we consider the step 4 to 9 of Π_1 in this proof, we pass only the corresponding transcripts to $\mathcal{E}_3^*$.

of ExtDiff; it generates two accepting transcripts and invokes $\mathcal{E}_3^*$ with $(\mathsf{tr}_1, \mathsf{tr}_2)$. When $\mathcal{E}_3^*$ outputs $(\sigma', \mathbf{y}')$, $\mathcal{A}$ gets d_i' by

$$d_i' = -\frac{\sigma' \cdot (c' - c^*) - (\zeta' - \zeta^*)}{\mathbf{y}_i \cdot (c' - c^*) - (\mathbf{z}_i' - \mathbf{z}_i^*)},$$

and outputs $(d_1', \cdots, d_{2n+2}')$.

Game $\mathsf{DL}^{\mathcal{A}}(\lambda)$:

1: $(d_1, \cdots, d_{2n+2}) \leftarrow_\$ \mathbb{Z}_q^{2n+2}$
2: $\forall i \in [2n+2]$, $g_i \leftarrow g^{d_i}$
3: $(d_1', \cdots, d_{2n+2}') \leftarrow \mathcal{A}(g, g_1, \cdots, g_{2n+2})$
4: **return** $\exists d_i' = d_i$ for $i \in [2n+2]$

$\mathcal{A}(g, g_1, \cdots, g_{2n+2})$:

1: $((z_1, z_2), c, (t, A), c', (\mathbf{z}', \zeta')) \leftarrow \Pi_1(\mathsf{c}_\mathbf{y}, \ell, \phi_1, \phi_2; \mathbf{y}, \sigma, f, h)$
2: $((z_1, z_2), c, (t, A), c^*, (\mathbf{z}^*, \zeta^*)) \leftarrow \Pi_1(\mathsf{c}_\mathbf{y}, \ell, \phi_1, \phi_2; \mathbf{y}, \sigma, f, h)$
3: $\mathsf{tr}_1 \leftarrow ((t, A), c', (\mathbf{z}', \zeta'))$, $\mathsf{tr}_2 \leftarrow ((t, A), c^*, (\mathbf{z}^*, \zeta^*))$
4: $(\sigma', \mathbf{y}') \leftarrow \mathcal{E}_3^*(\mathsf{tr}_1, \mathsf{tr}_2)$
5: $\forall i \in [2n+2]$,
6: **if** $\mathbf{y}_i' \neq \mathbf{y}_i$: $d_i' \leftarrow -\frac{\sigma' \cdot (c' - c^*) - (\zeta' - \zeta^*)}{\mathbf{y}_i' \cdot (c' - c^*) - (\mathbf{z}_i' - \mathbf{z}_i^*)}$
7: **else** : $d_i' \leftarrow 0$
8: **return** $(d_1', \cdots, d_{2n+2}')$

Figure 13: The reduction between DL and ExtDiff.

Now suppose that $\mathcal{E}_3^*$ wins ExtDiff. Without loss of generality, suppose that $\mathcal{E}_3^*$ outputs $(\sigma', \mathbf{y}')$ such that $\sigma' \neq \sigma$, $\mathbf{y}_1' \neq \mathbf{y}_1$, but other elements are the same. As $(\sigma', \mathbf{y}')$ are witnesses that satisfy Equation 13, all terms will cancel out except

$$g^{\zeta' - \zeta^*} \cdot g_1^{\mathbf{z}_1' - \mathbf{z}_1^*} = g^{\sigma'(c' - c^*)} \cdot g_1^{\mathbf{y}_1' \cdot (c' - c^*)}. \tag{14}$$

Notice that since $g_1 = g^{d_1}$,

$$g^{\zeta' - \zeta^*} \cdot g^{d_1 \cdot (\mathbf{z}_1' - \mathbf{z}_1^*)} = g^{\sigma' \cdot (c' - c^*)} \cdot g^{d_1 \cdot (\mathbf{y}_1' \cdot (c' - c^*))}.$$

Thus, from

$$(\zeta' - \zeta^*) + d_1(\mathbf{z}_1' - \mathbf{z}_1^*) = \sigma'(c' - c^*) + d_1(\mathbf{y}_1' \cdot (c' - c^*))$$

$\mathcal{A}$ can get

$$d_1 \leftarrow -\frac{\sigma' \cdot (c' - c^*) - (\zeta' - \zeta^*)}{\mathbf{y}_1' \cdot (c' - c^*) - (\mathbf{z}_1' - \mathbf{z}_1^*)},$$

since $\sigma' \neq \frac{\zeta'-\zeta^*}{c'-c^*}$ and $\mathbf{y}'_1 \neq \frac{\mathbf{z}'_1-\mathbf{z}^*_1}{c'-c^*}$, and thus the denominator will not be 0. Therefore, $\mathcal{A}'$ can successfully derive the discrete logarithm relation between g and g_1 and win DL. In conclusion,

$$\mathrm{Adv}^{\mathsf{ExtDiff}}(\mathcal{E}_3^*) \leq \mathrm{Adv}^{\mathsf{DL}}(\mathcal{A}).$$

However, due to the discrete logarithm assumption, $\mathrm{Adv}^{\mathsf{DL}}(\mathcal{A}) \leq \mathsf{negl}$. Accordingly, the probability that $\mathcal{E}_3^*$ outputs $(\sigma', \mathbf{y}') \neq (\sigma, \mathbf{y})$ is also upper-bounded by negligible probability. Therefore, the extractor $\mathcal{E}_3$ always outputs the correct witnesses $(\sigma, \mathbf{y})$ for the statement $(\mathsf{c}_\mathbf{y}, \ell, \phi_1, \phi_2)$ under the discrete logarithm assumption.

(ii) We now show that the second part (step 4-5) is 3-special sound. We show that with three distinct ρ branches, the extracted $\mathbf{y}$ by $\mathcal{E}_3$ is ensured to satisfy the three uncombined equations: $\ell(\mathbf{y}) = 1$, $\phi_1(\mathbf{y}) = z_1$, and $\phi_2(\mathbf{y}) = z_2$.

Fix a node of tree where (c, z_1, z_2) is fixed. Let $\rho_1, \rho_2, \rho_3 \in \mathbb{Z}_q$ be distinct and let

$$L_{\rho_i} = \ell + \rho_i\phi_1 + \rho_i^2\phi_2$$

and

$$\theta_{\rho_i} = 1 + \rho z_1 + \rho^2 z_2.$$

Since the verification accepts, $L_{\rho_i}(\mathbf{y}) = \theta_{\rho_i}$ for $i \in \{1, 2, 3\}$. Define a polynomial

$$P(X) = (\ell(\mathbf{y}) - 1) + (\phi_1(\mathbf{y}) - z_1)X + (\phi_2(\mathbf{y}) - z_2)X^2.$$

It is easy to see that P vanishes at ρ_1, ρ_2, and ρ_3.

Equivalently, its coefficients satisfy the linear system

$$\begin{pmatrix} 1 & \rho_1 & \rho_1^2 \\ 1 & \rho_2 & \rho_2^2 \\ 1 & \rho_3 & \rho_3^2 \end{pmatrix} \begin{pmatrix} \ell(\mathbf{y}) - 1 \\ \phi_1(\mathbf{y}) - z_1 \\ \phi_2(\mathbf{y}) - z_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

The matrix is a Vandermonde matrix with determinant

$$(\rho_2 - \rho_1)(\rho_3 - \rho_1)(\rho_3 - \rho_2).$$

Because the ρ_j are distinct elements of $\mathbb{Z}_q$, this determinant is nonzero and hence invertible. Therefore, all three coefficients must be zero, which gives

$$\ell(\mathbf{y}) = 1, \qquad \phi_1(\mathbf{y}) = z_1, \qquad \phi_2(\mathbf{y}) = z_2.$$

Hence, three ρ-forks, with two c'-forks under each ρ-fork, allow the extractor to recover the three individual linear equations from the batched equation L.

(iii). We now show that the first part is $(2n + 1)$-special sound, by showing that there exists an efficient extractor $\mathcal{E}_1$ that extracts (f, h) from $(2n + 1)$ accepting transcripts $(c^{(1)}, (z_1^{(1)}, z_2^{(1)}))$, $\cdots, (c^{(2n+1)}, (z_1^{(2n+1)}, z_2^{(2n+1)}))$ where all challenges are distinct.

By the definition of Lagrange interpolation, $\phi_1(\mathbf{y}) = f(c)$ and $\phi_2(\mathbf{y}) = h(c)$ for every first-round challenge c. Also recall that $\deg(f) \leq n$ and $\deg(h) \leq 2n$. By (ii), each transcript satisfies

$$f(c^{(i)}) = z_1^{(i)} \text{ and } h(c^{(i)}) = z_2^{(i)}$$

for $i \in [2n + 1]$. Since the challenges are pairwise distinct and theier pairwise differences are invertible, $\mathcal{E}_1$ can efficiently recover f and h by Lagrange interpolation. In more detail, the $2n + 1$ evaluations uniquely determine h while $n + 1$ of them uniquely determine f.

Also, since the transcripts are accepting, $z_2^{(i)} = z_1^{(i)}\big(1 - z_1^{(i)}\big)$. Let's define

$$Q(X) \;=\; h(X) - f(X)\big(1 - f(X)\big), \qquad \deg(Q) \leq 2n.$$

For every $i \in [2n + 1]$,

$$\begin{aligned} Q(c^{(i)}) &= h(c^{(i)}) - f(c^{(i)})\big(1 - f(c^{(i)})\big) \\ &= z_2^{(i)} - z_1^{(i)}\big(1 - z_1^{(i)}\big) \\ &= 0 \end{aligned}$$

Thus, by the same argument as in (ii), Q is zero polynomial, and

$$h(X) = f(X)\big(1 - f(X)\big)$$

holds.

Finally, for every $i \in [n]$, the encoding of extracted polynomials gives

$$f(i) = \mathbf{x}_i \text{ and } h(i) = 0.$$

Thus,

$$\mathbf{x}_i\big(1 - \mathbf{x}_i\big) = 0 \qquad \text{for all } i \in [n],$$

and from (ii), we can conclude that $\sum_{i=1}^{n} \mathbf{x}_i = \ell(\mathbf{y}) = 1$.

(Special HVZK). Given the input and challenge $(\mathsf{c}_\mathbf{y}, \ell, \phi_1, \phi_2, L, c, \rho, c')$, we show that a simulator $\mathcal{S}$ can output an accepting transcript that has the same distribution as that of a real transcript. The simulator $\mathcal{S}$ works as shown in Figure 14.

$\mathcal{S}(\mathsf{c}_\mathbf{y}, \ell, \phi_1, \phi_2, c, \rho, c')$:

1: Sample $z_1 \leftarrow_\$ \mathbb{Z}_q$ uniformly at random and set $z_2 \leftarrow z_1 \cdot (1 - z_1)$.
2: $L(\cdot) \leftarrow \ell(\cdot) \cdot \rho^0 + \phi_1(\cdot) \cdot \rho^1 + \phi_2(\cdot) \cdot \rho^2$
3: Sample uniformly at random $\mathbf{z} \leftarrow_\$ \mathbb{Z}_q^{2n+2}, \zeta \leftarrow_\$ \mathbb{Z}_q$.
4: $A \leftarrow g^\zeta \prod_{i=1}^{2n+2} g_i^{\mathbf{z}_i} / \mathsf{c}_\mathbf{y}^{c'}$, $t \leftarrow L(\mathbf{z}) - c' \cdot (1 + z_1\rho + z_2\rho^2)$
5: **return** $(z_1, z_2, t, A, \mathbf{z}, \zeta)$

Figure 14: The simulator $\mathcal{S}$ of the protocol Π_1.

It is obvious that the resulting transcript is an accepting transcript. The right-hand side of the verification equation will be

$$\begin{aligned}\mathsf{c}_\mathbf{y}^{c'} \cdot A \cdot h^{c' \cdot (1 + z_1\rho + z_2\rho^2) + t} &= \mathsf{c}_\mathbf{y}^{c'} \cdot g^\zeta \prod_{i=1}^{2n+2} g_i^{\mathbf{z}_i} / \mathsf{c}_\mathbf{y}^{c'} \cdot h^{L(\mathbf{z})} \\ &= g^\zeta \prod_{i=1}^{2n+2} g_i^{\mathbf{z}_i} \cdot h^{L(\mathbf{z})},\end{aligned}$$

which is exactly the left-hand side of the verification equation. We now argue that the output $(z_1, z_2, t, A, \mathbf{z}, \zeta)$ of $\mathcal{S}$ has the right distribution by analyzing each element.

First, z_1 in a real transcript is computed as $z_1 \leftarrow f(c)$ where f is a random polynomial such that evaluations are fixed at $\{1, \cdots, n\}$. Thus, for other evaluation points $c \in \mathbb{Z}_q \backslash [n]$, $f(c)$ is uniformly distributed over $\mathbb{Z}_q$. Therefore, simulated $z_1 \leftarrow_\$ \mathbb{Z}_q$ has the same distribution as $z_1 \leftarrow f(c)$ in a real transcript. Since $z_2 \leftarrow h(c) = f(c) \cdot (1 - f(c))$ in a real transcript, its distribution is the same as the simulated $z_2 \leftarrow z_1 \cdot (1 - z_1)$. In a real transcript, since all elements in $\mathbf{r}$ and γ are mutually independent and uniform over $\mathbb{Z}_q$ and c is uniform over $\mathbb{Z}_q \setminus [n]$, $\mathbf{z}$ and ζ are uniformly distributed over $\mathbb{Z}_q^{2n+2}$ and $\mathbb{Z}_q$. This is the exact distribution of $(\mathbf{z}, \zeta)$ of $\mathcal{S}$'s output. Also, observe that A in a real transcript is uniformly distributed over $\mathbb{G}$ as the base group elements are raised to $\mathbf{r}$ and γ. The simulated A has the same distribution, because $g^\zeta \prod_{i=1}^{2n+2} g_i^{\mathbf{z}_i}$ is uniformly distributed over $\mathbb{G}$, and thus dividing it by $\mathsf{c}_\mathbf{y}^{c'}$ is also uniformly distributed over $\mathbb{G}$. $t \leftarrow L(\mathbf{r})$ in a real transcript is uniformly distributed over $\mathbb{Z}_q$ due to $\mathbf{r}$. In a simulated world, $L(\mathbf{z})$ is uniformly distributed over $\mathbb{Z}_q$ due to $\mathbf{z}$, and thus $t \leftarrow L(\mathbf{z}) - c' \cdot (1 + z_1\rho + z_2\rho^2)$ has a uniform distribution over $\mathbb{Z}_q$. This completes the proof. □

Theorem 9. *The communication cost of Π_1 from the prover to verifier is $1\mathbb{G} + (2n + 6)\mathbb{Z}_q$ and that from the verifier to prover is $3\mathbb{Z}_q$. Since all verifier's messages are public-coin, we can make this non-interactive using Fiat-Shamir heuristics. Then, the proof size is $1\mathbb{G} + (2n + 6)\mathbb{Z}_q$.*

Proof. The proof contains one group element A. It also contains $(z_1, z_2, t, \mathbf{z}, \zeta)$ where $|\mathbf{z}| = 2n + 2$. □

C Full Protocol of Π_1^{comp}

Figure 15 shows the full compressed zero-knowledge proof protocol Π_1^{comp} that proves statement (1) with the compressed proving technique from [5]. For simplicity of analysis, we suppose that $\mathbf{y}$ was padded with zeroes until its size becomes $2^m - 1$, i.e. a power of 2 minus 1. Also, we denote $\mathbf{g} = (g_1, \cdots, g_{|\mathbf{y}|})$

Protocol $\Pi_1^{\mathsf{comp}}(\mathsf{c}_\mathbf{y}, \ell, \phi_1, \phi_2; \mathbf{y}, \sigma, f, h)$:

1: V: sends a challenge $c \leftarrow_\$ \mathbb{Z}_q \setminus [n]$ to P.
2: P: computes $z_1 \leftarrow f(c)$, $z_2 \leftarrow h(c)$ and sends z_1, z_2 to V.
3: V: verifies $z_2 = z_1(1 - z_1)$.
4: V: sends $\rho \leftarrow_\$ \mathbb{Z}_q$ to P.
5: P, V: compute $L(\cdot) \leftarrow \ell(\cdot) \cdot \rho^0 + \phi_1(\cdot) \cdot \rho^1 + \phi_2(\cdot) \cdot \rho^2$.
6: P: samples $\mathbf{r} \leftarrow_\$ \mathbb{Z}_q^{|\mathbf{y}|}, \gamma \leftarrow_\$ \mathbb{Z}_q$ and sends $t \leftarrow L(\mathbf{r}), A \leftarrow g^\gamma \prod \mathbf{g}^\mathbf{r}$
7: V: sends $c_0, c_1 \leftarrow \mathbb{Z}_q$
8: P: gets $\mathbf{z} \leftarrow c_0 \cdot \mathbf{y} + \mathbf{r}, \zeta \leftarrow c_0 \cdot \sigma + \gamma$, and prepares
   - $\hat{\mathbf{g}} := (\mathbf{g}, g)$
   - $\hat{\mathbf{z}} := (\mathbf{z}, \zeta)$
9: P, V: prepares
   - $\tilde{L}(\hat{\mathbf{z}}) := c_1 L(\mathbf{z})$
   - P: $Q := A\mathsf{c}_y^{c_0} u^{c_1(c_0 L(\mathbf{y}) + t)}$
   - V: $Q := A\mathsf{c}_y^{c_0} u^{c_1(c_0(1 + z_1\rho + z_2\rho^2) + t)}$
10: P: sends the commitments μ, ν
   - $\mu := \hat{\mathbf{g}_R}^{\hat{\mathbf{z}}_L} \cdot u^{\tilde{L}_R(\hat{\mathbf{z}}_L)}$
   - $\nu := \hat{\mathbf{g}_L}^{\hat{\mathbf{z}}_R} \cdot u^{\tilde{L}_L(\hat{\mathbf{z}}_R)}$
11: V: sends a challenge $c' \leftarrow \mathbb{Z}_q$
12: P: folds $\hat{\mathbf{g}}$ and $\hat{\mathbf{z}}$
   - $\hat{\mathbf{g}} := \hat{\mathbf{g}}_L^{c'} * \hat{\mathbf{g}}_R$
   - $\hat{\mathbf{z}} := \hat{\mathbf{z}_L} + c' \cdot \hat{\mathbf{z}_R}$
13: P, V: folds $\tilde{L}$ and Q
   - $\tilde{L} := c'\tilde{L}_L + \tilde{L}_R$
   - $Q := \mu \cdot Q^{c'} \cdot \nu^{c'^2}$
14: P: if $|\hat{\mathbf{z}}| = 2$: send $\hat{\mathbf{z}}$ to V. Go back to step 9 otherwise.
15: V: verifies $\hat{\mathbf{g}}^\mathbf{z} \cdot u^{\tilde{L}(\hat{\mathbf{z}})} = Q$

Figure 15: Compressed proof protocol Π_1^{comp} proving (1). $*$ denotes a component-wise vector multiplication.

In step 8, the prover prepares a vector of third round messages $\mathbf{z}$ and a vector of their corresponding bases $\mathbf{g}$. Notice that Q would be the right-hand side of verification in Π_1 (without compression). In Π_1^{comp}, instead of sending the third round messages $\mathbf{z}$, the prover keeps folding $\mathbf{g}, \mathbf{z}, \tilde{L}$ and Q.

We now justify the folding in step 12-13. We show that with one folding step, the verification in step 9

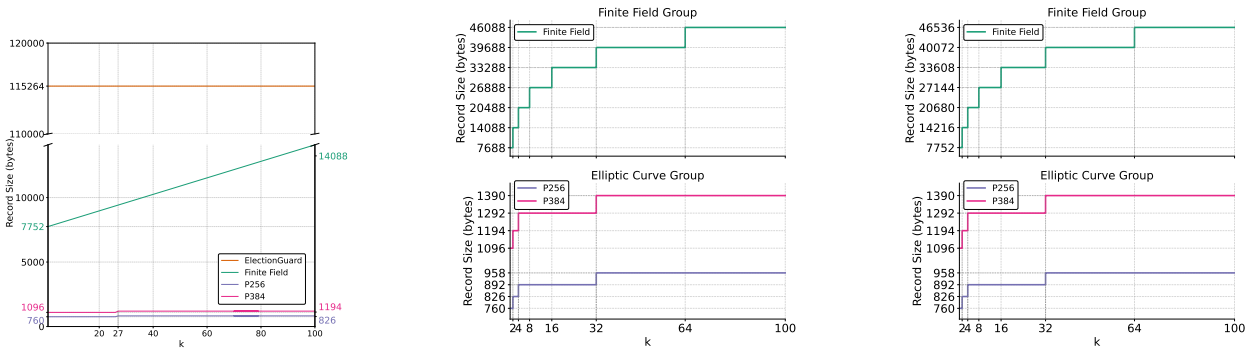
in Π_1 (Figure 2) is reduced to checking $\hat{\mathbf{g}}^{\mathbf{z}} \cdot u^{\tilde{L}(\hat{\mathbf{z}})} = Q$.

$$\begin{aligned}
Q &= \mu \cdot Q^{c'} \cdot \nu^{c'^2} \\
&= \hat{\mathbf{g}_R}^{\hat{\mathbf{z}}_L} \cdot u^{\tilde{L}_R(\hat{\mathbf{z}}_L)} \cdot (\hat{\mathbf{g}}_L^{\hat{\mathbf{z}}_L} \cdot \hat{\mathbf{g}}_R^{\hat{\mathbf{z}}_R} \cdot u^{c_1 \cdot L(\mathbf{z})})^{c'} \cdot (\hat{\mathbf{g}_L}^{\hat{\mathbf{z}}_R} \cdot u^{\tilde{L}_L(\hat{\mathbf{z}}_R)})^{c'^2} \\
&= \hat{\mathbf{g}}_R^{\hat{\mathbf{z}}_L + c'\hat{\mathbf{z}}_R} \cdot \hat{\mathbf{g}}_L^{c'(\hat{\mathbf{z}}_L + c'\hat{\mathbf{z}}_R)} \cdot u^{\tilde{L}_R(\hat{\mathbf{z}}_L) + c'(\tilde{L}_R(\mathbf{z}_R) + \tilde{L}_L(\mathbf{z}_L)) + c'^2 \tilde{L}_L(\mathbf{z}_R)} \\
&= \hat{\mathbf{g}}_R^{\hat{\mathbf{z}}_L + c'\hat{\mathbf{z}}_R} \cdot \hat{\mathbf{g}}_L^{c'(\hat{\mathbf{z}}_L + c'\hat{\mathbf{z}}_R)} \cdot u^{c'\tilde{L}_L(\hat{\mathbf{z}}_L + c'\hat{\mathbf{z}}_R) + \tilde{L}_R(\hat{\mathbf{z}}_L + c'\hat{\mathbf{z}}_R)} \\
&= \hat{\mathbf{g}}_R^{\hat{\mathbf{z}}} \cdot \hat{\mathbf{g}}_L^{c'\hat{\mathbf{z}}} \cdot u^{c'\tilde{L}_L(\hat{\mathbf{z}}) + \tilde{L}_R(\hat{\mathbf{z}})} \\
&= \hat{\mathbf{g}}^{\mathbf{z}} \cdot u^{\tilde{L}(\hat{\mathbf{z}})}.
\end{aligned}$$

With recursion, the verification of a statement of size 2^m is reduced to the statement of size 2^{m-1} at each step until $|\hat{\mathbf{z}}| = 2$, at which point the final message is sent and verified.

D Proof of Ballot Well-formedness for Various Election Rules

Now we discuss how we can extend Π_1 for more complicated election rules. We show the empirical analysis of the proof size of Haechi with following schemes with compression in Figure 16.



(a) Election record size with Π_2 (b) Election record size with Π_3 (c) Election record size with Π_4

Figure 16: The size of an election record per voter for different proof schemes when the number of candidates $n = 100$. With Π_2, a voter can cast at most one vote per candidate, for up to k different candidates. In Π_3, voters may split their k votes arbitrarily among candidates, assigning multiple votes to the same candidate if desired, as long as the total number of votes cast is exactly k. In Π_4, voters may again split their k votes among candidates, but is allowed to cast fewer than k votes in total.

D.1 Approval Voting

Consider a case where each selection must be either 0 or 1, but the sum of the selections can be up to k:

$$\Pi_2 : \forall i \in [n], \mathbf{x}_i \in \{0, 1\} \wedge \sum_{i=1}^{n} \mathbf{x}_i \leq k. \tag{15}$$

Notice that if $k = 1$, then it becomes the most common standard voting.

We propose two approaches: (i) constructing a distinct circuit for checking $\sum_{i=1}^{n} \mathbf{x}_i \in \{0, \cdots, k\}$ and (ii) introducing "dummy" candidates in order to make the sum exactly k.

D.1.1 A distinct circuit for the sum of selections

We use two circuits $\mathcal{C}$ and $\mathcal{C}'$. $\mathcal{C}$ is the same as Π_1, i.e. $\mathcal{C}(\mathbf{x}) : \mathbf{x} \mapsto \mathbf{x} * (1 - \mathbf{x})$, and proves that $\mathbf{x}_i \in \{0, 1\}$. We construct another circuit $\mathcal{C}' : \mathbb{Z}_q \to \mathbb{Z}_q$ that proves that the sum $s = \sum_{i=1}^{n} \mathbf{x}_i \in \{0, \cdots, k\}$.

$$\mathcal{C}'(x) : x \mapsto x \cdot (1 - x) \cdots (k - x)$$

As before, we represent input x as a polynomial f'. In this case, the prover chooses a random degree-1 polynomial f' such that $f'(1) = s$. Then, we define a degree-$(k+1)$ polynomial $h'(x) = f'(x) \cdot (1 - f'(x)) \cdots (k - f'(x))$.

Now we construct $\mathbf{y}$. Since we use $\mathcal{C}$ for the first clause $\mathbf{x}_i \in \{0, 1\}$, $\mathbf{y}$ contains $(\mathbf{x}, f(0), h(0), h(n+1), \cdots, h(2n))$ as before. Now, we append more elements corresponding to $\mathcal{C}'$ to $\mathbf{y}$. We first put s, the claimed sum. Then, we put $f'(0)$. Notice that since $s = f'(1)$, s and $f'(0)$ will uniquely denote f'. Similarly, we append $h'(0), \cdots, h'(k+1)$. However, since $h'(1) = 0$ is desired, we can omit $h'(1)$ from $\mathbf{y}$ and still uniquely commit to h'. Therefore, $\mathbf{y}$ is as follows.

$$\begin{aligned}\mathbf{y} =&(\mathbf{x}, f(0), h(0), h(n+1), \cdots, h(2n),\\ &s, f'(0), h'(0), h'(2), \cdots, h'(k+1)) \in \mathbb{Z}_q^{2n+k+5}.\end{aligned}$$

We now show the linear equation L. Recall that for $\mathcal{C}$, given the verifier's challenge $c \in \mathbb{Z}_q \setminus [n]$, the prover sends $z_1 \leftarrow f(c)$ and $z_2 \leftarrow h(c)$ and the verifier checks $z_2 = z_1 \cdot (1 - z_1)$. Then, the prover additionally proves that z_1 and z_2 are honestly generated by using the affine maps $\phi_1 : \mathbf{y} \mapsto f(c)$ and $\phi_2 : \mathbf{y} \mapsto h(c)$ and proving $\phi_1(\mathbf{y}) = z_1$ and $\phi_2(\mathbf{y}) = z_2$.

We do the same for $\mathcal{C}'$. The challenge space is different since only $f'(1)$ and $h'(1)$ are fixed; the challenge can now be sampled from $\mathbb{Z}_q \setminus \{1\}$ uniformly at random; we denote the challenge for $\mathcal{C}'$ as c^*. Given c^*, the prover additionally sends $z_1' \leftarrow f'(c^*)$ and $z_2' \leftarrow h'(c^*)$, and the verifier checks $z_2' = z_1' \cdot (1 - z_1') \cdots (k - z_1')$. Then, the prover shows that for $\phi_1' : \mathbf{y} \mapsto f'(c^*)$ and $\phi_2' : \mathbf{y} \mapsto h'(c^*)$, $\phi_1'(\mathbf{y}) = z_1'$ and $\phi_2'(\mathbf{y}) = z_2'$. Also, we need a check that the claimed s in $\mathbf{y}$ is the actual sum of $(\mathbf{x}_i)$. We modify ℓ to reflect this.

In more detail, given the purported z_1, z_2, z_1', z_2' and the verifier's challenge c, c^*, ρ, we construct L as follows.

$$L(\mathbf{y}) = \ell(\mathbf{y}) \cdot \rho^0 + \sum_{i=1}^{2} \phi_i(\mathbf{y}) \cdot \rho^i + \sum_{i=1}^{2} \phi_i'(\mathbf{y}) \cdot \rho^{2+i}$$

where $\ell, \phi_1, \phi_2, \phi_1', \phi_2'$ are defined as follows. Since we use the same circuit as Π_1, ϕ_1 and ϕ_2 are the same as Π_1. Regarding ϕ_1' and ϕ_2', notice that

$$f'(c) = \ell_{0,1}(c^*) \cdot f'(0) + \ell_{1,1}(c^*) \cdot f'(1)$$

and

$$h'(c^*) = \sum_{i=0}^{k+1} \ell_{i,k+1}(c^*) \cdot h'(i).$$

Since the index of $f'(0)$ and $s = f'(1)$ is $2n+4$ and $2n+3$ respectively,

$$\phi_1'(\mathbf{y}) = \frac{c^* - 1}{0 - 1} \cdot \mathbf{y}_{2n+4} + \frac{c^* - 0}{1 - 0} \cdot \mathbf{y}_{2n+3}.$$

Similarly,

$$\phi_2'(\mathbf{y}) = \prod_{j=1}^{k+1} \frac{c^* - j}{0 - j} \cdot \mathbf{y}_{2n+5} + \sum_{i=2}^{k+1} \prod_{j=0, j\neq i}^{k+1} \frac{c^* - j}{i - j} \cdot \mathbf{y}_{2n+5+i-1}.$$

Finally,

$$\begin{aligned}\ell(\mathbf{y}) &= \sum_{i=1}^{n} \mathbf{x}_i - s\\ &= \sum_{i=1}^{n} \mathbf{y}_i - \mathbf{y}_{2n+3}\end{aligned}$$

As before, $\ell, \phi_1, \phi_2, \phi_1', \phi_2'$ are public and L can be constructed by both prover and verifier after the verifier randomly selects and sends (c, c^*, ρ) to the prover.

The prover proves that $L(\mathbf{y})$ equals $0 \cdot \rho^0 + z_1 \cdot \rho^1 + z_2 \cdot \rho^2 + z_1' \cdot \rho^3 + z_2' \cdot \rho^4$. We denote this protocol as $\Pi_2^{\mathcal{C}'}$ and describes the full protocol in Figure 17.

Protocol $\Pi_2^{\mathcal{C}'}(\mathsf{c}_{\mathbf{y}}, \ell, \phi_1, \phi_2, \phi_1', \phi_2'; \mathbf{y}, \sigma, f, h, f', h')$:

1: V: sends a challenge $c \leftarrow_{\$} \mathbb{Z}_q \setminus [n]$ to P.
2: P: gets $z_1 \leftarrow f(c), z_2 \leftarrow h(c)$ and sends z_1, z_2 to V.
3: V: checks $z_2 = z_1(1 - z_1)$.
4: V: sends a challenge $c^* \leftarrow_{\$} \mathbb{Z}_q \setminus \{1\}$ to P.
5: P: gets $z_1' \leftarrow f'(c^*), z_2' \leftarrow h'(c^*)$ and sends to V.
6: V: checks $z_2' = z_1' \cdot (1 - z_1') \cdots (k - z_1')$
7: V: sends $\rho \leftarrow_{\$} \mathbb{Z}_q$ to P.
8: P, V: computes $L(\cdot) \leftarrow \ell(\cdot) \cdot \rho^0 + \sum_{i=1}^{2} \phi_i(\cdot) \cdot \rho^i + \sum_{i=1}^{2} \phi_i'(\cdot) \cdot \rho^{2+i}$
9: P: $\mathbf{r} \leftarrow_{\$} \mathbb{Z}_q^{2n+k+5}, \gamma \leftarrow_{\$} \mathbb{Z}_q$, and sends $t \leftarrow L(\mathbf{r}), A \leftarrow g^{\gamma} \Pi_{i=1}^{2n+k+5} g_i^{\mathbf{r}_i}$
10: V: sends $c' \leftarrow_{\$} \mathbb{Z}_q$
11: P: sends $\mathbf{z} \leftarrow c' \cdot \mathbf{y} + \mathbf{r}, \zeta \leftarrow c' \cdot \sigma + \gamma$.
12: V: return $g^{\zeta} \cdot \prod_{i=1}^{2n+k+5} g_i^{\mathbf{z}_i} \cdot h^{L(\mathbf{z})} \stackrel{?}{=} \mathsf{c}_{\mathbf{y}}^{c'} \cdot A \cdot h^{c' \cdot (z_1 \cdot \rho + z_2 \cdot \rho^2 + z_1' \cdot \rho^3 + z_2' \cdot \rho^4) + t}$

Figure 17: The zero-knowledge proof protocol $\Pi_2^{\mathcal{C}'}$ that proves the selection $\mathbf{x}_i \in \{0, 1\}$ and the sum of selections $\sum_{i=1}^{n} \mathbf{x}_i \leq k$ for some $k \geq 1$. Blue text denotes the difference from Π_1.

Theorem 10. *$\Pi_2^{\mathcal{C}'}$ is perfectly complete, special honest-verifier zero-knowledge, and $(2n + 1, k + 2, 5, 2)$-special-sound under the discrete logarithm assumption.*

Proof. The proof is the same as proof of Theorem 1 with a slight change due to additional polynomials f' and h' and the difference in the size of vector $\mathbf{y}$. The completeness directly follows from the protocol. The special honest-verifier zero-knowledge can be proven using the same simulator $\mathcal{S}$ as Figure 14 but samples $\mathbf{z} \leftarrow_{\$} \mathbb{Z}_q^{2n+k+5}$ uniformly at random. $\mathcal{S}$ also additionally samples $z_1' \leftarrow_{\$} \mathbb{Z}_q$ and $z_2' \leftarrow z_1' \cdot (1 - z_1') \cdots (k - z_1')$. Since f' is a random polynomial where the evaluation is fixed at an evaluation point 1, $z_1' \leftarrow f'(c^*)$ for $c^* \in \mathbb{Z}_q \setminus \{1\}$ in a real transcript looks uniformly distributed from the view of the verifier. Thus, the simulated $z_1' \leftarrow_{\$} \mathbb{Z}_q$ has the same distribution as a real $z_1' \leftarrow f'(c^*)$. Also, in a real transcript, $z_2' \leftarrow h'(c^*) = f'(c^*) \cdot (1 - f'(c^*)) \cdots (k - f'(c^*))$ by the definition. Thus, the distribution of a simulated $z_2' \leftarrow z_1' \cdot (1 - z_1') \cdots (k - z_1')$ is the same as that of a real transcript.

Notice that $\Pi_2^{\mathcal{C}}$ is $(2n+1, k+2, 5, 2)$-special sound. The additional element $(k+2)$ is due to the additional transcript (z_1', z_2') at step 4-6. Since f' is a degree-1 polynomial, an extractor needs at least 2 accepting transcripts to extract f'. However, since h' is a degree-$(k+1)$ polynomial, an extractor needs at least $(k+2)$ accepting transcripts to extract h'. Thus, with $(k+2)$ pairs of (z_1', z_2'), an extractor can extract both f' and h'. The additional 5 ρ-forks are needed to recover 5 individual equations. □

Theorem 11. *The communication cost of $\Pi_2^{\mathcal{C}'}$ from the prover to verifier is $1\mathbb{G} + (2n+k+11)\mathbb{Z}_q$ and that from the verifier to prover is $4\mathbb{Z}_q$. Since all verifier's messages are public-coin, we can make this non-interactive using Fiat-Shamir heuristics. Then, the proof size is $1\mathbb{G} + (2n + k + 11)\mathbb{Z}_q$.*

Proof. The proof contains one group element A. It also contains $(z_1, z_2, z_1', z_2', t, \mathbf{z}, \zeta)$ where $|\mathbf{z}| = 2n + k + 5$. □

D.1.2 Dummy candidates

We can prove the same thing another way. We introduce k dummy candidates (so there are $n+k$ candidates total). If the voter's selection sum l is less than k, then the device sets 1 for $k - l$ dummy candidates. This makes the selection sum over $n + k$ candidates become exactly k. In other words, we modify the statement as follows

$$\forall i \in [n + k], \mathbf{x}_i \in \{0, 1\} \wedge \sum_{i=1}^{n+k} \mathbf{x}_i = k.$$

This is the exact statement as the base case, except that the sum of selections is k instead of 1. Therefore, we use the protocol Π_1.

In more detail, let $\mathbf{x}' \in \mathbb{Z}_q^{n+k}$ be the modified input vector that contains dummy selections. Since the size of input elements become $n+k$ instead of n, the degree of f will be $n+k$, and accordingly the degree of h will be $2(n+k)$. Then,

$$\mathbf{y} = (\mathbf{x}', f(0), h(0), h(n+k+1), \cdots, h(2(n+k)) \in \mathbb{Z}_q^{2(n+k)+2}.$$

The linear equation is the same except that ℓ checks the sum equals to k and ϕ_1 and ϕ_2 are over $n+k$ evaluation points and $2(n+k)$ evaluation points respectively. Let the proof protocol Π_2. We omit the full description of Π_2^D since it is the same as Π_1 except that c is chosen from $\mathbb{Z}_q \setminus [n+k]$ uniformly at random and the size of vector $\mathbf{r}$ is $2(n+k)+2$.

Theorem 12. *Π_2^D is perfectly complete, special honest-verifier zero-knowledge, and $(2(n+k)+1, 3, 2)$-special-sound under the discrete logarithm assumption.*

Proof. Since Π_2 is exactly the same as Π_1 except the challenge size and the size of vector $\mathbf{y}$, it is obvious that Π_2^D is perfectly complete and special honest-verifier zero-knowledge. Also, it is obvious that Π_2 is $(2(n+k)+1, 3, 2)$-special sound in accordance with the change of $\deg(f) = n+k$ and $\deg(h) = 2(n+k)$ from n and $2n$ respectively. □

Theorem 13. *The communication cost of Π_2^D from the prover to verifier is $1\mathbb{G} + (2(n+k)+6)\mathbb{Z}_q$ and that from the verifier to prover is $3\mathbb{Z}_q$. Since all verifier's messages are public-coin, we can make this non-interactive using Fiat-Shamir heuristics. Then, the proof size is $1\mathbb{G} + (2(n+k)+6)\mathbb{Z}_q$.*

If $k \leq 5$, Π_2^D is more efficient in terms of proof size. If $k > 5$, $\Pi_2^{\mathcal{C}'}$ is more efficient. Since the proof size of $\Pi_2^{\mathcal{C}'}$ is $1\mathbb{G} + (2n+k+11)\mathbb{Z}_q$ and the proof size of Π_2 is $1\mathbb{G} + (2n+2k+6)\mathbb{Z}_q$,

$$\begin{aligned} 2n+k+11 &< 2n+2k+6 \\ k+11 &< 2k+6 \\ 5 &< k \end{aligned}$$

D.2 Cumulative Voting

D.2.1 Ordinary Cumulative Voting

Now, consider a cumulative voting case where each selection can be larger than 1, e.g. $\mathbf{x}_i \in \{0, 1, \cdots, k\}$ and the sum of all selections should be exactly k:

$$\Pi_3 : \forall i \in [n], x_i \in [0,k] \wedge \sum_{i=1}^{n} x_i = k. \tag{16}$$

A naive way to prove this is modifying the circuit $\mathcal{C} : \mathbb{Z}_q^n \mapsto \mathbb{Z}_q^n$ as follows: given n-length vector $\mathbf{x}$,

$$\mathcal{C}(\mathbf{x}) = \mathbf{x} * (1-\mathbf{x}) * \cdots * (k-\mathbf{x}).$$

The prover constructs the polynomial f as before; the prover randomly samples a polynomial f such that $f(i) = \mathbf{x}_i$. The degree of f is n. Then, the prover constructs the polynomial h as $h(\mathbf{x}) = f(x) \cdot (1 - f(x)) \cdots (k - f(x))$. Thus, the degree of h will blow up to $(k+1)n$.

Alternatively, we can make the degree of h logarithmic to k. Let $\kappa = \lfloor \log k \rfloor + 1$ and represent each selection $\mathbf{x}_i$ as a bit vector $\mathbf{x}_i = (b_{i,1}, \cdots, b_{i,\kappa})$. Then,

$$\mathbf{b} = (b_{1,1}, \cdots, b_{1,\kappa}, \cdots, b_{n,1}, \cdots, b_{n,\kappa}) \in \{0,1\}^{n \cdot \kappa},$$

and we run the same circuit $\mathcal{C} : \mathbf{x} \mapsto \mathbf{x} * (1-\mathbf{x})$ over this $n \cdot \kappa$-length vector $\mathbf{b}$. In other words, the prover shows that each bit $b_{i,j} \in \{0,1\}$. Notice that this ensures that $\mathbf{x}_i \in \{0, \cdots, 2^\kappa - 1\}$ instead of $\{0, \cdots, k\}$. However, checking $\mathbf{x}_i \in \{0, \cdots, 2^\kappa - 1\}$ is sufficient in this case due to the other constraint $\sum_{i=1}^n \mathbf{x}_i = k$. If there exists any $\mathbf{x}_i > k$, the sum constraint will not be satisfied.

Now consider how $\mathbf{y}$ is constructed. Since $|\mathbf{b}| = \kappa \cdot n$, the degree of f will be $\kappa \cdot n$. As we use the same circuit, $h(x) = f(x) \cdot (1 - f(x))$, $\deg(h) = 2\kappa n$. Similar to the previous protocol, we can skip committing to $h(1), \cdots h(\kappa n)$ as they should be 0. Thus,

$$\mathbf{y} = (\mathbf{b}, f(0), h(0), h(\kappa n + 1), \cdots h(2\kappa n)) \in \mathbb{Z}_q^{2\kappa n+2}.$$

Recall that we reduce checking the circuit into checking a linear equation L,

$$L(\mathbf{y}) = \ell(\mathbf{y}) \cdot \rho^0 + \sum_{i=1}^{2} \phi_i(\mathbf{y}) \cdot \rho^i \stackrel{?}{=} k + z_1 \cdot \rho + z_2 \cdot \rho^2,$$

which verifies whether the sum of selections is k, $f(c) = z_1$, and $h(c) = z_2$ where z_1, z_2 are the purported evaluation given the verifier's challenge c. Since $\mathbf{y}$ in this case contains bit representations of $\mathbf{x}_i$, we change $\ell(\mathbf{y})$ as follows

$$\ell(\mathbf{y}) = \sum_{i=0}^{n-1} \sum_{j=1}^{\kappa} \mathbf{y}_{i \cdot \kappa + j} \cdot 2^{j-1}.$$

ϕ_1 and ϕ_2 are defined similarly as those in Π_1. In sum, given c, ρ, the linear equation $L(\mathbf{y})$ is defined as follows.

$$\begin{aligned}
L(\mathbf{y}) &= \ell(\mathbf{y}) \cdot \rho^0 + \sum_{i=1}^{2} \phi_i(\mathbf{y}) \cdot \rho^i \\
&= \left(\sum_{i=0}^{n-1} \sum_{j=1}^{\kappa} \mathbf{y}_{i \cdot \kappa + j} \cdot 2^{j-1} \right) \cdot \rho^0 \\
&\quad + \left(\sum_{i=1}^{\kappa n} \prod_{j=0, j \neq i}^{\kappa n} \frac{c - j}{i - j} \cdot \mathbf{y}_i + \prod_{j=1}^{\kappa n} \frac{c - j}{0 - j} \cdot \mathbf{y}_{\kappa n + 1} \right) \cdot \rho^1 \\
&\quad + \left(\prod_{j=1}^{\kappa n} \frac{c - j}{0 - j} \cdot \mathbf{y}_{\kappa n + 2} + \sum_{i=\kappa n+1}^{2\kappa n} \prod_{j=0, j \neq i}^{2\kappa n} \frac{c - j}{i - j} \cdot \mathbf{y}_{i+2} \right) \cdot \rho^2
\end{aligned}$$

The full protocol is described in Figure 18.

Protocol $\Pi_3(\mathsf{c}_{\mathbf{y}}, \ell, \phi_1, \phi_2; \mathbf{y}, \sigma, f, h)$:

1: V: sends a challenge $c \leftarrow_{\$} \mathbb{Z}_q \setminus [\kappa n]$ to P.
2: P: gets $z_1 \leftarrow f(c), z_2 \leftarrow h(c)$ and sends z_1, z_2 to V.
3: V: checks $z_2 = z_1(1 - z_1)$.
4: V: sends $\rho \leftarrow_{\$} \mathbb{Z}_q$ to P.
5: P, V: computes $L(\cdot) \leftarrow \ell(\cdot) \cdot \rho^0 + \phi_1(\cdot) \cdot \rho^1 + \phi_2(\cdot) \cdot \rho^2$.
6: P: gets $\mathbf{r} \leftarrow_{\$} \mathbb{Z}_q^{2\kappa n+2}, \gamma \leftarrow_{\$} \mathbb{Z}_q$, and sends $t \leftarrow L(\mathbf{r}), A \leftarrow g^{\gamma} \Pi_{i=1}^{2\kappa n+2} g_i^{\mathbf{r}_i}$
7: V: sends $c' \leftarrow_{\$} \mathbb{Z}_q$
8: P: sends $\mathbf{z} \leftarrow c' \cdot \mathbf{y} + \mathbf{r}, \zeta \leftarrow c' \cdot \sigma + \gamma$.
9: V: checks $g^{\zeta} \cdot \prod_{i=1}^{2\kappa n+2} g_i^{\mathbf{z}_i} \cdot h^{L(\mathbf{z})} \stackrel{?}{=} \mathsf{c}_{\mathbf{y}}^{c'} \cdot A \cdot h^{c' \cdot (k + z_1 \cdot \rho + z_2 \cdot \rho^2) + t}$

Figure 18: The zero-knowledge proof protocol Π_3 that proves the selection $\mathbf{x}_i \in \{0, \cdots, k\}$ and the sum of selections is exactly k. Blue text denotes the difference from Π_1.

Theorem 14. *Π_3 is perfectly complete, special honest-verifier zero-knowledge, and $(2\kappa n + 1, 3, 2)$-special-sound under the discrete logarithm assumption.*

Proof. The proof is essentially the same as the proof of Theorem 1 with small changes due to the difference in the size of vector $\mathbf{y}$. Completeness follows directly from the protocol. The special honest-verifier zero-knowledge property can be proved using the same simulator $\mathcal{S}$ as Figure 14 but samples $\mathbf{z} \leftarrow_{\$} \mathbb{Z}_q^{2\kappa n+2}$. Notice that Π_3 is $(2\kappa n+1, 3, 2)$-special sound whereas Π_1 is $(2n+1, 3, 2)$-special sound. This is because since f and h in Π_3 are degree $\leq \kappa n$ and $\leq 2\kappa n$ polynomial respectively. Thus, an extractor needs at least $2\kappa n+1$ accepting transcripts in order to extract both f and h. The rest of the special soundness proof is the same as Π_1. □

Theorem 15. *The communication cost of* Π_3 *from the prover to verifier is* $1\mathbb{G} + (2\kappa n+6)\mathbb{Z}_q$ *where* $\kappa = \lfloor \log k \rfloor + 1$ *and that from the verifier to prover is* $3\mathbb{Z}_q$*. Since all verifier's messages are public-coin, we can make this non-interactive using Fiat-Shamir heuristics. Then, the proof size is* $1\mathbb{G} + (2\kappa n+6)\mathbb{Z}_q$*.*

D.2.2 Partial Cumulative Voting

This is a case where each selection can be in range $\{0, \cdots, k\}$ and the sum of selections can be up to k:

$$\Pi_4 : \forall i \in [n], x_i \in [0, k] \wedge \sum_{i=1}^{n} x_i \leq k. \tag{17}$$

We show that we can prove this statement by using the dummy candidates as § D.1.2 and the approach in § D.2.1.

First, we represent the statement as follows:

$$\forall i \in [n], \mathbf{x}_i \in \{0, \cdots, k\} \wedge \sum_{i=1}^{n} \mathbf{x}_i \in \{0, \cdots, k\}$$

As Section D.1.2, we introduce a dummy candidate, and if the voter's selection sum s is less than k, assign $k - s$ for the dummy candidate. Then, we can modify the statement as follows.

$$\forall i \in [n+1], \mathbf{x}_i \in \{0, \cdots, k\} \wedge \sum_{i=1}^{n+1} \mathbf{x}_i = k.$$

Now observe that this is the exact statement as Section D.2.1 where the only difference is that we have $n+1$ candidates instead of n. Therefore, we run Π_2 with the input $\mathbf{x}' = (\mathbf{x}_1, \cdots, \mathbf{x}_{n+1})$. We denote this proof scheme as Π_4. The size of proof of Π_4 will be $1\mathbb{G} + (2(n+1)(\lfloor \log k \rfloor + 1) + 6)\mathbb{Z}_q$.

D.2.3 Limited Cumulative Voting

This is the most complicated case and unused case where each selection can be in range $\{0, \cdots, k\}$ and the sum of selections can be up to k' such that $k' > k$. First, we represent the statement as follows:

$$\forall i \in [n] \text{ and } k < k', \mathbf{x}_i \in \{0, \cdots, k\} \wedge \sum_{i=1}^{n} \mathbf{x}_i \in \{0, \cdots, k'\}. \tag{18}$$

As shown before, we introduce dummy candidates. In more detail, we can introduce $d = \lceil \frac{k'}{k} \rceil$ dummy candidates; rather than assigning 1 for dummy candidates, the device can assign up to k for each dummy candidate. Now, we can modify the statement as follows.

$$\forall i \in [n+d], \mathbf{x}_i \in \{0, \cdots, k\} \wedge \sum_{i=1}^{n+d} \mathbf{x}_i = k'.$$

We now do not need the circuit $\mathcal{C}'$ as the sum of selections is fixed as k'. We need to run the circuit $\mathcal{C}(\mathbf{x}) = \mathbf{x} * (1 - \mathbf{x}) * \cdots * (k - \mathbf{x})$ over $\mathbf{x}' \in \mathbb{Z}_q^{n+d}$, a vector that has $\mathbf{x}$ and the selections for d dummy candidates. Then, $\deg(f) = n + d$ and $\deg(h) = (k+1)(n+d)$.

$$\mathbf{y} = (\mathbf{x}', f(0), h(0), \\ h(n+d+1), \cdots, h((k+1)(n+d)) \in \mathbb{Z}_q^{(k+1)(n+d)+2}.$$

In the protocol, say Π_6, given the verifier's challenge c, the prover only needs to respond with $z_1 \leftarrow f(c)$ and $z_2 \leftarrow g(c)$. After the verifier checks $z_2 = z_1 \cdot (1 - z_1) \cdots (k - z_1)$, the prover and verifier run a Σ protocol against the linear equation L:

$$L(\mathbf{y}) = \ell(\mathbf{y}) \cdot \rho^0 + \phi_1(\mathbf{y}) \cdot \rho^1 + \phi_2(\mathbf{y}) \cdot \rho^2$$

where $\ell(\mathbf{y}) = \sum_{i=1}^{n+d} \mathbf{y}_i$, which has to be k', and $\phi_1(\mathbf{y})$ and $\phi_2(\mathbf{y})$ are defined the same way as those in Π_1. The full construction is in Figure 19.

Protocol $\Pi_6(\mathsf{c}_\mathbf{y}, \phi_1, \phi_2; \mathbf{y}, \sigma, f, h)$:

1: V: sends a challenge $c \leftarrow_\$ \mathbb{Z}_q \setminus [n+d]$ to P.
2: P: gets $z_1 \leftarrow f(c), z_2 \leftarrow h(c)$ and sends z_1, z_2 to V.
3: V: checks $z_2 = z_1 \cdot (1 - z_1) \cdots (k - z_1)$.
4: V: sends $\rho \leftarrow_\$ \mathbb{Z}_q$ to P.
5: P, V: computes $L(\cdot) \leftarrow \ell(\cdot) \cdot \rho^0 + \phi_1(\cdot) \cdot \rho^1 + \phi_2(\cdot) \cdot \rho^2$.
6: P: gets $\mathbf{r} \leftarrow_\$ \mathbb{Z}_q^{(k+1)(n+d)+2}, \gamma \leftarrow_\$ \mathbb{Z}_q$, and sends $t \leftarrow L(\mathbf{r}), A \leftarrow g^\gamma \Pi_{i=1}^{(k+1)(n+d)+2} g_i^{\mathbf{r}_i}$
7: V: sends $c' \leftarrow_\$ \mathbb{Z}_q$
8: P: sends $\mathbf{z} \leftarrow c' \cdot \mathbf{y} + \mathbf{r}, \zeta \leftarrow c' \cdot \sigma + \gamma$.
9: V: checks $g^\zeta \cdot \prod_{i=1}^{(k+1)(n+d)+2} g_i^{\mathbf{z}_i} \cdot h^{L(\mathbf{z})} \stackrel{?}{=} \mathsf{c}_\mathbf{y}^{c'} \cdot A \cdot h^{c' \cdot (k' + z_1 \cdot \rho + z_2 \cdot \rho^2) + t}$

Figure 19: The zero-knowledge proof protocol Π_6 that proves the selection $\mathbf{x}_i \in \{0, \cdots, k\}$ and the sum of selections $\sum_{i=1}^n \mathbf{x}_i \le k'$ for some $k' \ge k$ using dummy candidates.

Theorem 16. *The communication cost of Π_6 from the prover to verifier is $1\mathbb{G} + ((k+1)(n+d)+6)\mathbb{Z}_q$ and that from the verifier to prover is $3\mathbb{Z}_q$. Since all verifier's messages are public-coin, we can make this non-interactive using Fiat-Shamir heuristics. Then, the proof size is $1\mathbb{G} + ((k+1)(n+d)+6)\mathbb{Z}_q$.*

Proof. The proof contains $\pi = (z_1, z_2, t, A, \mathbf{z}, \zeta)$ where A is a group element and $|\mathbf{z}| = (k+1)(n+d)+2$. □

E Security Analysis of the Amortized Proof Π_1^{AM}

Theorem 17. *Π_1^{Am} is perfectly complete, special honest-verifier zero-knowledge, and computationally $(2n+1, 3, m+1)$-special-sound under the discrete logarithm assumption.*

Proof. (**Completeness**). Completeness follows directly from the protocol.

(**Special Soundness**). The proof is similar to the proof of Π_1. We first prove $(m+1)$ fork on c'. For a fixed transcript $(c, \{z_{1,i}, z_{i,2}\}, \rho)$, the linear equation L and θ_i are also fixed. We take $m+1$ accepting transcripts that share (t, A) but have distinct Sigma challenges $c'_0, \cdots, c'_m$. Since the transcripts are accepting,

$$g^{\zeta^{(j)}} \cdot \prod_{i=1}^{2n+2} g_i^{\mathbf{z}_i^{(j)}} \cdot u^{L(\mathbf{z}^{(j)})} = \prod_{i=1}^{m} \mathsf{c}_{\mathbf{y}i}^{{c'_{(j)}}^i} \cdot A \cdot u^{\sum_{i=1}^m {c'_{(j)}}^i \theta_i + t}.$$

The response is $\mathbf{z}^{(j)} = \mathbf{r} + \sum_{i=1}^m (c'_{(j)})^i \cdot \mathbf{y}_i$. This is a degree-$m$ polynomial over c' with vector coefficients $(\mathbf{r}, \mathbf{y}_1, \cdots, \mathbf{y}_m)$. $\zeta^{(j)}$, similarly, is a polynomial with coefficients $(\gamma, \sigma_1, \cdots, \sigma_m)$. This creates $(m+1) \times (m+1)$ Vandermonde matrix in $c'_{(j)}$ which is invertible. Thus, an extractor can efficiently extract $(\mathbf{y}_i, \sigma_i)$ and $(\mathbf{r}, \gamma)$. Each $(\mathbf{y}_i, \sigma_i)$ satisfies $\mathsf{c}_{\mathbf{y}_i} = \mathsf{Com}(\sigma_i, \mathbf{y}_i)$; otherwise, we can build an adversary that finds another pair that constitutes the same commitment and wins the binding game of the commitment scheme.

Also, let's define $P(X) = \sum_{i=1}^m X^i (L(\mathbf{y}_i) - \theta_i)$. This polynomial P has degree $\le m$. For extracted values, we force $P(c'_{(j)}) = 0$, i.e. it vanishes at $m+1$ distinct points. Thus, $L(\mathbf{y}_i) = \theta_i$ for all i.

The 3-fork and $(2n+1)$-fork analysis is the same as that in Appendix B, thus we omit the proof here.

(**Special HVZK**). We can construct an efficient simulator. Essentially, the simulator works the same as Figure 14, but samples $z_{1,i} \leftarrow_\$ \mathbb{Z}_q$ for $i \in [m]$ and constructs $z_{2,i}$ accordingly. It then proceeds with the step 2, 3 exactly as $\mathcal{S}$ in Figure 14 and gets $(\mathbf{z}, \zeta)$ pair. It constructs A and t as:

$$A \leftarrow g^{\zeta} \prod_{i=1}^{2n+2} g_i^{\mathbf{z}_i} / \prod_{i=1}^{m} \mathsf{C}_{\mathbf{y}_i}^{c'^i}$$

and $t \leftarrow L(\mathbf{z}) - \sum_{i=1}^{m} c'^i \theta_i$. The distribution of the simulated transcript is the same as the real transcript, using the same argument as that in §B. □

F Generators for Vector Commitments

To instantiate Pedersen vector commitments, n base generators $g_i \in G$, $i \in \{1, 2, \ldots, n\}$ are needed in addition to the generator g when committing to vectors of length n. These group elements must be generated in a deterministic way such that the discrete logarithms between any pair of them as well as between any one of them and the generator g are unknown, even to the party generating them. The standard way to achieve this is to use a cryptographic hash function $\mathsf{HashToGroup} : \{0,1\}^* \to G$ to generate the i-th generator as

$$g_i = \mathsf{HashToGroup}(\texttt{context}, i),$$

where `context` is a context string unique to the use case that also ensures domain separation.

The following two sections describe how to instantiate HashToGroup for finite field groups and elliptic curve groups. For elliptic curves, HashToGroup has been extensively studied, and it is usually denoted by HashToCurve. To implement this hash function, we follow RFC9380 [37].[2] For hashing to a finite field group, HashToGroup is implemented by hashing to the finite field with HashToField, followed by a cofactor exponentiation to move the value into the group G. As HashToField is a prerequisite for HashToCurve, we follow the procedure specified in Section 5 of RFC9380.

F.1 Finite Field Groups

This section specializes the function HashToField from RFC9380 to the specific use case of a finite field group of a fixed size. RFC9380 already provides the functionality for a single HashToField call to output a limited number of field elements. However, the number of needed generators can easily exceed the allowed number of elements. For simplicity and because this functionality is not performance critical, we only output a single element in $\mathbb{F}_p$ for each counter value and map it to the group G of order q by an exponentiation with the cofactor $(p-1)/q$.

The function HashToField makes use of the function `expand_message` shown in Figure 20. It specializes the algorithm in Section 5.3.1 of RFC9380. The input message to be hashed is denoted by `msg`. The string `DST` is a domain separation tag of at most 255 bytes. The desired length of the output byte string is `len_in_bytes`. We instantiate the hash function H with `SHA-256`, which has an output size of 256 bits. Therefore the parameter `b_in_bytes` in RFC9380 is equal to 32 and `s_in_bytes` is 64. The function `I2OSP` encodes nonnegative integers into octet strings in big endian order, it is specified in RFC8017.[3] Its first input is the integer to be converted, its second input denotes the length of the output byte string. The function `strxor` computes the bitwise XOR of two byte strings. The function $\texttt{substr}(b, i, l)$ returns the substring of the byte string b starting at byte i of length l.

The function `expand_message` is used in HashToField as shown in Figure 21. It is a specialization of the algorithm in Section 5.2 of RFC9380. The extension degree is $m = 1$ here. The function HashToField takes as input a message `msg` to be hashed and the number of desired output elements, which we set to 1. The parameter L is the number of hash output bytes needed for each $\mathbb{F}_p$ element. We use $L = \lceil (\lceil \log_2(p) \rceil + k)/8 \rceil$ as defined in RFC9380, where $k = 128$ is an appropriate choice when using $H = \texttt{SHA-256}$. When hashing to a finite field defined by a 4096-bit prime p and using $k = 128$, we have $L = 528$. The string `DST` is a domain separation tag as above. The function for accessing substrings `substr` is as above and `OS2IP` converts a big endian octet string into an unsigned integer as specified in RFC8017.

[2] `https://www.rfc-editor.org/rfc/rfc9380.pdf`
[3] `https://www.rfc-editor.org/rfc/rfc8017.txt`

expand_message(msg, DST, len_in_bytes)

1: $\ell \leftarrow \lceil \texttt{len_in_bytes}/32 \rceil$
2: **ABORT** if $\ell > 255$ or $\texttt{len_in_bytes} > 65535$ or $|\texttt{DST}| > 255$
3: $\texttt{DST}' \leftarrow \texttt{DST} \| \texttt{I2OSP}(|\texttt{DST}|, 1)$
4: $\texttt{Z_pad} \leftarrow \texttt{I2OSP}(0, 64)$ // a byte string of 64 0x00 bytes
5: $\texttt{l_i_b_str} \leftarrow \texttt{I2OSP}(\texttt{len_in_bytes}, 2)$
6: $\texttt{msg_prime} \leftarrow \texttt{Z_pad} \| \texttt{msg} \| \texttt{l_i_b_str} \| \texttt{I2OSP}(0, 1) \| \texttt{DST}'$
7: $b_0 \leftarrow H(\texttt{msg_prime})$
8: $b_1 \leftarrow H(b_0 \| \texttt{I2OSP}(1, 1) \| \texttt{DST}')$
9: For $i \in \{2, \ldots, \ell\}$:
10: $\quad b_i \leftarrow H(\texttt{strxor}(b_0, b_{i-1}) \| \texttt{I2OSP}(i, 1) \| \texttt{DST}')$
11: $\texttt{uniform_bytes} \leftarrow b_1 \| \cdots \| b_\ell$
12: **return** $\texttt{substr}(\texttt{uniform_bytes}, 0, \texttt{len_in_bytes})$

Figure 20: The function expand_message.

HashToField(msg)

1: $L \leftarrow \lceil (\lceil \log_2(p) \rceil + k)/8 \rceil$
2: $\texttt{uniform_bytes} \leftarrow \texttt{expand_message}(\texttt{msg}, \texttt{DST}, L)$
3: $t \leftarrow \texttt{substr}(\texttt{uniform_bytes}, 0, L)$
4: **return** $\texttt{OS2IP}(t) \bmod p$

Figure 21: The function HashToField.

Altogether, the n generators $g_1, g_2, \ldots, g_n \in G$ are generated by obtaining

$$u_i = \mathsf{HashToField}(H_P, i) \text{ for } i = 0, 1, \ldots, n-1,$$

where H_P is the parameter hash and the input message to HashToField is H_P concatenated with the byte representation of the counter i, and then computing

$$g_i = (u_{i-1})^{\frac{p-1}{q}} \bmod p.$$

F.2 Elliptic Curve Groups

A function HashToCurve that hashes to an elliptic curve $E/\mathbb{F}_p$ can be implemented by first using HashToField to obtain an element in $\mathbb{F}_p$. This field element is then mapped to an elliptic curve point by a deterministic mapping usually denoted by map_to_curve. If the curve is not a prime order curve an additional function clear_cofactor is applied to map the point to the correct prime-order subgroup.

These functions are specified in RFC9380 in Sections 6 and 7. Our implementation uses NIST standardized curves P-256 and P-384, which are prime order curves and do not require the cofactor clearing. The deterministic mapping is given by the simplified Shallue-van de Woestijne-Ulas method. RFC9380 provides the recommended parameters and methods in Section 8.2 for P-256 and in Section 8.3 for P-384 together with optimized algorithms to implement HashToCurve. They use the expand_message function as described in the previous section, where $H =$ SHA-256 for P-256 and $H =$ SHA-384 for P-384. When hashing to integers modulo the group order q, SHA-384 is used for P-384 because the group order has 384 bits.

We do not reproduce the pseudo-code for HashToCurve and its components here because our implementation follows RFC9380 exactly.

G Proof Preliminaries

In this section, we provide formal definitions of security properties that will be used in full proofs.

Definition 3 (Non-Interactive Argument of Knowledge)**.** *Consider a relation $\mathcal{R}$ over public parameters, structure, instance, and witness tuples. A non-interactive argument of knowledge Π for $\mathcal{R}$ consists of PPT algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ and deterministic $\mathcal{K}$, denoting the generator, the prover, the verifier and the encoder respectively with the following interface.*

- $\mathcal{G}(1^\lambda) \to \mathsf{pp}$*: On input security parameter λ, samples public parameters* pp*.*
- $\mathcal{K}(\mathsf{pp}, \mathsf{s}) \to (\mathsf{pk}, \mathsf{vk})$*: On input structure* s*, representing common structure among instances, outputs the prover key* pk *and verifier key* vk*.*
- $\mathcal{P}(\mathsf{pk}, u, w) \to \pi$*: On input instance u and witness w, outputs a proof π proving that $(\mathsf{pp}, \mathsf{s}, u, w) \in \mathcal{R}$.*
- $\mathcal{V}(\mathsf{vk}, u, \pi) \to \{0, 1\}$*: Checks instance u given proof π.*

We define properties of arguments of knowledge:

- ***Completeness****: An argument of knowledge satisfies completeness if for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{lcl} & & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), \\ & & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}), \\ \mathcal{V}(\mathsf{vk}, x, \pi) = 1 & : & (\mathsf{s}, (x, w)) \leftarrow \mathcal{A}(\mathsf{pp}), \\ & & (\mathsf{pp}, \mathsf{s}, x, w) \in \mathcal{R}, \\ & & \pi \leftarrow \mathcal{P}(\mathsf{pk}, x, w) \end{array}\right] = 1.$$

- ***Knowledge Soundness****:* Π *is knowledge sound if there exists a polynomial time extractor $\mathcal{E}$ such that given the knowledge soundness games KS_0 and KS_1 defined in and for every PPT adversary $\mathcal{P}^*$, the following probability is negligible in λ:*

$$\begin{aligned} \boldsymbol{Adv}^{\mathsf{KS}}_{\Pi}(\mathcal{A}) &= \left|\Pr[{\mathsf{KS}_0}_{\Pi}^{\mathcal{P}^*}(\lambda) = 1] - \Pr[{\mathsf{KS}_1}_{\Pi,\mathcal{R}}^{\mathcal{P}^*,\mathcal{E}}(\lambda) = 1]\right| \\ &= \mathsf{negl}. \end{aligned}$$

Game ${\mathsf{KS}_0}_{\Pi}^{\mathcal{P}^*}(\lambda)$:	Game ${\mathsf{KS}_1}_{\Pi,\mathcal{R}}^{\mathcal{P}^*,\mathcal{E}}(\lambda)$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$	$\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$
$(x, \pi) \leftarrow \mathcal{P}^*(\mathsf{pp})$	$(x, \pi) \leftarrow \mathcal{P}^*(\mathsf{pp})$
$b \leftarrow \mathsf{Verify}(\mathsf{pp}, x, \pi)$	$b \leftarrow \mathsf{Verify}(\mathsf{pp}, x, \pi)$
return b	$w \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{pp}, x, \pi)$
	return $b \wedge (\mathsf{pp}, x, w) \in \mathcal{R}$

Figure 22: Knowledge soundness games

Zero-knowledge*:* Π *is zero-knowledge if there exists a PPT simulator $\mathcal{S}$ such that for any unbounded distinguisher $\mathcal{D}$, the following probability is negligible in λ:*

$$\begin{aligned} \boldsymbol{Adv}^{\mathsf{ZK}}_{\Pi}(\mathcal{A}) &= \left|\Pr[{\mathsf{ZK}_0}_{\Pi,\mathcal{R}}^{\mathcal{D},P}(\lambda) = 1] - \Pr[{\mathsf{ZK}_1}_{\Pi,\mathcal{R}}^{\mathcal{D},\mathcal{S}}(\lambda) = 1]\right| \\ &= \mathsf{negl}. \end{aligned}$$

Definition 4 (Commitment Scheme)**.** *A commitment scheme C for $\mathbb{F}$ is defined by a pair of efficient algorithms (*$\mathsf{Com}, \mathsf{Open}$*):*

Game $\mathsf{ZK}_0{}^{\mathcal{D},P}_{\Pi,\mathcal{R}}(\lambda)$:	Game $\mathsf{ZK}_1{}^{\mathcal{D},\mathcal{S}}_{\Pi,\mathcal{R}}(\lambda)$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$	$\mathsf{pp}, \tau \leftarrow \mathsf{Setup}(\lambda)$
$b \leftarrow \mathcal{D}^{\mathcal{O}(\mathsf{pp},\cdot,\cdot)}(1^\lambda)$	$b \leftarrow \mathcal{D}^{\mathcal{O}(\mathsf{pp},\cdot,\cdot)}(1^\lambda)$
return b	**return** b
$\underline{\mathcal{O}(\mathsf{pp}, x, w)}$:	$\underline{\mathcal{O}(\mathsf{pp}, x, w)}$:
if $(\mathsf{pp}, x, w) \notin \mathcal{R}$ **then return** $\perp$	if $(\mathsf{pp}, x, w) \notin \mathcal{R}$ **then return** $\perp$
else **return** $P(\mathsf{pp}, x, w)$	else **return** $\mathcal{S}(\mathsf{pp}, x, \tau)$

Figure 23: Zero-Knowledge games

Game $\mathsf{CH}^{\mathcal{A},b}(\lambda)$:	Game $\mathsf{CB}^{\mathcal{A}}(\lambda)$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$	$\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pp})$	$((m, r), (m', r')) \leftarrow \mathcal{A}(\mathsf{pp})$
$c \leftarrow \mathsf{Com}(m_b)$	$b \leftarrow \mathsf{Com}(m, r) = \mathsf{Com}(m', r')$
$b' \leftarrow \mathcal{A}(c)$	$\wedge\, m \neq m'$
return b'	**return** b

Figure 24: Commitment Hiding Game CH and Computational Binding Game CB

- $\mathsf{Com}(m; r) \to c$: *Given a message $m \in \{0,1\}^\lambda$, the committer samples $r \in \{0,1\}^\lambda$ at random and outputs the commitment c.*
- $\mathsf{Open}(c, m, r) \to \{0,1\}$: *Given a commitment c, outputs 1 if $\mathsf{Com}(m, r) = c$ and 0 otherwise.*

The commitment hiding security game (CH) in Figure 24 tasks an adversary with distinguishing which message a commitment commits to. The advantage of an adversary is defined as

$$Adv^{\mathsf{CH}}(\mathcal{A}) = |\Pr[\mathsf{CH}^{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{CH}^{\mathcal{A},1}(\lambda) = 1]|.$$

C satisfies the following properties:

- ***Perfect Hiding**: For every adversary $\mathcal{A}$,*

$$\boldsymbol{Adv}^{\mathsf{CH}}_C(\mathcal{A}) = |\Pr[\mathsf{CH}^{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{CH}^{\mathcal{A},1}(\lambda) = 1]| = 0,$$

- ***Computational Binding**: For every PPT adversary $\mathcal{A}$,*

$$\boldsymbol{Adv}^{\mathsf{CB}}_C(\mathcal{A}) = \Pr[\mathsf{CB}^{\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}.$$

where CH and CB are defined in Figure 24.

H Full Proof of Ballot Privacy (Theorem 2)

We now provide the proof of Theorem 2.

Proof. Recall that ballot record output by Vote in Haechi is $b_i = \left(\mathsf{c}_{\mathbf{y}_i}, \pi_i, \mathsf{conf}_i, H_{I_i}\right)$, where $\mathbf{y}_i = (\mathbf{x}_i, \mathsf{aux}_i)$ and $\mathsf{c}_{\mathbf{y}_i} = \mathsf{Com}(\sigma_i, \mathbf{y}_i)$ for a uniformly sampled σ_i. For a challenged ballot, the bulletin board additionally contains the opening information $(\sigma_i, \mathsf{aux}_i)$ and marks the ballot as challenged. f Also, in Haechi, the device keeps the aggregated randomness, vote vector, and the auxiliary values, and Tally outputs the aggregated vote vector along with the proof of tally. In this proof, we denote X as the aggregated vote vector of the cast ballots.

Recall that, in the PRIV game, $\mathcal{O}\mathsf{Tally}$ always computes the result from the real ballot box V_0, whose corresponding bulletin board is BB_0. When $\beta = 0$, the adversary receives BB_0, X, and the honestly

generated tally proof. When $\beta = 1$, the adversary sees BB_1, the same result X, and a simulated tally proof with respect to (BB_1, X).

Our proof strategy is to show that none of the ballot values can help distinguishing BB_0 and BB_1. More specifically, let G_0 be the experiment $\mathsf{PRIV}^{\mathcal{A},0}(\lambda)$ in Figure 3. We will proceed with a sequence of game hops and show that we eventually will hop to $\mathsf{PRIV}^{\mathcal{A},1}(\lambda)$.

Game $\mathsf{PRIV}^{\mathcal{A},\beta}(\lambda)$:
1: $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$,
2: $BB_0, BB_1, V_0, V_1, \Pi_0, \Pi_1 \leftarrow \emptyset$
3: β'
$\leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Vote},\mathcal{O}\mathsf{Proof},\mathcal{O}\mathsf{Tally}}(\mathsf{pp})$
4: **return** β'

$\mathcal{O}\mathsf{Proof}(i)$:
1: if $V_0[i] \neq V_1[i]$: **abort**
2: mark $BB_\beta[i]$ as challenged.
3: $\pi_\beta \leftarrow \Pi_\beta[i]$
4: for $\delta \in \{0,1\}$:
5: $V_\delta \leftarrow V_\delta \setminus V_\delta[i]$
6: $\Pi_\delta \leftarrow \Pi_\delta \setminus \Pi_\delta[i]$
7: **return** π_β

$\mathcal{O}\mathsf{Vote}(v_0, v_1)$:
1: for $\delta \in \{0,1\}$:
2: $(b_\delta, \pi_\delta) \leftarrow \mathsf{Vote}(v_\delta)$
3: $BB_\delta \leftarrow BB_\delta \| b_\delta$
4: $V_\delta \leftarrow V_\delta \| v_\delta$, $\Pi_\delta \leftarrow \Pi_\delta \| \pi_\delta$
5: **return** BB_β

$\mathcal{O}\mathsf{Tally}()$:
1: $(\alpha_{\mathbf{x}}, \pi_{\mathsf{Tally}}, BB_0)$
$\leftarrow \mathsf{Tally}(V_0, BB_0)$
2: for $\mathsf{c}_{\mathbf{y}} \in BB_\beta$,
3: $C \leftarrow \prod_{i \in I_{\mathsf{cast}}} \mathsf{c}_{\mathbf{y}\,i}$
4: $\pi'_{\mathsf{Tally}} \leftarrow \mathcal{S}_{\mathsf{Tally}}(C_\beta, X)$
5: **return** $(X, \pi'_{\mathsf{Tally}}, BB_\beta)$

Figure 25: The game hop from PRIV to G_1. $\mathcal{O}\mathsf{Tally}$ always runs the simulator $\mathcal{S}_{\mathsf{Tally}}$ for the tally correctness proof.

We first proceed with a game hop to G_1 (Figure 25). G_1 is identical to G_0 except that the tally proof is generated using the simulator always. When $\beta = 0$, the challenger first computes the real result X from Tally and the aggregate commitment C of the cast ballots from BB_0. Differently from G_0, adversary retrieves $\pi'_{\mathsf{Tally}} \leftarrow \mathcal{S}_{\mathsf{Tally}}(C, X)$ instead of the proof generated by the prover.

We bound this game hop using zero-knowledge property of the tally proof scheme Π_{Tally}. Consider a reduction of G_0 and G_1 to the zero-knowledge games ZK_0 and ZK_1 in Figure 23. The distinguisher $\mathcal{D}_{\mathsf{Tally}}$ simulates PRIV but obtains the tally proof using the proof oracle; when $\mathcal{A}$ queries $\mathcal{O}\mathsf{Tally}$, $\mathcal{D}$ gets X exactly as PRIV but obtains π_{Tally} through $\mathcal{O}$. Then ZK_0 simulates G_0 as G_0 generates π_{Tally} through the prover and ZK_1 simulates G_1 as G_1 generates π_{Tally} through the simulator $\mathcal{S}_{\mathsf{Tally}}$. Thus,

$$
\begin{aligned}
&\mathrm{Adv}^{G_0,G_1}(\mathcal{A}) \\
&= |\Pr[G_0^{\mathcal{A}}(\lambda) = 1] - \Pr[G_1^{\mathcal{A}}(\lambda) = 1]| \\
&\leq |\Pr[\mathsf{ZK}_0{}^{\mathcal{D},P}_{\Pi_{\mathsf{Tally}},\mathcal{R}_{\mathsf{Tally}}}(\lambda) = 1] - \Pr[\mathsf{ZK}_1{}^{\mathcal{D},\mathcal{S}}_{\Pi_{\mathsf{Tally}},\mathcal{R}_{\mathsf{Tally}}}(\lambda) = 1]| \\
&= \mathsf{negl}(\lambda).
\end{aligned}
$$

We proceed with the second game hop to G_2 (Figure 26). G_2 is the same as G_1 but every ballot well-formedness proof is generated by the simulator of Π_1. Specifically, when the adversary queries $\mathcal{O}\mathsf{Vote}$, the challenger proceeds Vote exactly but after getting $\mathsf{c}_{\mathbf{y}}$, the challenger samples the required challenges and computes π from the simulator.

We bound this game also by using the zero-knowledge property of the proof scheme Π. Consider a reduction of G_1 and G_2 to the zero-knowledge games ZK_0 and ZK_1 in Figure 23. $\mathcal{D}$ simulates the ballot privacy game but obtains the proof π using the proof oracle $\mathcal{O}$. Specifically, $\mathcal{D}$ follows the procedure in PRIV, and when $\mathcal{A}$ queries $\mathcal{O}\mathsf{Vote}$, $\mathcal{D}$ follows Vote exactly to generate the ballot except it obtains π through $\mathcal{O}$. Notice that ZK_0 simulates exactly G_1, since G_1 generates π using the proof scheme, and ZK_1 simulates G_2 exactly, since G_2 generates π using $\mathcal{S}$. Applying a standard hybrid arguments over the polynomially many

$G_2.\mathsf{Vote}(\mathsf{pp}, \mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_n))$:

1: $(\mathsf{pp}_\Pi, \mathbf{g}, H_P, H_B) \leftarrow \mathsf{pp}$,
   $(\ell, \phi_1, \phi_2) \leftarrow \mathsf{pp}_\Pi$
2: $f(x) \leftarrow_\$ \mathbb{Z}_q^{\leq n}[X]$ s.t. $f(i) = \sigma_i$
3: $h(x) \leftarrow f(x) \cdot (1 - f(x))$
4: $\mathsf{aux} \leftarrow (f(0), h(0), h(n+1), \cdots, h(2n))$
5: $\mathbf{y} \leftarrow (\mathbf{x}, \mathsf{aux})$
6: $\sigma \leftarrow_\$ \mathbb{Z}_q$, $\mathsf{c}_\mathbf{y} \leftarrow \mathsf{Com}(\sigma, \mathbf{y})$
7: $c \leftarrow_\$ \mathbb{Z}_q, \rho \leftarrow_\$ \mathbb{Z}_q, c' \leftarrow_\$ \mathbb{Z}_q$
8: $\pi \leftarrow \mathcal{S}(\mathsf{c}_\mathbf{y}, \ell, \phi_1, \phi_2, c, \rho, c')$
9: $(H_I, \mathsf{conf}) \leftarrow \mathsf{Conf}(H_B, \mathsf{c}_\mathbf{y})$
10: **return** $(\mathsf{c}_\mathbf{y}, \pi, \mathsf{conf}, H_I)$

Figure 26: The game hop of PRIV to G_2. In G_2, the proof of ballot well-formedness is simulated by $\mathcal{S}$.

ballots give

$$\begin{aligned}
&\mathrm{Adv}^{G_1, G_2}(\mathcal{A}) \\
&= |\Pr[G_1^{\mathcal{A}}(\lambda) = 1] - \Pr[G_2^{\mathcal{A}}(\lambda) = 1]| \\
&\leq Q_{\mathcal{O}\mathsf{Vote}} \cdot |\Pr[\mathsf{ZK}_0{}_{\Pi_1, \mathcal{R}_1}^{\mathcal{D}, P}(\lambda) = 1] - \Pr[\mathsf{ZK}_1{}_{\Pi_1, \mathcal{R}_1}^{\mathcal{D}, \mathcal{S}}(\lambda) = 1]| \\
&= \mathsf{negl}(\lambda).
\end{aligned}$$

Here, $Q_{\mathcal{O}\mathsf{Vote}}$ is the number of queries to $\mathcal{O}\mathsf{Vote}$, which is polynomially bound.

Game $\mathsf{PRIV}^{\mathcal{A}, \beta}(\lambda)$:

1: $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$,
2: $BB_0, BB_1, V_0, V_1, \Pi_0, \Pi_1 \leftarrow \emptyset$
3: $\beta' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Vote}, \mathcal{O}\mathsf{Proof}, \mathcal{O}\mathsf{Tally}}(\mathsf{pp})$
4: **return** β'

$\mathcal{O}\mathsf{Proof}(i)$:

1: if $V_0[i] \neq V_1[i]$: **abort**
2: mark $BB_\beta[i]$ as challenged.
3: $\pi_\beta \leftarrow \Pi_\beta[i]$
4: for $\delta \in \{0, 1\}$:
5: $V_\delta \leftarrow V_\delta \setminus V_\delta[i]$
6: $\Pi_\delta \leftarrow \Pi_\delta \setminus \Pi_\delta[i]$
7: **return** $\Pi_1[i]$

$\mathcal{O}\mathsf{Vote}(v_0, v_1)$:

1: for $\delta \in \{0, 1\}$:
2: $(b_\delta, \pi_\delta) \leftarrow \mathsf{Vote}(v_\delta)$
3: $BB_\delta \leftarrow BB_\delta \| b_\delta$
4: $V_\delta \leftarrow V_\delta \| v_\delta$, $\Pi_\delta \leftarrow \Pi_\delta \| \pi_\delta$
5: **return** BB_1

$\mathcal{O}\mathsf{Tally}()$:

1: $(\alpha_\mathbf{x}, \pi_{\mathsf{Tally}}, BB_0) \leftarrow \mathsf{Tally}(V_0, BB_0)$
2: For $\mathsf{c}_\mathbf{y} \in BB_1$, $C \leftarrow \prod_{i \in I_{\mathsf{cast}}} \mathsf{c}_{\mathbf{y}\,i}$
3: $\pi'_{\mathsf{Tally}} \leftarrow \mathcal{S}_{\mathsf{Tally}}(C, X)$
4: **return** $(X, \pi'_{\mathsf{Tally}}, BB_\beta)$

Figure 27: The game hop from G_2 to G_3. The blue text denotes the change. The adversary is given the view of the right world even if it is in the left world.

We proceed with another game hop to G_3 (Figure 27). In G_3, the adversary sees the transcripts of the right world always. In particular, for each ballot, the returned commitment, the simulated ballot well-formedness proof, identifier hash, and the confirmation code from $\mathcal{O}\mathsf{Vote}$ are replaced with those in BB_1. Also, a query to $\mathcal{O}\mathsf{Proof}(i)$ returns the right-world challenge information $\Pi_1[i] = (\sigma_i^1, \mathsf{aux}_i^1)$. When $\mathcal{O}\mathsf{Tally}$ is called, the challenger still computes X from V_0, but gets the aggregated commitment C from BB_1 and returns $\pi'_{\mathsf{Tally}} \leftarrow \mathcal{S}_{\mathsf{Tally}}(C, X)$.

We prove that the views of $\mathcal{A}$ in G_2 and G_3 are identically distributed. This game hop contains an internal sequence of hybrids $H_0, \cdots, H_Q$, where Q is the upper bound of the number of queries to $\mathcal{O}\mathsf{Vote}$.

A hybrid H_j proceeds as follows. For i-th $\mathcal{O}\mathsf{Vote}$ query, the $i \leq j$ entries of the bulletin board contain the ballots from the right world and $i > j$ entries contain the ballots from the left world. Specifically, as before, $\mathcal{O}\mathsf{Vote}$ generates $(b_\delta, \pi_\delta) \leftarrow \mathsf{Vote}(\mathbf{x}_\delta)$ for $\delta \in \{0, 1\}$. However, if $\delta = 0$,

- If $i \leq j$, $BB_0 \leftarrow BB_0 \| b_1$,
- If $i > j$, $BB_0 \leftarrow BB_0 \| b_0$.

And updates V_δ, Π_δ the same as the original game.

Also, when $\mathcal{O}\mathsf{Proof}$ is queried with i,

- If $i \leq j$, return $\Pi_1[i]$,
- If $i > j$, return $\Pi_0[i]$.

Finally, when $\mathcal{O}\mathsf{Tally}$ is queried, gets C from aggregating the commitments in BB_0 where $i \leq j$ entries contain right-world ballots and $> j$ entries contain left-world ballots.

We now show that for $j \in \{1, \cdots, Q\}$, $H_{j-1} \equiv H_j$. Notice that $i \leq j-1$th queries and $i \geq j+1$th queries to $\mathcal{O}\mathsf{Vote}$ and $\mathcal{O}\mathsf{Proof}$, the two hybrids proceed with the exact same procedure. Thus, the transcripts to the adversary are identically distributed. However, for the jth query, H_{j-1} and H_j proceed with differently.

Consider that $\mathcal{A}$ makes j-th query to $\mathcal{O}\mathsf{Vote}$ with $(\mathbf{x}_0, \mathbf{x}_1)$. Consider two cases:

Case 1: $\mathbf{x}_0 \neq \mathbf{x}_1$. In H_{j-1}, BB_0 will add b_0 generated from $\mathbf{x}_0$, whereas in H_j, BB_0 will add b_1 generated from $\mathbf{x}_1$. We analyze the distribution of commitment, ballot well-formedness proof, identifier hash, and the confirmation code of the ballot that is added to BB_0 in two hybrids.

In H_{j-1}, the commitment is $\mathsf{Com}(\sigma_0, (\mathbf{x}_0, \mathsf{aux}_0))$, and in H_j, the commitment is $\mathsf{Com}(\sigma_1, (\mathbf{x}_1, \mathsf{aux}_1))$, where $\sigma_0, \mathsf{aux}_0, \sigma_1, \mathsf{aux}_1$ are computed from Vote protocol. In particular, σ_0 and σ_1 are sampled uniformly at random and both aux_0 and aux_1 consist of the evaluation results of random polynomial f and the derived polynomial h. By the perfect hiding property of our commitment scheme,

$$\{\mathsf{Com}(\sigma_0, (\mathbf{x}_0, \mathsf{aux}_0))\} \equiv \{\mathsf{Com}(\sigma_1, (\mathbf{x}_1, \mathsf{aux}_1))\}.$$

The ballot well-formedness proof is generated by the simulator $\mathcal{S}$ on the statement $(\mathsf{c}_\mathbf{y}, \ell, \phi_1, \phi_2)$. ℓ is the same in both hybrids, and ϕ_1 and ϕ_2 are derived from the challenge c, which is sampled uniformly at random in both hybrids. Since the commitments are identically distributed, the simulated proof π_0 in H_{j-1} and π_1 in H_j are also identically distributed.

Also, H_I and conf are identically distributed in two hybrids. H_I is the hash of an identifier, which is sampled uniformly at random in both hybrids. Finally, the confirmation code is also the hash of the commitment and the identifier hash. Since the identifiers and commitments are identically distributed, applying the same hash and confirmation-code algorithms produces identically distributed outputs.

Now consider the query to $\mathcal{O}\mathsf{Proof}$ with j. Since $\mathbf{x}_0 \neq \mathbf{x}_1$, $\mathcal{O}\mathsf{Proof}$ will abort without returning any opening. Thus, the complete ballot records that $\mathcal{A}$ sees in H_{j-1} and H_j are identically distributed.

Case 2: $\mathbf{x}_0 = \mathbf{x}_1$. In this case, the j-th ballot b_0 in H_{j-1} and b_1 in H_j are generated using the same selection vector. Since both ballots are produced using the same procedure Vote, it is clear that the ballots are identically distributed.

When $\mathcal{A}$ queries $\mathcal{O}\mathsf{Proof}$ with j, in H_{j-1}, it returns $(\sigma_0, \mathsf{aux}_0)$ used to construct b_0 whereas in H_j, the oracle returns $(\sigma_1, \mathsf{aux}_1)$ used to construct b_1. As shown in the earlier case, these values are identically distributed.

Thus, in both cases, the ballots and the challenge responses in H_{j-1} and H_j have identical distributions.

When $\mathcal{O}\mathsf{Tally}$ is invoked, both hybrids calculate the same result X from V_0. (Notice that V_0 in H_{j-1} and H_j are the same; only bulletin boards are different.) However, since two hybrids have different bulletin board BB_0, the aggregated commitments are different. Say C^{j-1} be the aggregated commitment of BB_0 in H_{j-1} and C^j be that in H_j. Since the complete commitment transcripts are identically distributed, i.e. $\{C^{j-1}, X\} \equiv \{C^j, X\}$. Both hybrids invoke the same simulator $\mathcal{S}_{\mathsf{Tally}}$ on these public inputs. Therefore, for the simulated proof $\pi^{(j-1)}$ and $\pi^{(j)}$ in the two hybrids,

$$\{C^{j-1}, X, \pi^{(j-1)}\} \equiv \{C^j, X, \pi^{(j)}\}.$$

Thus, the complete views of $\mathcal{A}$ in the adjacent hybrids H_{j-1} and H_j are identically distributed.

$$\mathsf{View}_{\mathcal{A}}^{H_{j-1}} \equiv \mathsf{View}_{\mathcal{A}}^{H_j}.$$

Therefore, for every $j \in 1, \ldots, Q$,

$$\mathrm{Adv}^{H_{j-1},H_j}(\mathcal{A}) = 0.$$

Since $H_0 \equiv G_2$ and $H_Q \equiv G_3$, it follows that

$$\begin{aligned}\mathrm{Adv}^{G_2,G_3}(\mathcal{A}) &:= \left|\Pr[G_2^{\mathcal{A}}(\lambda) = 1] - \Pr[G_3^{\mathcal{A}}(\lambda) = 1]\right| \\ &\leq \sum_{j=1}^{Q} \mathrm{Adv}^{H_{j-1},H_j}(\mathcal{A}) \\ &\leq Q \cdot \mathrm{Adv}_{\mathsf{Com}}^{\mathsf{CH}}(\mathcal{A}) = 0.\end{aligned}$$

The last equality comes from the perfect hiding property of Com, Pedersen vector commitment scheme.

Now we are in a world where the adversary always sees the right-world transcript regardless of β. However, the ballot well-formedness proof is still simulated. Thus, we proceed with our final game hop to G_4. G_4 is identical to G_3 except that for every query $\mathcal{O}\mathsf{Vote}(\mathbf{x}_0, \mathbf{x}_1)$ with $\mathbf{x}_0 \neq \mathbf{x}_1$, the simulated ballot well-formedness proof contained in the right-world ballot is replaced by an honestly generated proof using the corresponding right-world witness. Notice that for queries satisfying $\mathbf{x}_0 = \mathbf{x}_1$, the ballot well-formedness proofs have remained honestly generated throughout the preceding games. Thus, in G_4, the adversary always sees the right-world transcript with the proof generated by the prover, instead of $\mathcal{S}$.

We show that G_3 and G_4 are computationally indistinguishable by the zero-knowledge property Π_1. The proof is similar to the game hop from G_1 to G_2 but we are hopping to an opposite direction; consider a reduction of G_3 and G_4 to ZK_1 and ZK_0. $\mathcal{D}$ simulates G_3 but obtains the proof π using the proof oracle $\mathcal{O}$. $\mathcal{D}$ follows G_3, but when $\mathcal{A}$ queries $\mathcal{O}\mathsf{Vote}$, $\mathcal{D}$ follows the procedure but obtains π through $\mathcal{O}$. Since G_3 generates π using the simulator, ZK_1 simulates G_3 exactly, and since G_4 generates π using the proof scheme, ZK_0 simulates G_4 exactly. Since there are $Q \leq Q_{\mathcal{O}\mathsf{Vote}}$ queries,

$$\begin{aligned}&\mathrm{Adv}^{G_3,G_4}(\mathcal{A}) \\ &= |\Pr[G_3^{\mathcal{A}}(\lambda) = 1] - \Pr[G_4^{\mathcal{A}}(\lambda) = 1]| \\ &\leq Q_{\mathcal{O}\mathsf{Vote}} \cdot |\Pr[\mathsf{ZK}_1{}_{\Pi_1,\mathcal{R}_1}^{\mathcal{D},P}(\lambda) = 1] - \Pr[\mathsf{ZK}_0{}_{\Pi_1,\mathcal{R}_1}^{\mathcal{D},\mathcal{S}}(\lambda) = 1]| \\ &= \mathsf{negl}(\lambda).\end{aligned}$$

In G_4, the adversary receives the honestly generated right-world bulletin board from $\mathcal{O}\mathsf{Vote}$, $\mathcal{O}\mathsf{Proof}$ returns the corresponding right-world openings, and in $\mathcal{O}\mathsf{Tally}$, the tally result X is computed from the left-world vote sequence and the tally proof is generated by the simulator. These are exactly the responses specified by $\mathsf{PRIV}^{\mathcal{A},1}(\lambda)$. Therefore,

$$G_4 \equiv \mathsf{PRIV}^{\mathcal{A},1}(\lambda),$$

and

$$\begin{aligned}&\mathcal{A}_{\mathsf{Haechi}}^{\mathsf{PRIV}}(\mathcal{A}) \\ &= \left|\Pr[G_0^{\mathcal{A}}(\lambda) = 1] - \Pr[G_4^{\mathcal{A}}(\lambda) = 1]\right| \\ &\leq \mathrm{Adv}^{G_0,G_1}(\mathcal{A}) + \mathrm{Adv}^{G_1,G_2}(\mathcal{A}) + \mathrm{Adv}^{G_2,G_3}(\mathcal{A}) \\ &\qquad + \mathrm{Adv}^{G_3,G_4}(\mathcal{A}) \\ &\leq \mathcal{A}_{\Pi_{\mathsf{Tally}}}^{\mathsf{ZK}} + 2Q_{\mathcal{O}\mathsf{Vote}} \cdot \mathcal{A}_{\Pi_1}^{\mathsf{ZK}} + Q_{\mathcal{O}\mathsf{Vote}} \cdot \mathcal{A}_{\mathsf{Com}}^{\mathsf{CH}}(\lambda) \\ &= \mathsf{negl}(\lambda),\end{aligned}$$

since the number of queries are polynomially bound.

□

I Full Proof of Cast-as-Intended Verifiability (Theorem 3)

Proof. Recall sCAI instantiation for Haechi. A vote v corresponds to a vote vector $\mathbf{x}$. The adversary outputs a ballot $b' = (\mathsf{c}'_{\mathbf{y}}, \pi', H'_I, \mathsf{conf}')$ together with $\mathbf{y}', \sigma'$. Recall that $\mathbf{y}' = (\mathbf{x}', \mathsf{aux}')$. The extraction algorithm in Haechi runs Open with $(\mathsf{c}'_{\mathbf{y}}, \mathbf{y}', \sigma')$ and verifies that $\mathsf{c}'_{\mathbf{y}}$ is the commitment to $\mathbf{y}'$ where $\mathbf{x}' \neq \mathbf{x}$.

Due to the wining condition, if $\mathcal{A}$ wins, then $\mathbf{x}' \neq \mathbf{x}$ while $\mathsf{Voter.Chal}$ still accepts. This means that $\mathsf{Com}((\mathbf{x}, \mathsf{aux}'), \sigma') = \mathsf{c}'_{\mathbf{y}}$ where $\mathsf{c}'_{\mathbf{y}}$ was computed by $\mathsf{Com}((\mathbf{x}', \mathsf{aux}'), \sigma')$. Since $\mathbf{x}' \neq \mathbf{x}$, the adversary has produced two distinct openings of the same commitment.

Thus, our strategy is to show a reduction from sCAI to the commitment binding game CB in Figure 24. In other words, we show that for an adversary $\mathcal{A}'$ against CB, it can use $\mathcal{A}$ against sCAI to win CB. The reduction proceeds as follows. Given the public parameters of the commitment scheme Com, $\mathcal{A}'$ does the following:

1: Generate the remaining public parameters pp for Haechi.
2: Get the valid vote selections $\mathbf{x}$.
3: Invoke $\mathcal{A}$ with $(\mathsf{pp}, \mathbf{x})$.
4: When $\mathcal{A}$ outputs (b', σ'), get $\mathbf{y}' = (\mathbf{x}', \mathsf{aux}')$ by running $\mathsf{open}(\mathsf{c}'_{\mathbf{y}}, \sigma')$.
5: Output $((\mathbf{x}', \mathsf{aux}'), \sigma')$ and $((\mathbf{x}, \mathsf{aux}'), \sigma')$

As pointed out above, if $\mathcal{A}$ wins, then $\mathsf{Com}((\mathbf{x}, \mathsf{aux}'), \sigma') = \mathsf{Com}((\mathbf{x}', \mathsf{aux}'), \sigma')$, and $\mathcal{A}'$ wins CB as well. However, by the binding property of our underlying commitment scheme, this is bounded by negligible probability. In other words,

$$\mathrm{Adv}^{\mathsf{sCAI}}_{\mathsf{Haechi}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{CB}}_{\mathsf{Com}}(\mathcal{A}') = \mathsf{negl}(\lambda).$$

□

Remark 1. *Corollary 1 treats p as a lower bound on the per-ballot challenge probability. A malicious device may do far better than chance at guessing which voters are less likely to challenge, so the corollary should be read as a min-entropy style guarantee than a model of voter behavior.*

There may also be situations in which the number of challenges that can be issued by a single voter is limited to a constant c. If this is the case, once the c challenges are exhausted, a malicious device could alter a ballot record without fear of the alteration being detected. Here if a voter repeatedly issues challenges with probability p, there is a probability of p^c that the voter's challenges will be exhausted and the voter will be unable to detect an alteration of the next ballot record. Because of this, somewhat ironically, increasing the challenge probability may make a voter more vulnerable – while a voter is also vulnerable if the challenge probability is too low. The optimal strategy for a voter restricted to a fixed challenge probability[4] is to choose p such that $p^c + p - 1 = 0$ which gives $p = 0.5$ when $c = 1$ and p increasing towards 1 as c increases. Here the chance of a malicious device altering a single vote undetected is $1 - p$, and as above the chance of altering t votes without detection is $(1-p)^t < e^{-tp}$.

J Full Proof of Tallied-as-Recorded Verifiability (Theorem 5)

Before proving Theorem 5, we first show that Π_1 satisfies knowledge soundness.

Corollary 3 (Corollary from Theorem 1)**.** *The proof scheme of ballot well-formedness Π_1 is knowledge sound. The proof scheme Π_1 is knowledge sound with the knowledge soundness error $\epsilon \leq (2n+3)/(|\mathbb{Z}_q| - n)$ where q is the group order and n is the number of candidates.*

Proof. It is proven in [7] that $(k_1, \cdots, k_\mu)$-special sound $(2\mu+1)$-move interactive protocol for a relation where the verifier samples each challenge uniformly at random from a challenge set N_i $(i \in [\mu])$ implies knowledge soundness with error

$$\epsilon \leq \sum_{i=1}^{\mu} \frac{k_i - 1}{N_i}.$$

[4] An even better strategy would be for a voter to decrease the challenge probability with each iteration. But there is no need to analyze that here.

Haechi uses groups with prime order $q(\lambda)$. The number of candidates n is polynomially bounded in λ. Plugging our proof scheme Π_1, Π_1 satisfies knowledge soundness with error

$$\begin{aligned}\epsilon &\leq \frac{2n+1-1}{|\mathbb{Z}_q|-n} + \frac{3-1}{|\mathbb{Z}_q|} + \frac{2-1}{|\mathbb{Z}_q|} \\ &\leq \frac{2n+3}{|\mathbb{Z}_q|-n}.\end{aligned}$$

The error ϵ is negligible in λ. This concludes the proof. □

Corollary 4 (Corollary from Theorem 8)**.** *The proof scheme of tally correctness Π_{Tally} is knowledge sound with the knowledge soundness error $\epsilon \leq 1/q$ where q is the order of the group G used in the proof scheme.*

Proof. The proof is the same as above. Since Π_{Tally} is 2-special sound and the challenge space is $\mathbb{Z}_q$, by [7], the knowledge soundness is $\epsilon \leq 1/q$. □

By [8], the above corollaries give another corollary about knowledge soundness when the proof schemes are made non-interactive via Fiat-shamir transform.

Corollary 5 (Knowledge soundness of Fiat-Shamir transform)**.** *Let $\Pi \in \{\Pi_1, \Pi_1^{\mathsf{comp}}, \Pi_1^{Am}, \Pi_{\mathsf{Tally}}\}$ be a $(k_1, \cdots, k_\mu)$-special sound public-coin interactive proof scheme with knowledge error ϵ. Let $\mathsf{FS}[\Pi]$ denote its Fiat-Shamir transform in which each challenge is derived by hashing a round tag, base hash H_B, the statement, and the running transcript. In the random oracle model, $\mathsf{FS}[\Pi]$ is adaptively knowledge sound with knowledge error at most $(Q+1)\cdot\epsilon$ against a prover making at most Q random oracle queries. The knowledge extractor runs in expected polynomial time.*

We remark that Corollary 5 applies to Π_1^{comp} despite it contains $\mu = O(\log n)$ rounds where n is the number of selectable options. The bound from [8] is $(Q+1)\cdot\epsilon$ where ϵ is the knowledge error of Π_1^{comp} independently of μ.

Corollary 6 (Zero-knowledge of Fiat-Shamir transform)**.** *Let $\Pi \in \{\Pi_1, \Pi_1^{\mathsf{comp}}, \Pi_1^{Am}, \Pi_{\mathsf{Tally}}\}$ be a public-coin interactive proof scheme with perfect special HVZK simulator $\mathcal{S}_\Pi$, and let α_Π denote the min-entropy of the first prover message of Π. Then, $\mathsf{FS}[\Pi]$ is zero-knowledge in the random oracle model. Specifically, there exists a simulator $\mathcal{S}_{\mathsf{FS}[\Pi]}$, programming the random oracle at the challenge points, such that for any adversary making at most Q random oracle queries and receiving at most Q_S simulated proofs, the distinguishing advantage is at most $Q_S \cdot (Q+Q_S) \cdot 2^{-\alpha_\Pi}$*

Proof of Theorem 5. Let's denote the original TAR as G_0. We proceed with a game hop to G_1, which is the same as G_0 but $\mathsf{VerifyProof}$ additionally extracts the witness of each proof π_i and verifies that the extracted witness satisfies the relation $\mathcal{R}_1$ (see Figure 28).

Game $G_1^{\mathcal{A}}(\lambda)$:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

$(BB, \alpha'_{\mathbf{x}}, \pi'_{\mathsf{Tally}}, \alpha_{\mathbf{x}}, \alpha_\sigma, \alpha_{\mathsf{aux}}) \leftarrow \mathcal{A}(\mathsf{pp})$

Abort if $\mathsf{VerifyProof}(\mathsf{pp}, BB) = 0$

$\forall i \in I_{\mathsf{cast}}$:

  $w_i \leftarrow \mathcal{E}^{\Pi_1}(\mathsf{pp}_\Pi, \mathsf{c}_{\mathbf{y}i}, \pi_i)$

  if $(\mathsf{pp}_\Pi, \mathsf{c}_{\mathbf{y}i}, w_i) \notin \mathcal{R}_1$: **return** 0

$C \leftarrow \prod_{i \in I_{\mathsf{cast}}} \mathsf{c}_{\mathbf{y}i}$

Abort if $C \neq \mathsf{Com}\big(\alpha_\sigma, (\alpha_{\mathbf{x}}, \alpha_{\mathsf{aux}})\big)$

return $\rho(\alpha'_{\mathbf{x}}) \neq \rho(\alpha_{\mathbf{x}}) \ \wedge \ \mathsf{VerifyTally}(\mathsf{pp}, BB, \alpha'_{\mathbf{x}}, \pi'_{\mathsf{Tally}}) = 1$

Figure 28: Game hop from TAR to G_1. G_1 is the same as TAR but additionally extracts the witnesses of the proof and verify if the witnesses satisfy the relation. I_{cast} denotes the index set of cast ballots in BB.

We bound this game hop using the knowledge soundness of Π_1. We cannot directly reduce G_0 and G_1 to KS_0 and KS_1 in Figure 22 because $\mathsf{VerifyProof}$ verifies all $|BB|$ proofs in the election record output by the adversary whereas KS_0 and KS_1 are defined over a single proof.

Thus, we give a hybrid argument. Let $|BB| \leq R_{\mathsf{max}}(\lambda)$ where R_{max} is polynomial. We define games

$$G^{(0)}, G^{(1)}, \cdots, G^{(|BB|)},$$

where $G^{(0)} = G_0$, $G^{(|BB|)} = G_1$, and $G^{(j)}$ is the game where valid witnesses are extracted for the first j accepted proofs. In other words, we proceed with the step 5-6 in Figure 28 only for the first j proofs. Notice that the difference between $G^{(j-1)}$ and $G^{(j)}$ is whether a witness is extracted for the j-th proof.

We reduce $G^{(j-1)}$ and $G^{(j)}$ to KS_0 and KS_1 in Figure 22. First consider the left reduction. Given pp_Π, the adversary $\mathcal{A}'$ against KS_0 runs $\mathcal{A}$ against $G^{(j-1)}$ exactly. When $\mathcal{A}$ outputs the bulletin board and tally result, $\mathcal{A}'$ outputs the j-th commitment and proof $\{\mathsf{c}_\mathbf{y}, \pi\}_j$. If $\mathcal{A}$ wins TAR, since $\mathsf{VerifyProof}(\mathsf{pp}, R)$ accepts all proofs, $\mathcal{A}'$ also wins KS_0.

Now consider the right reduction. Similar as above, $\mathcal{A}'$ uses $\mathcal{A}$ as its subroutine. Since $\mathcal{A}$ extracts the witness of j-th proof, if $\mathcal{A}$ wins $G^{(j)}$, the extracted witness w_j satisfy the relation. Thus, $\mathcal{A}$ also wins KS_1. Therefore,

$$|\Pr[G^{(j-1)}(\lambda) = 1] - \Pr[G^{(j)}(\lambda) = 1]| \leq \mathrm{Adv}^{\mathsf{KS}}_{\Pi_1}(\lambda).$$

Summing over the hybrids,

$$\begin{aligned}\mathrm{Adv}^{G_0,G_1}(\mathcal{A}) &= |\sum_{j=1}^{|BB|} \Pr[G^{(j-1)}(\lambda) = 1] - \Pr[G^{(j)}(\lambda) = 1]| \\ &\leq R_{\mathsf{max}} \cdot \mathrm{Adv}^{\mathsf{KS}}_{\Pi}(\mathcal{A}') = \mathsf{negl}(\lambda).\end{aligned}$$

The last equation holds due to Corollary 3.

We are now in a world where for each i-th entry in R, the challenger obtains the accepting witness $(\mathbf{y}_i, \sigma_i, f_i, h_i)$ from the extractor, i.e. $\mathsf{c}_{\mathbf{y}_i} = \mathsf{Com}(\sigma_i, \mathbf{y}_i)$.

Game $G_2^{\mathcal{A}}(\lambda)$:

1: $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2: $(BB, \alpha'_\mathbf{x}, \pi'_{\mathsf{Tally}}, \alpha_\mathbf{x}, \alpha_\sigma, \alpha_{\mathsf{aux}}) \leftarrow \mathcal{A}(\mathsf{pp})$
3: Abort if $\mathsf{VerifyProof}(\mathsf{pp}, BB) = 0$
4: $\forall i \in I_{\mathsf{cast}}$:
5: $\quad w_i = (\mathbf{y}_i = (\mathbf{x}_i, \mathsf{aux}_i), \sigma_i, f_i, h_i) \leftarrow \mathcal{E}^{\Pi_1}(\mathsf{pp}_\Pi, \mathsf{c}_{\mathbf{y}_i}, \pi_i)$
6: $\quad$ **if** $(\mathsf{pp}_\Pi, \mathsf{c}_{\mathbf{y}_i}, w_i) \notin \mathcal{R}_1$: **return** 0
7: $C \leftarrow \prod_{i \in I_{\mathsf{cast}}} \mathsf{c}_{\mathbf{y}_i}$
8: Abort if $C \neq \mathsf{Com}\big(\alpha_\sigma, (\alpha_\mathbf{x}, \alpha_{\mathsf{aux}})\big)$
9: $\alpha^* = (\alpha^*_\sigma, \alpha^*_\mathbf{y} = (\alpha^*_\mathbf{x}, \alpha^*_{\mathsf{aux}})) \leftarrow (\sum_{i=1}^{|BB|} \sigma_i, \sum_{i=1}^{|BB|} \mathbf{y}_i)$
10: **return** $\rho(\alpha'_\mathbf{x}) \neq \rho(\alpha^*_\mathbf{x}) \ \wedge \ \mathsf{VerifyTally}(\mathsf{pp}, BB, \alpha'_\mathbf{x}, \pi'_{\mathsf{Tally}}) = 1$

Figure 29: Game hop from G_1 to G_2. G_2 is the same as G_1 but gets the honest aggregated values α from the extracted values instead of opening C.

We now proceed with another game hop to G_2 in Figure 29 where it derives the honest aggregated randomness and messages from the extracted witnesses rather than opening C. We bound this game hop using the binding property of commitment scheme Com. By additive homomorphism,

$$C = \prod_{i=1}^{|BB|} \mathsf{c}_{\mathbf{y}_i} = \mathsf{Com}(\sum_{i=1}^{|BB|} \sigma_i, \sum_{i=1}^{|BB|} \mathbf{y}_i).$$

Since $C = \mathsf{Com}(\alpha_\sigma, \alpha_\mathbf{y})$ as well, if $(\sum_{i=1}^{|BB|} \sigma_i, \sum_{i=1}^{|BB|} \mathbf{y}_i)$ in G_2 and $(\alpha_\sigma, \alpha_\mathbf{y})$ in G_1 are different, we can build an adversary against CB. The reduction proceeds as follows:

1. Generate the remaining public parameters pp for Haechi.
2. Invoke $\mathcal{A}$ with pp.
3. When $\mathcal{A}$ outputs the aggregate result and the bulletin board $(\alpha', \alpha_\sigma, BB)$, compute $C \leftarrow \prod_{i=1}^{|BB|} \mathsf{c}_{\mathbf{y}_i}$ and $\alpha_{\mathbf{y}} \leftarrow \mathsf{open}(C, \alpha_\sigma)$. If $\alpha_{\mathbf{y}} = \perp$, abort.
4. Run the verification $\mathsf{VerifyProof}$ and extract $(\mathbf{y}_i, \sigma_i, f_i, h_i)$ for each i-th element in R.
5. Output $(\alpha_\sigma, \alpha_{\mathbf{y}}), (\sum_{i=1}^{|BB|} \sigma_i, \sum_{i=1}^{|BB|} \mathbf{y}_i)$.

Notice that $(\alpha_\sigma, \alpha_{\mathbf{y}})$ exactly simulates the honest aggregated values in G_1 and $(\sum_{i=1}^{|BB|} \sigma_i, \sum_{i=1}^{|BB|} \mathbf{y}_i)$ is the sum of extracted values in G_2. If $\mathcal{A}$ wins, since all extracted values are in relation,

$$\mathsf{Com}(\alpha_\sigma, \alpha_{\mathbf{y}}) = Com(\sum_{i=1}^{|BB|} \sigma_i, \sum_{i=1}^{|BB|} \mathbf{y}_i).$$

Thus, $\mathcal{A}'$ can win CB as well. Therefore,

$$\mathrm{Adv}^{G_1, G_2}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{CB}}_{\mathsf{Com}}(\mathcal{A}') = \mathsf{negl}(\lambda).$$

Game $G_3^{\mathcal{A}}(\lambda)$:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(BB, \alpha'_{\mathbf{x}}, \pi'_{\mathsf{Tally}}, \alpha_{\mathbf{x}}, \alpha_\sigma, \alpha_{\mathsf{aux}}) \leftarrow \mathcal{A}(\mathsf{pp})$
Abort if $\mathsf{VerifyProof}(\mathsf{pp}, BB) = 0$
$\forall i \in I_{\mathsf{cast}}$:
  $w_i = (\mathbf{y}_i = (\mathbf{x}_i, \mathsf{aux}_i), \sigma_i, f_i, h_i) \leftarrow \mathcal{E}^{\Pi_1}(\mathsf{pp}_\Pi, \mathsf{c}_{\mathbf{y}_i}, \pi_i)$
  if $(\mathsf{pp}_\Pi, \mathsf{c}_{\mathbf{y}_i}, w_i) \notin \mathcal{R}_1$: **return** 0
$C \leftarrow \prod_{i \in I_{\mathsf{cast}}} \mathsf{c}_{\mathbf{y}_i}$
Abort if $C \neq \mathsf{Com}\big(\alpha_\sigma, (\alpha_{\mathbf{x}}, \alpha_{\mathsf{aux}})\big)$
$\alpha^* = (\alpha^*_\sigma, \alpha^*_{\mathbf{y}} = (\alpha^*_{\mathbf{x}}, \alpha^*_{\mathsf{aux}})) \leftarrow (\sum_{i=1}^{|BB|} \sigma_i, \sum_{i=1}^{|BB|} \mathbf{y}_i)$
if $\mathsf{VerifyTally}(\mathsf{pp}, BB, \alpha'_{\mathbf{x}}, \pi'_{\mathsf{Tally}}) = 0$: **return** 0
$(\alpha'_\sigma, \alpha'_{\mathsf{aux}}) \leftarrow \mathcal{E}^{\Pi_{\mathsf{Tally}}}(\mathsf{pp}, C, \alpha'_{\mathbf{x}})$
if $(\mathsf{pp}, C, \alpha'_{\mathbf{x}}; \alpha'_\sigma, \alpha'_{\mathsf{aux}}) \notin \mathcal{R}_{\mathsf{Tally}}$:
  return 0
return $\rho(\alpha'_{\mathbf{x}}) \neq \rho(\alpha^*_{\mathbf{x}})$

Figure 30: Game hop from G_2 to G_3. G_3 is the same as G_2 but additionally extracts the witness of π'_{Tally} and verify if the witness satisfies $\mathcal{R}_{\mathsf{Tally}}$.

We now proceed with another game hop to G_3 (Figure 30). G_3 is the same as G_2, but the challenger additionally extracts the witness for $\mathsf{VerifyTally}$ and checks if the witness together with the statement $(C, \alpha'_{\mathbf{x}})$ satisfies the tally relation $\mathcal{R}_{\mathsf{Tally}}$.

We bound this game hop using knowledge soundness of Π_{Tally}. We reduce G_2 and G_3 to KS_0 and KS_1 respectively. Given pp, the adversary $\mathcal{A}'$ against KS_0 runs $\mathcal{A}$ against G_2 as subroutine. When $\mathcal{A}$ outputs, $\mathcal{A}'$ outputs $(C, \pi'_{\mathsf{Tally}})$. If $\mathcal{A}$ wins G_2, since $\mathsf{VerifyTally}$ accepts, $\mathcal{A}'$ also wins KS_0. Now,, the adversary $\mathcal{A}'$ against KS_1 runs $\mathcal{A}$ against G_3 as subroutine. Since $\mathcal{A}$ extracts the witness used to construct π'_{Tally}, if $\mathcal{A}$ wins G_3, the extracted witness pair $(\alpha'_\sigma, \alpha'_{\mathsf{aux}})$ satisfies the relation $\mathcal{R}_{\mathsf{Tally}}$. Thus, $\mathcal{A}'$ also wins KS_1.

In other words,

$$\begin{aligned}\mathrm{Adv}^{G_2, G_3}(\mathcal{A}) &= |\Pr[G_2(\lambda) = 1] - \Pr[G_3(\lambda) = 1]| \\ &\leq \mathrm{Adv}^{\mathsf{KS}}_{\Pi_{\mathsf{Tally}}}(\lambda) = \mathsf{negl}(\lambda).\end{aligned}$$

We are now in a world where the challenger learns the witness $(\alpha'_\sigma, \alpha'_{\mathsf{aux}})$. We finish the proof by showing that the advantage of $\mathcal{A}$ in G_3 is bounded by the commitment binding assumption. The intuition is as follows. If the adversary wins, since $(\mathsf{pp}, C, \alpha'_{\mathbf{x}}; \alpha'_\sigma, \alpha'_{\mathsf{aux}}) \in \mathcal{R}_{\mathsf{Tally}}$,

$$C = \mathsf{Com}(\alpha'_\sigma, (\alpha'_{\mathbf{x}}, \alpha'_{\mathsf{aux}})),$$

and

$$C = \mathsf{Com}(\alpha^*_\sigma, (\alpha^*_{\mathbf{x}}, \alpha^*_{\mathsf{aux}})),$$

where $\alpha'_{\mathbf{x}} \neq \alpha^*_{\mathbf{x}}$ by the winning condition. Thus, we can construct another adversary $\mathcal{A}'$ against CB in Figure 24.

For an adversary $\mathcal{A}'$ against CB, given the public parameters for commitment, it proceeds as following:

1. Generate the remaining public parameters pp for Haechi.
2. Invoke $\mathcal{A}$ with pp.
3. When $\mathcal{A}$ outputs, run the verification $\mathsf{VerifyProof}$ and extract $(\mathbf{y}_i, \sigma_i, f_i, h_i)$ for each i-th element in R.
4. Get $\alpha^* = (\alpha^*_\sigma, \alpha^*_{\mathbf{x}}, \alpha^*_{\mathsf{aux}})$ by summing all $(\sigma_i, \mathbf{y}_i)$ as step 8 in Figure 30.
5. Run verification $\mathsf{VerifyTally}$ and extract the witness $(\alpha'_\sigma, \alpha'_{\mathsf{aux}})$ for the statement $(C, \alpha'_{\mathbf{x}})$ and $\mathcal{R}_{\mathsf{Tally}}$.
6. Output $(\alpha^*_\sigma, (\alpha^*_{\mathbf{x}}, \alpha^*_{\mathsf{aux}})), (\alpha'_\sigma, (\alpha'_{\mathbf{x}}, \alpha'_{\mathsf{aux}}))$.

Thus, if $\mathcal{A}$ wins, $\mathcal{A}'$ can win CB. However, by the binding property of our underlying commitment scheme, this reduction is bounded by negligible probability.

$$\mathrm{Adv}^{G_3}_{\mathsf{Haechi}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{CB}}_{\mathsf{Com}}(\mathcal{A}') = \mathsf{negl}(\lambda).$$

In conclusion,

$$\begin{aligned}
&\mathrm{Adv}^{\mathsf{TAR}}_{\mathsf{Haechi}}(\mathcal{A}) \\
&\leq R_{\mathsf{max}} \cdot \mathrm{Adv}^{\mathsf{KS}}_{\Pi}(\lambda) + \mathrm{Adv}^{G_1}_{\mathsf{Haechi}}(\mathcal{A}) \\
&\leq R_{\mathsf{max}} \cdot \mathrm{Adv}^{\mathsf{KS}}_{\Pi}(\lambda) + \mathrm{Adv}^{\mathsf{CB}}_{\mathsf{Com}}(\lambda) + \mathrm{Adv}^{G_2}_{\mathsf{Haechi}}(\mathcal{A}) \\
&\leq R_{\mathsf{max}} \cdot \mathrm{Adv}^{\mathsf{KS}}_{\Pi}(\lambda) + \mathrm{Adv}^{\mathsf{CB}}_{\mathsf{Com}}(\lambda) \\
&\quad + \mathrm{Adv}^{\mathsf{KS}}_{\Pi_{\mathsf{Tally}}}(\lambda) + \mathrm{Adv}^{G_3}_{\mathsf{Haechi}}(\mathcal{A}) \\
&\leq R_{\mathsf{max}} \cdot \mathrm{Adv}^{\mathsf{KS}}_{\Pi}(\lambda) + 2 \cdot \mathrm{Adv}^{\mathsf{CB}}_{\mathsf{Com}}(\lambda) + \mathrm{Adv}^{\mathsf{KS}}_{\Pi_{\mathsf{Tally}}}(\lambda) \\
&= \mathsf{negl}(\lambda).
\end{aligned}$$

□

We give a corollary that shows tallied-as-recorded verifiability with Haechi instantiated with non-interactive proof schemes that were transformed via Fiat-Shamir heuristics.

Corollary 7. *Consider Haechi instantiated with $\mathsf{FS}[\Pi_1]$ and $\mathsf{FS}[\Pi_{\mathsf{Tally}}]$. Also consider the tallied-as-recorded verifiability game TAR modified so that $\mathcal{A}$ additionally has oracle access to Hash modeled as random oracle and $\mathcal{A}$ has at most Q_{RO} queries. Then, for any PPT adversary $\mathcal{A}$ and polynomial bound $R_{\max}$ on the length of BB, $Adv^{\mathsf{TAR}}_{\mathsf{Haechi}}(\mathcal{A}) \leq (Q+1) \cdot R_{\max} \cdot \epsilon_{\Pi_1} + 2 \cdot Adv^{\mathsf{CB}}_{\mathsf{Com}}(\lambda) + (Q+1) \cdot \epsilon_{\Pi_{\mathsf{Tally}}}$, where ϵ_{Π_1} and $\epsilon_{\Pi_{\mathsf{Tally}}}$ denote the knowledge error of interactive proof schemes.*

Proof. The proof is identical to the proof of Theorem 5 with two changes. First, the reduction simulates the random oracle Hash for $\mathcal{A}$ by lazy sampling. Also, the knowledge extractor of Corollary 5 is used. Since $\mathcal{A}$ chooses the election record BB after its oracle queries, each of at most $R_{\max}+1$ extractor invocations is charged against $\mathcal{A}$'s query budget Q. The union bound over the extracted proofs gives $(Q+1)$ factors. □

K Receipt-freeness of Haechi

We analyze the receipt-freeness of Haechi using the definition of Moran and Naor [52]. The definition follows a real vs. ideal simulation approach: for every adversary against the real protocol, there must exist an ideal adversary that can reproduce an indistinguishable view using only the information retrieved during ideal executions.

In both ideal and real world, there are m parties $(p_1, \cdots, p_m)$ with real inputs $(x_1, \cdots, x_m)$ and fake inputs $(x'_1, \cdots x'_m)$. Each party also contains coercion-response bits $(c_1, \cdots, c_m)$ which defines the party's behavior when coerced. If $c_i = 1$, the party becomes the puppet and follows the adversary's command exactly. If $c_i = 0$, the party plays a coercion-resistance strategy.

In the ideal world, a trusted party collects the real inputs, computes the ideal functionality $f(x_1, \cdots, f_m)$, and outputs the result. A protocol is receipt-free if, for every real-world adversary, there exists an ideal-world adversary whose output is indistinguishable from the real adversary's complete view for every choice of real inputs, fake inputs, and coercion-response bits. Intuitively, this guarantees that the real protocol does not give a coercer any power to determine if a voter obeyed than what is already available from the ideal computation's output. We refer readers to Section 4 in [52] for the complete definition.

Theorem 18. *For every PPT real-world adversary $\mathcal{A}$ against Haechi, there exists a PPT ideal-world adversary $\mathcal{I}$ such that, for every real vote vector $\boldsymbol{x}$, fake vote vector $\boldsymbol{x'}$, and coercion response vector $\mathbf{c}$,*

$$View_{\mathcal{A}}^{real}(\boldsymbol{x}, \boldsymbol{x'}, \mathbf{c}) \approx_c View_{\mathcal{I}}^{ideal}(\boldsymbol{x}, \boldsymbol{x'}, \mathbf{c}),$$

and $\mathcal{I}$ corrupts or coerces only the parties respectively corrupted or coerced by $\mathcal{A}$.

Proof. We show that for an adversary $\mathcal{A}$, we can build an ideal adversary $\mathcal{I}$: $\mathcal{I}$ internally executes $\mathcal{A}$. Whenever $\mathcal{A}$ corrupts a party p_i, $\mathcal{I}$ corrupts the corresponding ideal party. The ideal functionality reveals x_i and lets $\mathcal{I}$ replace it. If $\mathcal{A}$ sets p_i's vote as x''_i, $\mathcal{I}$ replaces x_i with x''_i.

Whenever $\mathcal{A}$ coerces a party p_i and $c_i = 1$ (a puppet party), $\mathcal{I}$ also coerces the corresponding ideal party exactly. Whenever $\mathcal{A}$ replaces x_i with x''_i, $\mathcal{I}$ also replaces x_i of the ideal party.

Whenever $\mathcal{A}$ coerces a party p_i and $c_i = 0$ (a resisting party), $\mathcal{I}$ also coerces the corresponding ideal party exactly. We define the coercion-resistance strategy for a resisting party as follows.

- When $\mathcal{A}$ requests internal information, p_i reports the fake vote x'_i.
- When $\mathcal{A}$ commands to cast x'_i, p_i secretly casts x_i and outputs the received ballot b_i to the adversary.
- When $\mathcal{A}$ commands to challenge, p_i submits x'_i to the device and outputs the received commitment, randomness, and auxiliary values to the adversary.

The ideal functionality gives $\mathcal{I}$ a fake input x'_i, but maintains the real input x_i for f.

After the executions, the view of $\mathcal{I}$ consists of its random coins, the real (replaced) inputs of corrupted and puppet parties $\{x''_i\}$, the faked inputs of resisting parties $\{x'_i\}$, and the output (tally result) of the ideal functionality,

$$T_{\mathbf{x}} = f(\tilde{x_1}, \cdots, \tilde{x_m})$$

where

$$\tilde{x_i} = \begin{cases} x_i, & \text{if } p_i \text{ is resisting or honest,} \\ x''_i, & \text{if } p_i \text{ is corrupted or a puppet,} \\ \perp & \text{if } \mathcal{A} \text{ forces } p_i \text{ to abstain.} \end{cases}$$

T includes the aggregated votes.

It remains to show that the components of $\mathcal{A}$'s view can be simulated from $\mathcal{I}$'s view. The $\mathcal{A}$'s view contains its random coins, the complete states of the corrupted and puppet parties, its interaction with resisting parties, and the final output T. In other words,

$$\mathsf{View}_{\mathcal{A}}^{\text{real}} = (\{x''_i\}_{i \in C \cup P}, \{x'_i\}_{i \in R}, BB, CH, T_{\mathbf{x}}, \pi_{\mathsf{Tally}})$$

where C, P, R refers the set of corrupted, puppet, and resisting parties respectively and CH refers to the information retrieved from challenge procedure.

We first show that the ideal adversary $\mathcal{I}$ can construct the challenge response that has the same distribution as CH. $\mathcal{I}$ constructs a simulated challenge transcript $\widehat{CH}$. For every challenge request made by $\mathcal{A}$ to a resisting party p_i, the simulator merely executes the challenge procedure with x'_i. It samples $\widehat{\sigma}_i^{\mathsf{ch}}$ and the auxiliary values $\widehat{\mathsf{aux}}_i^{\mathsf{ch}}$ according to the protocol. Then, it computes

$$\widehat{\mathsf{c}}_i^{\mathsf{ch}} = \mathsf{Com}\big(\widehat{\sigma}_i^{\mathsf{ch}}, (x'_i, \widehat{\mathsf{aux}}_i^{\mathsf{ch}})\big)$$

and returns

$$\widehat{CH}_i = \big(\widehat{\mathsf{c}}_i^{\mathsf{ch}}, \widehat{\sigma}_i^{\mathsf{ch}}, \widehat{\mathsf{aux}}_i^{\mathsf{ch}}\big).$$

Since the resisting party in the real world also challenges using x'_i and the simulator samples the challenge randomness and auxiliary values from the same distribution as in the real world, $\widehat{CH}_i$ and CH_i are identically distributed.

For the remaining proof, we show that the ideal adversary $\mathcal{I}$ can construct a simulated election record $\widehat{BB}$ from information retrieved throughout the ideal executions and show that $(T_{\mathbf{x}}, \widehat{\pi}_{\mathsf{Tally}}, \widehat{BB}) \approx_c (T_{\mathbf{x}}, BB)$.

The ideal adversary $\mathcal{I}$ internally executes $\mathcal{A}$ and constructs a simulated election record $\widehat{BB}$ as follows.

1. Construct valid simulated votes $\widehat{\mathbf{x}}$
   - For corrupted and puppet parties, set $\widehat{\mathbf{x}}_i = x''_i$.
   - For honest and resisting parties, choose values such that $\sum_{i \in \mathsf{Cast}(\widehat{BB})} \widehat{\mathbf{x}}_i = T_{\mathbf{x}}$.
2. For every simulated ballot, it runs the ballot-witness generation procedure on $\widehat{\mathbf{x}}_i$ to get valid auxiliary values $\widehat{\mathsf{aux}}_i$ and sample commitment randomness $\widehat{\sigma}_i$ according to the protocol. It then computes
$$\widehat{\mathsf{c}}_{\mathbf{y}_i} = \mathsf{Com}\left(\widehat{\sigma}_i, (\widehat{\mathbf{x}}_i, \widehat{\mathsf{aux}}_i)\right).$$
3. It uses the zero-knowledge simulator for Π_1 to generate the corresponding ballot well-formedness proofs $\widehat{\pi}_i$.
4. It generates the identifiers, identifier hashes, confirmation codes, and other derived election-record values according to Haechi.
5. It computes
$$\widehat{C} = \prod_{i \in \mathsf{Cast}(\widehat{BB})} \widehat{\mathsf{c}}_{\mathbf{y}_i}$$
and invokes the zero-knowledge simulator for Π_{Tally} on the public statement $(\widehat{C}, T_{\mathbf{x}})$ to obtain $\widehat{\pi}_{\mathsf{Tally}}$.

Now, we show that

$$(T_{\mathbf{x}}, \rho(T_{\mathbf{x}}), BB, \pi_{\mathsf{Tally}}) \approx_c (T_{\mathbf{x}}, \rho(T_{\mathbf{x}}), \widehat{BB}, \widehat{\pi}_{\mathsf{Tally}})$$

through the following hybrids.

Hybrid H_0. This is the real execution and produces

$$(T_{\mathbf{x}}, \rho(T_{\mathbf{x}}), BB, \pi_{\mathsf{Tally}}).$$

Hybrid H_1. Replace every ballot well-formedness proof in BB with a proof produced by the simulator for Π_1. Also replace π_{Tally} with a simulated proof for the public statement $(C, T_{\mathbf{x}})$, where

$$C = \prod_{i \in \mathsf{Cast}(BB)} \mathsf{c}_{\mathbf{y}_i}.$$

By zero-knowledge properties of Π_1 and Π_{Tally},

$$H_0 \approx_c H_1.$$

Hybrid H_2. Replace the commitments of honest and resisting parties with commitments to the simulated messages $(\widehat{\mathbf{x}}_i, \widehat{\mathsf{aux}}_i)$ using freshly sampled randomness $\widehat{\sigma}_i$. We proceed with the replacement one by one.

Whenever a commitment is replaced, recompute all public record values that depend on it and simulate the tally proof for the resulting aggregate commitment and the unchanged aggregate $T_{\mathbf{x}}$. Because both ballot and tally proofs are simulated, they reveal no information about the replaced witnesses.

Each commitment replacement is statistically indistinguishable by the perfect hiding property of Com. The simulated proofs, hashes, and confirmation codes can be computed efficiently. A hybrid argument over the polynomially many cast ballots gives

$$H_1 \approx_c H_2.$$

The aggregate selection vector remains $T_{\mathbf{x}}$ in H_2, while the aggregate randomness and auxiliary values remain hidden. The simulated tally proof is generated for the statement $(\widehat{C}, T_{\mathbf{x}})$ and is therefore distributed indistinguishably from a real tally proof.

Hybrid H_3. We now replace honestly generated identifiers with fresh identifiers sampled from the same distribution and recompute the identifier hashes and confirmation codes according to Haechi. The identifiers are identically distributed, and all derived values are computed in the same manner as in the real protocol. Hence,

$$H_2 \approx_c H_3.$$

In H_3, every component is distributed as in the simulated output

$$(T_{\mathbf{x}}, \rho(T_{\mathbf{x}}), \widehat{BB}, \widehat{\pi}_{\mathsf{Tally}}).$$

Thus,

$$\mathsf{View}_{\mathcal{A}}(\mathbf{x}, \mathbf{x}', \mathbf{c}) \approx_c \mathsf{View}^{\mathcal{I}}_{\mathcal{I}}(\mathbf{x}, \mathbf{x}', \mathbf{c}).$$

□