Compact Verifiable Shuffles for ElGamal Ciphertexts

Yuxi Xue, Xingye Lu, and Man Ho Au

The Hong Kong Polytechnic University, Hong Kong
yuxi-ivy.xue@connect.polyu.hk
{xing-ye.lu,mhaau}@polyu.edu.hk

Abstract. A verifiable shuffle proves that output ciphertexts are a rerandomized permutation of the inputs without revealing the permutation or rerandomization factors. It is a core primitive in mix-nets for electronic voting and blockchain-based anonymization, where each mix server publishes an auditable proof. Existing deployed schemes typically have proof size $O(N)$ or $O(\sqrt{N})$ in the number of ciphertexts N, making shuffle proofs a major bandwidth cost.

We present a logarithmic-size verifiable shuffle for ElGamal ciphertexts. The logarithmic term in our proof size is one third of that in the previously known logarithmic construction for the same shuffle relation by Hoffmann et al. (CCS 2019). Our protocol is public coin, non-interactive via the Fiat–Shamir transform, and relies on an updatable structured reference string that can be generated once in a powers-of-tau ceremony and reused across applications.

We implement the protocol in Rust and provide, to our knowledge, the first benchmarks for a logarithmic-size ElGamal shuffle. At $N = 2^{20}$, the proof is 2.6 KiB. In a four-server mix-net election, the four shuffle proofs occupy approximately 10.6 KiB in total, excluding ballot-validity and decryption proofs.

Keywords: Zero-knowledge proofs · Verifiable shuffles · Mix-nets · E-voting · ElGamal encryption

1 Introduction

Mix-nets, introduced by Chaum [11], are a foundational tool for anonymous communication. A mix-net processes a batch of encrypted messages through one or more mix servers. Each server secretly permutes and rerandomizes the ciphertexts before forwarding them. If at least one server is honest, the correspondence between inputs and outputs remains hidden. This paradigm is used in privacy-sensitive systems such as e-voting [19,4,3] and blockchain-based anonymization protocols. In mix-net-based e-voting, encrypted ballots are shuffled through one or more mix servers and are then decrypted, hiding the correspondence between voters' input ciphertexts and the published plaintext ballots. Verifiable mix-nets are currently used in Estonia's Internet-voting system and in the Swiss Post

system deployed in Swiss e-voting trials [38,39,40]. More recently, mix-net shuffles have also been proposed for blockchain-based coin-mixing protocols such as MixEth [37].

The central security requirement in these applications is *verifiability*. A malicious shuffler may drop, replace, or modify ciphertexts, so each mix server must prove in zero knowledge that its output ciphertexts are exactly a rerandomized permutation of its input ciphertexts. A protocol providing such a proof is called a *verifiable shuffle*, and its efficiency is measured by proof size, prover time, and verifier time. Proof size is especially important in publicly auditable systems because each shuffle proof becomes part of the permanent record. In e-voting, these proofs are posted to a public bulletin board, while in blockchain applications they may be stored and verified on-chain, where proof bytes incur storage and transaction costs. Since every mix server and every mixing round adds another proof, this proof-specific overhead accumulates across servers and rounds. These considerations make compact verifiable shuffle proofs a natural goal for secure publicly auditable systems.

A general-purpose zero-knowledge succinct non-interactive argument of knowledge (zkSNARK), such as PLONK [21] or Groth16 [25], can prove a shuffle relation by encoding it as an arithmetic circuit. For shuffles, however, this approach introduces substantial overhead. Routing N ciphertexts through a secret permutation requires a permutation network of size $\Omega(N \log N)$, and the shuffle relation is naturally expressed over elliptic-curve groups. When the SNARK's native field does not match the ciphertext group, these group operations must be implemented by non-native arithmetic inside the circuit. Dedicated shuffle arguments avoid this overhead by working directly with group elements and by exploiting the algebraic structure of rerandomized ElGamal ciphertexts.

Early verifiable shuffles [20,33,24,41] require $O(N)$ communication. Groth and Ishai [26] gave the first sublinear shuffle argument, and Bayer and Groth [6] later obtained an $O(\sqrt{N})$-size argument that became a standard benchmark. Hoffmann et al. [29] combined the Bayer–Groth approach with inner product arguments [7,9], giving an $O(\log N)$-size verifiable shuffle for ElGamal ciphertexts in the plain discrete-log setting with transparent setup. This is the closest prior work to ours. Other logarithmic-size shuffle arguments, such as CurdleProofs [12], address shuffles of public keys or commitments rather than the rerandomized ElGamal ciphertext shuffle relation considered here.

Our main contribution is a more compact logarithmic-size argument for rerandomized ElGamal shuffles. By combining a constant-size KZG permutation argument with a single aggregated inner-product argument for ciphertext consistency, our construction reduces the leading proof-size term of Hoffmann et al. [29] from $6 \log N$ to $2 \log N$ group elements. The resulting proof has size $(2 \log N + 11)\mathbb{G}_1 + 8\mathbb{F}$, which is 2.64 KiB for $N = 2^{20}$. The prover and verifier require $18N$ and $7N$ group exponentiations, respectively, with three additional pairings for verification. This reduction in proof size comes with a universal updatable structured reference string produced by a powers-of-tau ceremony [8]

Table 1. Comparison of verifiable shuffle protocols for N ciphertexts, where $N = mn$ when applicable.

Scheme	Proof Size	Prover	Verifier	Setup
[20]	$5N\,\mathbb{G}_1 + N\,\mathbb{F}$	$8N$ exp	$10N$ exp	Uniform
[19]	$5N\,\mathbb{G}_1 + N\,\mathbb{F}$	$9N$ exp	$10N$ exp	Uniform
[24]	$3N\,\mathbb{F}$	$6N$ exp	$6N$ exp	Uniform
[18]	$N\,\mathbb{G}_1 + 2N\,\mathbb{F}$	$7N$ exp	$8N$ exp	Uniform
[41]	$3N\,\mathbb{G}_1 + 4N\,\mathbb{F}$	$9N$ exp	$11N$ exp	Uniform
[31]	$\sim N\,\mathbb{G}_1 + 2N\,\mathbb{F}$	$6N$ exp	$6N$ exp	Uniform
[13]	$7N\,\mathbb{G}_1$	$16N$ exp	$18N$ p	Structured
[14]	$\sim 7N\,\mathbb{G}_1$	$\sim 7N$ exp	$\sim 3N$ exp + $3N$ p	Structured
[26]	$3m^2\,\mathbb{G}_1 + 3n\,\mathbb{F}$	$3mN$ exp	$4N$ exp	Uniform
[6]	$11m\,\mathbb{G}_1 + 5n\,\mathbb{F}$	$\sim 6N \log m$ exp	$4N$ exp	Uniform
[29]	$(6\log N{+}8)\,\mathbb{G}_1 + 4\,\mathbb{F}$	$30N$ exp	$10N$ exp	Uniform
This work	$(2\log N + 11)\mathbb{G}_1 + 8\mathbb{F}$	$18N$ exp	$7N$ exp + 3 p	Structured (updatable)

Notes. "exp" denotes a group exponentiation in $\mathbb{G}_1$, and "p" denotes a pairing computation. A uniform setup requires only publicly verifiable randomness. A structured setup uses an SRS whose trapdoor must remain unknown after setup. An updatable setup is structured but remains secure if at least one party contributes fresh honest randomness. Prior-work counts are reproduced as reported and may not use the same counting convention.

and reused for all shuffles up to a fixed maximum size, whereas Hoffmann et al. use a transparent setup.

Our contributions are as follows.

1. We present a verifiable shuffle for ElGamal ciphertexts whose proof has size $(2\log N + 11)\mathbb{G}_1 + 8\mathbb{F}$. At $N = 2^{20}$, the proof is 2.64 KiB. The protocol is public coin and can be non-interactive using the Fiat–Shamir transform.
2. We provide a Rust implementation over BLS12-381 and evaluate prover time, verifier time, and proof size. To the best of our knowledge, these are the first benchmarks for a verifiable shuffle of ElGamal ciphertexts with logarithmic proof size.
3. We discuss how our argument serves as the mixing-stage proof in verifiable mix-net elections and compare its public audit cost with prior shuffle arguments. For a four-server mix over $N = 2^{20}$ ballots, the shuffle proofs occupy about 10.6 KiB in total.

1.1 Technical Overview

At a high level, our shuffle proof follows the standard two-part structure used in prior verifiable shuffle protocols [33,24,6,29]: a proof that the prover knows a permutation, and a proof that the output ciphertexts are obtained by applying that permutation together with valid rerandomization factors. We follow this

structure for ElGamal ciphertexts. Let the input and output ciphertext vectors be $\mathbf{C} = (\mathsf{C}_1, \ldots, \mathsf{C}_N)$ and $\widetilde{\mathbf{C}} = (\widetilde{\mathsf{C}}_1, \ldots, \widetilde{\mathsf{C}}_N)$, where $\mathsf{C}_i = (E_i, F_i)$ and $\widetilde{\mathsf{C}}_i = (\widetilde{E}_i, \widetilde{F}_i)$. The witness is a permutation $\pi \in \Sigma_N$ and rerandomization factors $\boldsymbol{\rho} \in \mathbb{F}^N$ such that $\widetilde{\mathsf{C}}_i = \mathsf{C}_{\pi(i)}\,\mathsf{Enc}_Y(1, \rho_i)$ for all $i \in [N]$, where $\pi(i)$ denotes the input position corresponding to output position i.

Following [33,24,6,29], the verifier samples a random challenge x and uses $x, x^2, \ldots, x^N$ as position-dependent weights. A valid shuffle satisfies

$$\prod_{i=1}^{N} \mathsf{C}_i^{x^i} = \prod_{i=1}^{N} \left(\widetilde{\mathsf{C}}_i\,\mathsf{Enc}_Y(1, -\rho_i)\right)^{x^{\pi(i)}}.$$

Let $a_i := x^{\pi(i)}$ and $\boldsymbol{a} := (a_1, \ldots, a_N)$. The proof therefore has two tasks: showing that $\boldsymbol{a}$ is correctly derived from the committed permutation, and showing that the same vector satisfies the ciphertext consistency relation with $\boldsymbol{\rho}$.

Committing to the permutation The prover first commits to the permutation vector $\boldsymbol{p} := (\pi(1), \ldots, \pi(N))$ and the rerandomization vector $\boldsymbol{\rho}$. After receiving x, it computes vector $\boldsymbol{a}$. Let $H := \{\omega^i : i \in [N]\}$, where ω has order N, let $\mathcal{L}_i(X)$ be the Lagrange basis polynomial associated with ω^i, and let $Z_H(X) := \prod_{\xi \in H}(X - \xi) = X^N - 1$. The prover encodes $\boldsymbol{a}$ as the polynomial

$$f_a(X) := \textstyle\sum_i a_i \mathcal{L}_i(X) + (r_{a,0} + r_{a,1}X)Z_H(X)$$

and sends the hiding KZG commitment C_a to f_a.

One commitment, two readings The commitment C_a ties the two arguments together. In the permutation argument, it is used as a KZG commitment to the polynomial f_a. In the consistency argument, the same group element is read as the vector commitment $C_a = \mathbf{g}^{\boldsymbol{a}}\mathsf{u}_1^{r_{a,0}}\mathsf{u}_2^{r_{a,1}}$, where $r_{a,0}, r_{a,1}$ are hiding randomizers and $\mathbf{g}$ contains the Lagrange-basis commitment elements. These are two views of the same Lagrange-basis commitment to $\boldsymbol{a}$. Thus the permutation argument and the consistency argument share one commitment, and the protocol does not need a separate proof to link their witnesses.

Permutation argument After the verifier samples a second challenge y, the prover compresses each pair (p_i, a_i) to $p_i + ya_i$ and shows that the multiset $\{p_i + ya_i\}$ equals $\{i + yx^i\}$ over H. Equality of the compressed multisets implies, except with negligible probability over the choice of y, that

$$\{(p_i, a_i) : i \in [N]\} = \{(i, x^i) : i \in [N]\}.$$

Hence $\boldsymbol{p}$ is a permutation of $[N]$ and $a_i = x^{p_i}$ for every i. A grand-product argument over H, verified through a constant number of KZG openings, proves this relation with constant-size communication.

Consistency argument The permutation argument fixes $\boldsymbol{a}$, but it does not yet prove that this same vector relates the input and output ciphertexts. Write each ciphertext as $\mathsf{C}_i = (E_i, F_i)$ and $\widetilde{\mathsf{C}}_i = (\widetilde{E}_i, \widetilde{F}_i)$. Define

$$V_E := \prod_{i=1}^{N} E_i^{x^i}, \qquad V_F := \prod_{i=1}^{N} F_i^{x^i},$$

and let $\widetilde{\boldsymbol{E}} := (\widetilde{E}_1, \ldots, \widetilde{E}_N), \widetilde{\boldsymbol{F}} := (\widetilde{F}_1, \ldots, \widetilde{F}_N)$. If the shuffle relation holds, the homomorphic structure of ElGamal gives

$$V_E = \widetilde{\boldsymbol{E}}^{\boldsymbol{a}}\, \mathsf{g}^{\langle \boldsymbol{a}, -\boldsymbol{\rho} \rangle}, \quad V_F \;= \widetilde{\boldsymbol{F}}^{\boldsymbol{a}}\, Y^{\langle \boldsymbol{a}, -\boldsymbol{\rho} \rangle}, \quad C_a = \mathbf{g}^{\boldsymbol{a}}\, \mathsf{u}_1^{r_{a,0}}\, \mathsf{u}_2^{r_{a,1}}.$$

Together, these relations ensure that the vector committed in C_a is the same vector used in the ciphertext consistency check. The remaining steps combine these consistency relations into one inner-product argument.

1. The prover first commits to $\boldsymbol{\rho}$ as $C_\rho := \mathbf{k}^{\boldsymbol{\rho}} \mathsf{u}_\rho^{r_\rho}$.
2. The verifier samples a random challenge w. The two ciphertext equations above are combined by defining

   $$V_w := V_E V_F^w, \quad \mathbf{h}_w := \widetilde{\boldsymbol{E}} \circ \widetilde{\boldsymbol{F}}^w, \quad \mathsf{u}_w := \mathsf{g} Y^w,$$

   where $\circ$ denotes entry-wise multiplication. Then the two equations become $V_w = \mathbf{h}_w^{\boldsymbol{a}} \mathsf{u}_w^{\langle \boldsymbol{a}, -\boldsymbol{\rho} \rangle}$. This equation is checked together with the opening of C_a, so the vector used in the shuffle equation must be the same vector committed in C_a.
3. To make the proof zero-knowledge, the prover samples random mask vectors $\boldsymbol{s}_a, \boldsymbol{s}_\rho \in \mathbb{F}^N$. After receiving the challenge u, it forms

   $$\widehat{\boldsymbol{a}} := \boldsymbol{a} + u\boldsymbol{s}_a, \qquad \widehat{\boldsymbol{\rho}} := \boldsymbol{\rho} + u\boldsymbol{s}_\rho.$$

   These responses give the verifier three group elements: one for the masked opening of C_a, one for the masked opening of C_ρ, and one for the masked ciphertext equation. The prover then sends a single group element Q to ensure that the same masked $\boldsymbol{a}$ and $\boldsymbol{\rho}$ are used in all three checks. Finally, three random challenges combine these checks and Q into one IPA instance.

Together, these steps bind the permutation and consistency arguments through the single commitment C_a. The permutation argument proves that the vector committed in C_a is derived from the committed permutation values, while the IPA proves the aggregated ciphertext consistency equation with the committed rerandomization vector. Since the consistency conditions are combined into one IPA, the logarithmic part of the proof consists of a single $2 \log N$ inner-product transcript. Consequently, except with negligible probability, the output ciphertexts form a rerandomized permutation of the inputs.

2 Related Work

The study of verifiable shuffles has developed along several lines, with protocols differing in their proof size, setup assumptions, and whether they target ElGamal ciphertexts or related shuffle relations over commitments and public keys. We survey the most relevant constructions below.

Linear-size shuffle arguments The first paradigm, introduced by Furukawa and Sako [20], has the prover commit to a permutation matrix and use Schwartz–Zippel-style checks to show that it is well formed and consistent with the shuffled ciphertexts. Furukawa et al. [19] gave an implementation for large-scale elections, and Furukawa [17,18] later extended the idea to combine shuffle-and-decryption proofs and aggregate arguments for multi-server mix-nets. Wikström [42] decomposed the argument into offline and online phases, which Terelius and Wikström [41] simplified into a compact argument based on polynomial evaluation over restricted challenge spaces. This line has influenced practical mix-net systems such as Verificatum [43] and has been further optimized by Krips and Lipmaa [31]. The same broad permutation-matrix viewpoint also underlies a line of pairing-based non-interactive shuffle arguments in the common reference string model [28,32,13,15,14,2], which progressively improved efficiency and soundness guarantees. These constructions differ in their setup and computational costs, but their proof sizes remain linear in the number of shuffled ciphertexts.

Polynomial-root paradigm and sublinear shuffles The second major paradigm, due to Neff [33], exploits the invariance of polynomials under permutation of their roots. Groth [22,24] refined this approach with Pedersen commitments and multi-exponentiation techniques. Building on ideas from both paradigms, Groth and Ishai [26] achieved the first sublinear shuffle argument by arranging the N ciphertexts in an $m \times n$ matrix, obtaining $O(m^2 + n)$ communication at the cost of an $O(Nm)$ prover. Bayer and Groth [6] substantially improved this approach, bringing the communication down to $O(\sqrt{N})$ with an $O(N \log N)$ prover by combining Groth's sublinear arguments for linear algebra [23] with techniques for computing diagonal sums in the exponent.

Logarithmic-size schemes Hoffmann, Klooß, and Rupp [29] gave a logarithmic-size argument for rerandomized ElGamal ciphertext shuffles by combining the Bayer–Groth framework with inner product arguments [7,9]. Their scheme achieves $O(\log N)$ proof size, $O(N)$ prover complexity, and $O(N)$ verifier complexity, in the plain discrete-log setting with a transparent setup.

Recent works have also obtained logarithmic-size shuffle arguments in related settings. CurdleProofs [12] gives a logarithmic-size shuffle argument for public keys or commitments and was developed in the context of shuffle-based secret-leader-election proposals. Abdolmaleki et al. [1] propose a subset-checking approach whose communication is logarithmic in both the shuffled set size and the security parameter, relying on standard discrete-log assumptions together with a short integer solution (SIS) assumption.

Among these works, Hoffmann et al. [29] is the closest baseline because it proves the same ElGamal shuffle relation with logarithmic communication and transparent setup. Our work keeps logarithmic communication for the same relation, reduces the concrete proof size, and trades transparent setup for a universal updatable SRS.

3 Preliminaries

3.1 Notation

Let λ be the security parameter. All protocol parameters, including N and the supported polynomial degree, are polynomially bounded in λ. The complexity of algorithms is computed as a function of λ. Let $\mathsf{negl}(\lambda)$ denote an arbitrary negligible function. We write $\mathbb{F}$ for a finite field of prime order p (i.e., $\mathbb{F} = \mathbb{F}_p$). We denote by $\mathbb{F}^{<d}[X]$ the set of univariate polynomials over $\mathbb{F}$ of degree less than d, and by $\mathbb{F}^{\leq d}[X]$ those of degree at most d. For $n \in \mathbb{N}$, let $[n] := \{1, 2, \ldots, n\}$, and let Σ_n denote the set of permutations of $[n]$. For a finite set $S \subseteq \mathbb{F}$, define its vanishing polynomial as $Z_S(X) := \prod_{\alpha \in S}(X - \alpha)$. For a finite set S, we write $s \xleftarrow{\$} S$ to mean that s is sampled uniformly at random from S.

Vectors are typeset in bold. For a vector $\boldsymbol{f}$, we denote its i-th coordinate by f_i. We use $\circ$ to denote the Hadamard (entry-wise) product in the group, i.e., $(\boldsymbol{u} \circ \boldsymbol{v})_i = u_i v_i$. Group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are written multiplicatively with juxtaposition denoting the group operation. For a group vector $\boldsymbol{u} = (u_1, \ldots, u_n)$ and a scalar $a \in \mathbb{F}$, we write $\boldsymbol{u}^a := (u_1^a, \ldots, u_n^a)$ for component-wise exponentiation. For a field vector $\boldsymbol{a} = (a_1, \ldots, a_n)$, we write $\boldsymbol{u}^{\boldsymbol{a}} := \prod_{i=1}^{n} u_i^{a_i}$. For two vectors $\boldsymbol{x}, \boldsymbol{y}$ of the same type, $\boldsymbol{x} \| \boldsymbol{y}$ denotes their concatenation.

Bilinear groups We work with bilinear groups $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathsf{g}, \mathsf{h})$ where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of prime order p, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear pairing. The elements $\mathsf{g} \in \mathbb{G}_1$ and $\mathsf{h} \in \mathbb{G}_2$ are generators. We assume the existence of efficient algorithms for group operations, pairing evaluation, membership testing, and equality testing.

Derived generators For a public string seed and a length n, $\mathsf{Gen}(\mathsf{seed}, n) \in \mathbb{G}_1^n$ deterministically derives n domain-separated generators. In the random-oracle model, outputs derived from a new seed are modeled as fresh group handles.

3.2 Assumptions

Definition 1 (d-DLOG assumption). *Fix a positive integer d. The d-DLOG assumption for $(\mathbb{G}_1, \mathbb{G}_2)$ states that given*

$$\mathsf{g},\ \mathsf{g}^x,\ \ldots,\ \mathsf{g}^{x^d}, \quad \mathsf{h},\ \mathsf{h}^x,\ \ldots,\ \mathsf{h}^{x^d}$$

for uniformly chosen $x \xleftarrow{\$} \mathbb{F}$, the probability that any efficient adversary $\mathcal{A}$ outputs x is $\mathsf{negl}(\lambda)$.

Lemma 1 (Schwartz–Zippel [35]). *Let $f \in \mathbb{F}[X]$ be a non-zero polynomial of degree $d \geq 0$. Let S be a finite subset of $\mathbb{F}$, and let x be selected uniformly at random from S. Then $\Pr[f(x) = 0] \leq d/|S|$.*

Lemma 2 (Blinding [36]). *Let $S \subset \mathbb{F}$ with $|S| = n$. Fix a polynomial $f \in \mathbb{F}[X]$. Define the blinded polynomial $\widetilde{f}(X) := f(X) + Z_S(X)b(X)$, where $b(X) \in \mathbb{F}^{<k}[X]$ is a uniformly random polynomial of degree less than k. Then, for any k distinct points $x_1, \ldots, x_k \in \mathbb{F} \setminus S$, the vector $(\widetilde{f}(x_1), \ldots, \widetilde{f}(x_k))$ is uniformly distributed in $\mathbb{F}^k$.*

3.3 Commitments

We use the standard notions of perfect hiding and computational binding for commitment schemes; see, e.g., [27]. In particular, we use Pedersen commitments over a cyclic group $\mathbb{G}_1$ of prime order p.

Definition 2 (Pedersen vector commitment). *The setup independently samples $\mathbf{g} := (\mathsf{g}_1, \ldots, \mathsf{g}_n)$ and u uniformly from $\mathbb{G}_1 \setminus \{1\}$, and publishes them as commitment bases. To commit to a vector $\boldsymbol{m} = (m_1, \ldots, m_n) \in \mathbb{F}^n$ with randomness $\nu \in \mathbb{F}$, compute $\mathsf{Com}(\boldsymbol{m}; \nu) := \mathbf{g}^{\boldsymbol{m}}\, \mathsf{u}^{\nu}$. For $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{F}^n$ and $\nu, \nu' \in \mathbb{F}$, we have*

$$\mathsf{Com}(\boldsymbol{m}; \nu)\, \mathsf{Com}(\boldsymbol{m}'; \nu') = \mathsf{Com}(\boldsymbol{m} + \boldsymbol{m}';\, \nu + \nu').$$

Theorem 1. *For bases generated by the setup in Definition 2, the Pedersen vector commitment scheme, including its scalar special case $n = 1$, is perfectly hiding and computationally binding under the discrete logarithm assumption in $\mathbb{G}_1$.*

3.4 ElGamal Encryption

We employ the ElGamal cryptosystem in the group $\mathbb{G}_1$ of prime order p with base g.

- $\mathsf{KeyGen}(1^\lambda)$: sample $x \xleftarrow{\$} \mathbb{F}$, set $sk := x$, $Y := \mathsf{g}^x$, and $pk := Y$.
- $\mathsf{Enc}_Y(M, \rho)$: for $M \in \mathbb{G}_1$ and $\rho \in \mathbb{F}$, output $\mathsf{C} := (E, F) := (\mathsf{g}^\rho,\, Y^\rho M) \in \mathbb{G}_1^2$.
- $\mathsf{Dec}_{sk}(E, F)$: compute $M := FE^{-x}$.

ElGamal is multiplicatively homomorphic under component-wise ciphertext multiplication: for any $M_1, M_2 \in \mathbb{G}_1$ and $\rho_1, \rho_2 \in \mathbb{F}$,

$$\mathsf{Enc}_Y(M_1, \rho_1)\, \mathsf{Enc}_Y(M_2, \rho_2) = \mathsf{Enc}_Y(M_1 M_2, \rho_1 + \rho_2).$$

For ciphertexts $\mathsf{C}_i := \mathsf{Enc}_Y(M_i, \rho_i)$, define $\mathbf{C} := (\mathsf{C}_1, \ldots, \mathsf{C}_n)$. For a scalar vector $\boldsymbol{a} := (a_1, \ldots, a_n) \in \mathbb{F}^n$, we have

$$\mathbf{C}^{\boldsymbol{a}} = \textstyle\prod_{i=1}^n \mathsf{C}_i^{a_i} = \mathsf{Enc}_Y\Big(\prod_{i=1}^n M_i^{a_i},\ \sum_{i=1}^n a_i \rho_i\Big).$$

3.5 Interactive Arguments

We adopt the standard conventions and definitions for interactive arguments as presented in [5]. An interactive argument system for a relation $\mathcal{R}$ consists of three probabilistic polynomial-time (PPT) algorithms $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$:

- $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$: outputs a common reference string ck.
- $\mathcal{P}(\mathsf{ck}, x, w)$ and $\mathcal{V}(\mathsf{ck}, x)$: the prover and verifier interact; $\mathcal{P}$ holds the witness w while $\mathcal{V}$ does not.

The transcript is denoted $\mathrm{tr} \leftarrow \langle \mathcal{P}(\mathsf{ck}, x, w), \mathcal{V}(\mathsf{ck}, x) \rangle$. We write $\langle \mathcal{P}, \mathcal{V} \rangle = 1$ if the verifier accepts. Let $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ be a polynomial-time-decidable ternary relation. For a ck, we define $\mathcal{L}_{\mathsf{ck}} = \{x \mid \exists\, w : (\mathsf{ck}; x; w) \in \mathcal{R}\}$.

Definition 3 (Completeness). *An argument system $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ has completeness if for all non-uniform PPT adversaries $\mathcal{A}$:*

$$\Pr\left[\begin{array}{r|l} (\mathsf{ck}; x; w) \notin \mathcal{R} \vee & \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda), \\ \langle \mathcal{P}(\mathsf{ck}, x, w),\, \mathcal{V}(\mathsf{ck}, x) \rangle = 1 & (x, w) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \geq 1 - \epsilon_c(\lambda).$$

We say the system has perfect completeness if $\epsilon_c(\lambda) = 0$, and statistical completeness if ϵ_c is negligible.

Definition 4 (Knowledge soundness). *An argument system $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is knowledge sound for $\mathcal{R}$ if for every non-uniform PPT prover $\mathcal{P}^*$, there exists a PPT extractor $\mathcal{E}$ such that*

$$\Pr\big[\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda), (x, \mathrm{tr}, w) \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{ck}) : \mathrm{tr} \textit{ is accepting} \wedge (\mathsf{ck}; x; w) \notin \mathcal{R}\big] \leq \mathsf{negl}(\lambda).$$

The extractor has black-box access to $\mathcal{P}^*$, emulates the honest verifier, and may rewind $\mathcal{P}^*$.

Definition 5 (Witness-extended emulation [26, Definition 3]). *A public-coin argument $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ has witness-extended emulation if for all deterministic polynomial-time provers $\mathcal{P}^*$, there exists an expected polynomial-time emulator $\mathcal{E}$ such that for all non-uniform polynomial-time adversaries $\mathcal{A}$:*

$$\begin{aligned} &\Pr\big[\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda);\, (x, z) \leftarrow \mathcal{A}(\mathsf{ck}); \mathrm{tr} \leftarrow \langle \mathcal{P}^*(\mathsf{ck}, x, z), \mathcal{V}(\mathsf{ck}, x) \rangle : \mathcal{A}(\mathrm{tr}) = 1\big] \\ &\quad \approx \Pr\big[\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda);\, (x, z) \leftarrow \mathcal{A}(\mathsf{ck}); (\mathrm{tr}, w) \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{ck}, x, z) : \\ &\qquad \mathcal{A}(\mathrm{tr}) = 1 \wedge \big(\mathrm{tr} \textit{ is accepting} \Rightarrow (\mathsf{ck}; x; w) \in \mathcal{R}\big)\big]. \end{aligned}$$

Here z denotes auxiliary input given to the prover, and $\approx$ denotes computational indistinguishability. The emulator $\mathcal{E}$ emulates the honest verifier, has oracle access to $\mathcal{P}^*$, and may rewind it.

Definition 6 (Public coin). *An argument system is public coin if all verifier messages are chosen uniformly at random and independently of the prover's messages.*

Definition 7 (Special honest-verifier zero-knowledge). *A public-coin argument* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *for* $\mathcal{R}$ *is special honest-verifier zero-knowledge with simulation error* $\delta(\lambda)$ *if there exists a PPT simulator* $\mathcal{S}$ *such that for every non-uniform PPT adversary* $\mathcal{A}$*,*

$$\left| \Pr\left[\begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, w, \rho) \leftarrow \mathcal{A}(\mathsf{ck}) \\ \mathrm{tr} \leftarrow \langle \mathcal{P}(\mathsf{ck}, x, w), \mathcal{V}(\mathsf{ck}, x; \rho) \rangle : \\ (\mathsf{ck}; x; w) \in \mathcal{R} \wedge \mathcal{A}(\mathrm{tr}) = 1 \end{array}\right] - \Pr\left[\begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, w, \rho) \leftarrow \mathcal{A}(\mathsf{ck}) \\ \mathrm{tr} \leftarrow \mathcal{S}(\mathsf{ck}, x, \rho) : \\ (\mathsf{ck}; x; w) \in \mathcal{R} \wedge \mathcal{A}(\mathrm{tr}) = 1 \end{array}\right] \right| \leq \delta(\lambda).$$

Definition 8 (Trapdoor simulation). *A public-coin argument* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *admits trapdoor simulation with error* $\delta(\lambda)$ *if there exist PPT algorithms* $(\mathsf{SimSetup}, \mathcal{S})$ *such that* $\mathsf{SimSetup}(1^\lambda)$ *outputs* (ck, τ) *with* ck *distributed identically to the output of* $\mathsf{Setup}(1^\lambda)$*, and for every* (x, w) *with* $(\mathsf{ck}; x; w) \in \mathcal{R}$ *and every verifier randomness* ρ*, the simulated transcript* $\mathcal{S}(\mathsf{ck}, \tau, x, \rho)$ *is within statistical distance* $\delta(\lambda)$ *of the honest transcript* $\langle \mathcal{P}(\mathsf{ck}, x, w), \mathcal{V}(\mathsf{ck}, x; \rho) \rangle$*. Here* ρ *denotes the verifier's coins. The simulation is statistical if* $\delta(\lambda)$ *is negligible.*

3.6 Algebraic Group Model

We analyze security in the Algebraic Group Model (AGM) [16]. In the AGM, an adversary that outputs a group element also provides its algebraic representation with respect to the group elements it has seen so far. This lets us interpret a successful pairing check as a corresponding polynomial identity.

Definition 9 (Algebraic adversary [16, Definition 2]). *Let* $\mathbb{G}$ *be a cyclic group of prime order* p*. An algorithm* $\mathcal{A}$ *is algebraic if, whenever* $\mathcal{A}$ *outputs a group element* $Z \in \mathbb{G}$*, it also outputs a representation* $\boldsymbol{v} = (v_1, \ldots, v_k) \in \mathbb{F}^k$ *such that* $Z = \prod_{i=1}^{k} L_i^{v_i}$*, where* $\boldsymbol{L} = (L_1, \ldots, L_k)$ *is the list of all group elements* $\mathcal{A}$ *has received so far, including the SRS and protocol messages.*

We say an SRS has degree d if every element in $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) is of the form $\mathsf{g}^{f(\tau)}$ (resp. $\mathsf{h}^{f'(\tau)}$) for some $f, f' \in \mathbb{F}^{\leq d}[X]$, where τ is the SRS trapdoor.

Lemma 3 (Real vs. ideal checks [21, Lemma 4.5]). *Let* $\mathcal{A}$ *be an algebraic adversary interacting with a protocol whose SRS has degree* d*. Suppose* $\mathcal{A}$ *outputs group elements that, together with SRS elements, are used in a pairing check of the form* $\prod_{i=1}^{k} e(A_i, B_i) = 1_{\mathbb{G}_T}$*, where each* $A_i \in \mathbb{G}_1$ *and* $B_i \in \mathbb{G}_2$*. Let* $a_i(X), b_i(X) \in \mathbb{F}[X]$ *be the polynomials corresponding to these elements, i.e.,* $A_i = \mathsf{g}^{a_i(\tau)}$ *and* $B_i = \mathsf{h}^{b_i(\tau)}$*.*

Under the d-DLOG assumption, the probability that $\mathcal{A}$ *satisfies the pairing check while the corresponding polynomial identity* $\sum_{i=1}^{k} a_i(X) b_i(X) \equiv 0$ *does not hold is negligible in* λ*.*

Definition 10 (Knowledge soundness in the AGM). *A protocol* Π *for a relation* $\mathcal{R}$ *has knowledge soundness in the AGM if there exists a PPT extractor* $\mathcal{E}$ *such that for every algebraic adversary* $\mathcal{A}$*: if* $\mathcal{A}$ *interacts with the honest verifier and produces an accepting transcript with probability* ε*, then* $\mathcal{E}$*, given access to* $\mathcal{A}$*'s algebraic representations, outputs a witness* w *with* $(\mathsf{ck}; x; w) \in \mathcal{R}$ *with probability at least* $\varepsilon - \mathsf{negl}(\lambda)$*.*

3.7 KZG Polynomial Commitment Scheme

We use the KZG polynomial commitment scheme [30].

Definition 11 (KZG polynomial commitment scheme [30]). *For a maximum polynomial degree d, the scheme consists of:*

- $\mathsf{KZG.Setup}(1^\lambda, d)$*: sample a trapdoor* $\tau \xleftarrow{\$} \mathbb{F}$ *and output the structured reference string* $\mathsf{srs} := \left(\{\mathsf{g}^{\tau^i}\}_{i=0}^{d},\ \mathsf{h},\ \mathsf{h}^{\tau},\ \mathsf{h}^{\tau^2}\right) \in \mathbb{G}_1^{d+1} \times \mathbb{G}_2^3$.
- $\mathsf{KZG.Com}(\mathsf{srs}, f(X))$*: for* $f(X) \in \mathbb{F}^{\leq d}[X]$*, output* $C_f := \mathsf{g}^{f(\tau)} \in \mathbb{G}_1$.

KZG commitments are additively homomorphic in the exponent:

$$\mathsf{KZG.Com}(f_1)\,\mathsf{KZG.Com}(f_2) = \mathsf{KZG.Com}(f_1 + f_2).$$

When we use KZG openings below, the usual opening and verification equations are incorporated directly into the displayed batched pairing checks. We therefore do not specify separate opening algorithms in the protocol description.

Theorem 2 (Knowledge soundness of KZG in the AGM [30,21]). *Under the d-DLOG assumption, for any algebraic adversary $\mathcal{A}$ that outputs a KZG commitment C, the polynomial $f(X)$ satisfying $C = \mathsf{g}^{f(\tau)}$ can be read directly from $\mathcal{A}$'s algebraic representation. In particular, the committed polynomial is uniquely determined except with negligible probability.*

3.8 Inner Product Argument

We use the inner product argument of Bootle et al. [7] as refined by Bünz et al. [9]. The argument allows a prover to convince a verifier that a committed element P is of the form $\mathbf{G}^{\boldsymbol{\ell}}\,\mathbf{H}^{\boldsymbol{r}}\,\mathsf{U}^{\langle \boldsymbol{\ell}, \boldsymbol{r}\rangle}$ for hidden vectors $\boldsymbol{\ell}, \boldsymbol{r}$.

Definition 12 (Inner product relation). *Let $\mathbf{G}, \mathbf{H} \in \mathbb{G}_1^N$ be generator vectors, let $\mathsf{U} \in \mathbb{G}_1$ be a generator, and let $P \in \mathbb{G}_1$. The inner product relation is* $\mathcal{R}_{\mathrm{IPA}} := \left\{(P, \mathbf{G}, \mathbf{H}, \mathsf{U};\ \boldsymbol{\ell},\ \boldsymbol{r}) \ :\ \boldsymbol{\ell}, \boldsymbol{r} \in \mathbb{F}^N,\ P = \mathbf{G}^{\boldsymbol{\ell}}\,\mathbf{H}^{\boldsymbol{r}}\,\mathsf{U}^{\langle \boldsymbol{\ell}, \boldsymbol{r}\rangle}\right\}$.

We use the asymmetric folding variant of [29], which avoids inverse exponentiations in the recursive step. Let $m := \log N$. We present the protocol in Figure 1.

Theorem 3 ([7,9,29]). *Protocol Π_{IPA} is a $(2m+1)$-move public-coin argument for $\mathcal{R}_{\mathrm{IPA}}$ with perfect completeness and witness-extended emulation under the discrete logarithm assumption in $\mathbb{G}_1$, for any deterministic polynomial-time prover. The proof transcript consists of $2m$ group elements $(L_1, R_1, \ldots, L_m, R_m)$ and 2 field elements $(\widehat{\ell}, \widehat{r})$, where $m = \log N$.*

Protocol Π_{IPA}

1. Initialize $\mathbf{G}_0 := \mathbf{G}$, $\mathbf{H}_0 := \mathbf{H}$, $\boldsymbol{\ell}_0 := \boldsymbol{\ell}$, $\boldsymbol{r}_0 := \boldsymbol{r}$, $P_0 := P$.

2. For $j = 1$ to m:

(a) Parse $\boldsymbol{\ell}_{j-1} = (\boldsymbol{\ell}_L \| \boldsymbol{\ell}_R)$, $\boldsymbol{r}_{j-1} = (\boldsymbol{r}_L \| \boldsymbol{r}_R)$, $\mathbf{G}_{j-1} = (\mathbf{G}_L \| \mathbf{G}_R)$, $\mathbf{H}_{j-1} = (\mathbf{H}_L \| \mathbf{H}_R)$, each split into halves of size $N/2^j$.

(b) $\mathcal{P} \to \mathcal{V}$: compute $c_L := \langle \boldsymbol{\ell}_L, \boldsymbol{r}_R \rangle$, $c_R := \langle \boldsymbol{\ell}_R, \boldsymbol{r}_L \rangle$, and send $L_j := \mathbf{G}_R^{\boldsymbol{\ell}_L} \mathbf{H}_L^{\boldsymbol{r}_R} \mathsf{U}^{c_L}$, $R_j := \mathbf{G}_L^{\boldsymbol{\ell}_R} \mathbf{H}_R^{\boldsymbol{r}_L} \mathsf{U}^{c_R}$.

(c) $\mathcal{V} \to \mathcal{P}$: sample $\xi_j \xleftarrow{\$} \mathbb{F}^*$.

(d) Both parties fold: $\mathbf{G}_j := \mathbf{G}_L \, \mathbf{G}_R^{\xi_j}$, $\mathbf{H}_j := \mathbf{H}_L^{\xi_j} \, \mathbf{H}_R$, $P_j := L_j^{\xi_j^2} \, P_{j-1}^{\xi_j} \, R_j$. $\mathcal{P}$ folds: $\boldsymbol{\ell}_j := \xi_j \boldsymbol{\ell}_L + \boldsymbol{\ell}_R$, $\boldsymbol{r}_j := \boldsymbol{r}_L + \xi_j \boldsymbol{r}_R$.

3. $\mathcal{P} \to \mathcal{V}$: send final scalars $\widehat{\ell} := \boldsymbol{\ell}_m \in \mathbb{F}$ and $\widehat{r} := \boldsymbol{r}_m \in \mathbb{F}$.

4. $\mathcal{V}$ computes challenge-product vectors $\boldsymbol{s}, \boldsymbol{s}' \in \mathbb{F}^N$ with $s_i := \prod_{j=1}^m \xi_j^{\mathrm{bit}_j(i-1)}$, $s'_i := \prod_{j=1}^m \xi_j^{1-\mathrm{bit}_j(i-1)}$, where $\mathrm{bit}_j(i-1) \in \{0,1\}$ denotes the j-th bit in $i-1 = \sum_{j=1}^m \mathrm{bit}_j(i-1)\, 2^{m-j}$, and accepts iff

$$\mathbf{G}^{\boldsymbol{s}\,\widehat{\ell}} \mathbf{H}^{\boldsymbol{s}'\widehat{r}} \mathsf{U}^{\widehat{\ell}\widehat{r}} = P^{\eta_1} \prod_{j=1}^m (L_j^{\xi_j^2 \eta_{j+1}} R_j^{\eta_{j+1}}),$$

where $\eta_j := \prod_{k=j}^m \xi_k$ for $j \in [m]$ and $\eta_{m+1} := 1$.

Fig. 1. Protocol Π_{IPA}.

4 Verifiable Shuffle Argument

Let N be the number of ciphertexts to be shuffled. The public input consists of the ElGamal public key Y and two ciphertext vectors $\mathbf{C} := (\mathsf{C}_1, \ldots, \mathsf{C}_N)$ and $\widetilde{\mathbf{C}} := (\widetilde{\mathsf{C}}_1, \ldots, \widetilde{\mathsf{C}}_N)$, where $\mathsf{C}_i = (E_i, F_i) \in \mathbb{G}_1^2$. The prover holds a permutation $\pi \in \Sigma_N$ and rerandomization factors $\boldsymbol{\rho} \in \mathbb{F}^N$ such that $\widetilde{\mathsf{C}}_i = \mathsf{C}_{\pi(i)} \, \mathsf{Enc}_Y(1, \rho_i)$ for every $i \in [N]$. We define the shuffle relation by

$$\mathcal{R}_{\mathsf{vs}} := \left\{ \begin{array}{l} \big(\mathsf{ck}, Y, \mathbf{C}, \widetilde{\mathbf{C}}; \pi, \boldsymbol{\rho}\big) : \pi \in \Sigma_N, \boldsymbol{\rho} \in \mathbb{F}^N, \\ \quad \widetilde{\mathsf{C}}_i = \mathsf{C}_{\pi(i)} \, \mathsf{Enc}_Y(1, \rho_i) \text{ for all } i \in [N] \end{array} \right\}.$$

The protocol operates over the multiplicative subgroup $H := \{\omega^i : i \in [N]\} \subset \mathbb{F}$, where ω has order $N = 2^m$. Let $\mathcal{L}_i(X)$ be the i-th Lagrange basis polynomial over H, and define $Z_H(X) := \prod_{\xi \in H}(X - \xi) = X^N - 1$.

Structured reference string. The protocol uses an updatable powers-of-tau SRS that supports KZG commitments to polynomials of degree at most $N+3$. The setup samples the secret evaluation point $\tau \xleftarrow{\$} \mathbb{F} \backslash H$. The KZG part contains the monomial powers $\{\mathsf{g}^{\tau^i}\}_{i=0}^{N+3} \subset \mathbb{G}_1$ and the verification elements $\{\mathsf{h}, \mathsf{h}^{\tau}, \mathsf{h}^{\tau^2}\} \in \mathbb{G}_2$. From these powers we also use the Lagrange-basis elements $\mathbf{g} = (\mathsf{g}_1, \ldots, \mathsf{g}_N)$, where $\mathsf{g}_i = \mathsf{g}^{\mathcal{L}_i(\tau)}$. We also define $\mathsf{u}_j = \mathsf{g}^{\tau^{j-1} Z_H(\tau)}$ for $j \in \{1,2,3\}$, which are used for the $Z_H(X)$-multiple in the committed polynomials.

After the powers-of-tau contribution transcript is finalized and publicly verified, let $\mathsf{id}_{\mathrm{SRS}}$ be its canonical digest, with the first N entries of $\mathsf{Gen}(\mathsf{id}_{\mathrm{SRS}}, N+1)$

defining k and the final entry defining u_ρ. Each accepted update changes $\mathsf{id}_{\mathrm{SRS}}$ and thereby rederives this family, whose elements are modeled as independent group handles in the random-oracle model, while the ceremony's one-honest-participant assumption is used to keep τ unknown. We write the public parameters as

$$\mathsf{ck} := \left(\{\mathsf{g}^{\tau^i}\}_{i=0}^{N+3}, \mathbf{g}, \mathbf{k}, \mathsf{u}_1, \mathsf{u}_2, \mathsf{u}_3, \mathsf{u}_\rho, \mathsf{h}, \mathsf{h}^\tau, \mathsf{h}^{\tau^2}\right).$$

The public key Y is part of the statement, not part of the SRS.

The protocol commits only to witness polynomials of degree at most $N+3$. A polynomial of the form $\sum_i c_i \mathcal{L}_i(X) + (d_2X^2 + d_1X + d_0)Z_H(X)$ is committed as $\prod_{i=1}^N \mathsf{g}_i^{c_i} \mathsf{u}_3^{d_2} \mathsf{u}_2^{d_1} \mathsf{u}_1^{d_0}$.

Permutation as a grand product. Let $I_{\mathrm{id}}(X) := \sum_{i=1}^N i\mathcal{L}_i(X)$. The prover first commits to the permutation values $\boldsymbol{p} := (\pi(1), \ldots, \pi(N))$ through a blinded polynomial f_p. It also commits to the rerandomization vector $\boldsymbol{\rho}$, which is used later in the consistency argument. Once the verifier samples x, both parties define $I_x(X) := \sum_{i=1}^N x^i \mathcal{L}_i(X)$, and the prover commits to $\boldsymbol{a} := (x^{\pi(1)}, \ldots, x^{\pi(N)})$ through a blinded polynomial f_a whose restriction to H is $\boldsymbol{a}$. The permutation check compares the committed pairs (p_i, a_i) with the public pairs (i, x^i).

After the verifier samples y, define $d(X) := f_p(X) + yf_a(X)$ and $I(X) := I_{\mathrm{id}}(X) + yI_x(X)$. Their evaluations on H are $d(\omega^i) = p_i + ya_i$ and $I(\omega^i) = i + yx^i$, respectively. Thus the permutation check reduces to proving equality of the compressed multisets

$$\{p_i + ya_i : i \in [N]\} \quad \text{and} \quad \{i + yx^i : i \in [N]\}.$$

After receiving the shift α, the prover determines the grand-product evaluations by $z(\omega) = 1$, $z(\omega^{i+1}) = z(\omega^i)\frac{d(\omega^i)-\alpha}{I(\omega^i)-\alpha}$, $\qquad i = 1, \ldots, N-1$. It interpolates these evaluations and adds a random degree-two multiple of $Z_H(X)$, obtaining a blinded polynomial extension $z(X)$ with the same values on H. The protocol enforces the recurrence

$$(d(X) - \alpha)z(X) = (I(X) - \alpha)z(\omega X)$$

over H, together with the boundary condition $(z(X) - 1)\mathcal{L}_1(X) = 0$. A fresh challenge β_b combines these two conditions in the quotient identity

$$(d(X) - \alpha)z(X) - (I(X) - \alpha)z(\omega X) + \beta_b(z(X) - 1)\mathcal{L}_1(X) = Z_H(X)q(X).$$

Finally, the verifier samples $v \notin H$, receives $d_v := d(v)$ and $z_{\omega v} := z(\omega v)$, and verifies the quotient identity at v together with these two evaluation claims using one batched KZG opening.

4.1 Protocol Construction

The complete protocol combines the grand-product permutation check above with the ciphertext consistency argument of Section 5.

Protocol Π_{vs}. The prover $\mathcal{P}(\mathsf{ck}, Y, \mathbf{C}, \widetilde{\mathbf{C}}, \pi, \boldsymbol{\rho})$ and the verifier $\mathcal{V}(\mathsf{ck}, Y, \mathbf{C}, \widetilde{\mathbf{C}})$ interact as follows.

1. $\mathcal{P} \to \mathcal{V}$: set $\boldsymbol{p} := (\pi(1), \ldots, \pi(N))$. Sample blinding randomness $r_{p,0}, r_{p,1}, r_\rho \xleftarrow{\$} \mathbb{F}$, define

$$f_p(X) := \textstyle\sum_i p_i \mathcal{L}_i(X) + (r_{p,0} + r_{p,1} X) Z_H(X),$$

and send $C_p := \prod_i \mathsf{g}_i^{p_i} \mathsf{u}_1^{r_{p,0}} \mathsf{u}_2^{r_{p,1}}$. The prover also sends a commitment to the rerandomization vector $C_\rho := \prod_i \mathsf{k}_i^{\rho_i} \mathsf{u}_\rho^{r_\rho}$. This commitment is later opened in Π_{cons}.

2. $\mathcal{V} \to \mathcal{P}$: sample a random challenge $x \xleftarrow{\$} \mathbb{F}^*$ and send it.

3. $\mathcal{P} \to \mathcal{V}$: set $\boldsymbol{a} := (x^{p_1}, \ldots, x^{p_N})$. Sample blinding randomness $r_{a,0}, r_{a,1} \xleftarrow{\$} \mathbb{F}$, define

$$f_a(X) := \textstyle\sum_i a_i \mathcal{L}_i(X) + (r_{a,0} + r_{a,1} X) Z_H(X),$$

and send $C_a := \prod_i \mathsf{g}_i^{a_i} \mathsf{u}_1^{r_{a,0}} \mathsf{u}_2^{r_{a,1}}$.

4. $\mathcal{V} \to \mathcal{P}$: sample random challenges $y \xleftarrow{\$} \mathbb{F}^*, \alpha \xleftarrow{\$} \mathbb{F} \setminus \{i + yx^i : i \in [N]\}$, and send them. The challenge y compresses each pair (p_i, a_i) into one field element.

5. $\mathcal{P} \to \mathcal{V}$: sample blinding randomness $r_{z,0}, r_{z,1}, r_{z,2} \xleftarrow{\$} \mathbb{F}$. Compute

$$\begin{aligned} z(X) := {} & \big(r_{z,0} + r_{z,1} X + r_{z,2} X^2\big) Z_H(X) + \mathcal{L}_1(X) \\ & + \textstyle\sum_{i=1}^{N-1} \mathcal{L}_{i+1}(X) \prod_{j=1}^{i} \frac{p_j + y a_j - \alpha}{j + y x^j - \alpha}, \end{aligned}$$

and send $C_z := \prod_i \mathsf{g}_i^{z(\omega^i)} \mathsf{u}_1^{r_{z,0}} \mathsf{u}_2^{r_{z,1}} \mathsf{u}_3^{r_{z,2}}$.

6. $\mathcal{V} \to \mathcal{P}$: sample a random challenge $\beta_b \xleftarrow{\$} \mathbb{F}$ and send it.

7. $\mathcal{P} \to \mathcal{V}$: let

$$I(X) := I_{\mathrm{id}}(X) + y I_x(X), \quad d(X) := f_p(X) + y f_a(X).$$

On H, these polynomials encode $I(\omega^i) = i + yx^i, d(\omega^i) = p_i + y a_i$. Compute

$$\begin{aligned} q(X) := \tfrac{1}{Z_H(X)} \Big[& (d(X) - \alpha) z(X) - (I(X) - \alpha) z(\omega X) \\ & + \beta_b \big(z(X) - 1\big) \mathcal{L}_1(X) \Big], \end{aligned}$$

and send $C_q := \mathsf{KZG.Com}(q)$. The numerator vanishes on H, and q is its quotient by Z_H.

8. $\mathcal{V} \to \mathcal{P}$: sample a random challenge $v \xleftarrow{\$} \mathbb{F} \setminus (H \cup \{0\})$ for the quotient check and send it.

9. $\mathcal{P} \to \mathcal{V}$: send the evaluations $d_v := d(v), z_{\omega v} := z(\omega v)$.

10. $\mathcal{V} \to \mathcal{P}$: sample random batching challenges $\beta_d, \beta_z \xleftarrow{\$} \mathbb{F}$ for the two evaluation claims and send them.

11. $\mathcal{P} \to \mathcal{V}$: define

$$\begin{aligned} R(X) := {} & (d_v - \alpha) z(X) - (I(v) - \alpha) z_{\omega v} \\ & + \beta_b \big(z(X) - 1\big) \mathcal{L}_1(v) - Z_H(v) q(X). \end{aligned}$$

Compute

$$W(X) := \frac{R(X) + \beta_d(d(X) - d_v)}{X - v} + \frac{\beta_z(z(X) - z_{\omega v})}{X - \omega v},$$

and send $C_W := \mathsf{KZG.Com}(W)$. The polynomial W combines the quotient relation at v with $d(v) = d_v$ and $z(\omega v) = z_{\omega v}$ in one KZG opening witness.

12. The verifier computes $Z_H(v)$, $\mathcal{L}_1(v)$,

$$I(v) := I_{\mathrm{id}}(v) + yI_x(v), \qquad C_d := C_p C_a^y.$$

By homomorphism, C_d is the commitment to d. It reconstructs the commitment to R as

$$C_R := C_z^{d_v - \alpha + \beta_b \mathcal{L}_1(v)} \mathsf{g}^{-(I(v)-\alpha) z_{\omega v} - \beta_b \mathcal{L}_1(v)} C_q^{-Z_H(v)}.$$

It accepts this check iff

$$e\big(C_W, \mathsf{h}^{(\tau - v)(\tau - \omega v)}\big) = e\big(C_R C_d^{\beta_d} \mathsf{g}^{-\beta_d d_v}, \mathsf{h}^{\tau - \omega v}\big) \cdot e\big(C_z^{\beta_z} \mathsf{g}^{-\beta_z z_{\omega v}}, \mathsf{h}^{\tau - v}\big).$$

This pairing equation verifies the quotient relation at v together with the two evaluations batched in W.

13. Writing $\widetilde{\mathsf{C}}_i = (\widetilde{E}_i, \widetilde{F}_i)$, the verifier computes

$$V_E := \textstyle\prod_i E_i^{x^i}, \quad V_F := \prod_i F_i^{x^i},$$

and $\widetilde{\boldsymbol{E}} := (\widetilde{E}_1, \ldots, \widetilde{E}_N), \widetilde{\boldsymbol{F}} := (\widetilde{F}_1, \ldots, \widetilde{F}_N)$. Both parties run Π_{cons} of Section 5 on public input $(\mathsf{ck}, Y, V_E, V_F, C_a, C_\rho, \widetilde{\boldsymbol{E}}, \widetilde{\boldsymbol{F}})$, with prover witness $(\boldsymbol{a}, \boldsymbol{\rho}, r_{a,0}, r_{a,1}, r_\rho)$. The protocol Π_{cons} checks

$$V_E = \textstyle\prod_i \widetilde{E}_i^{a_i} \mathsf{g}^{-\langle \boldsymbol{a}, \boldsymbol{\rho} \rangle}, \quad V_F = \prod_i \widetilde{F}_i^{a_i} Y^{-\langle \boldsymbol{a}, \boldsymbol{\rho} \rangle}.$$

14. The verifier outputs acc iff the permutation argument and Π_{cons} both accept.

Fig. 2. Protocol Π_{vs}.

Fiat–Shamir instantiation. The non-interactive argument is obtained by applying the Fiat–Shamir transformation to the public-coin protocol above. The transcript is initialized with a protocol identifier and version, N, ck, the complete public statement, and an optional public context string. All prover messages are absorbed in protocol order before the subsequent challenges are derived, using distinct domain-separation labels. The group challenge s is generated by domain-separated hash-to-curve, while field challenges are generated by domain-separated hash-to-field, with rejection sampling whenever required.

Theorem 4. *In the algebraic group and random-oracle models, under the* $(N + 3)$*-DLOG assumption and the discrete logarithm assumption in* $\mathbb{G}_1$*, Protocol* Π_{vs} *is perfectly complete and knowledge sound for* $\mathcal{R}_{\mathsf{vs}}$*. It also admits statistical trapdoor simulation as in Definition 8.*

The proof is given in Appendix A.1.

Corollary 1 (Fiat–Shamir version). *In the classical random-oracle model, applying the multi-round Fiat–Shamir transformation to Protocol Π_{vs} yields a non-interactive argument for $\mathcal{R}_{\mathsf{vs}}$ with perfect completeness and knowledge soundness under the assumptions of Theorem 4, with the corresponding reduction loss of Attema et al. [5]. It is statistically zero knowledge in the programmable random-oracle model.*

The proof is given in Appendix A.2.

5 Consistency Argument

This section specifies the consistency argument invoked by Π_{vs} in Section 4.1, realized by the protocol Π_{cons}. It proves that the commitment C_a produced in the permutation argument is consistent with the shuffled ciphertexts.

Write $\widetilde{\boldsymbol{E}} := (\widetilde{E}_i)_{i\in[N]}$ and $\widetilde{\boldsymbol{F}} := (\widetilde{F}_i)_{i\in[N]}$. The public inputs are Y, V_E, V_F, C_a, C_ρ, together with $\widetilde{\boldsymbol{E}}$ and $\widetilde{\boldsymbol{F}}$. The witness is $(\boldsymbol{a}, \boldsymbol{\rho}, r_{a,0}, r_{a,1}, r_\rho)$. The consistency relation is defined by

$$\begin{aligned}\mathcal{R}_{\mathrm{cons}} := \{\ (\mathsf{ck}, Y, V_E, V_F, C_a, C_\rho, \widetilde{\boldsymbol{E}}, \widetilde{\boldsymbol{F}}; \boldsymbol{a}, \boldsymbol{\rho}, r_{a,0}, r_{a,1}, r_\rho) :\\ \boldsymbol{a}, \boldsymbol{\rho} \in \mathbb{F}^N, r_{a,0}, r_{a,1}, r_\rho \in \mathbb{F}, V_E = \widetilde{\boldsymbol{E}}^{\boldsymbol{a}}\, \mathsf{g}^{-\langle \boldsymbol{a}, \boldsymbol{\rho}\rangle},\\ V_F = \widetilde{\boldsymbol{F}}^{\boldsymbol{a}} Y^{-\langle \boldsymbol{a}, \boldsymbol{\rho}\rangle}, C_a = \mathbf{g}^{\boldsymbol{a}} \mathsf{u}_1^{r_{a,0}} \mathsf{u}_2^{r_{a,1}}, \quad C_\rho = \mathbf{k}^{\boldsymbol{\rho}} \mathsf{u}_\rho^{r_\rho}\ \}.\end{aligned}$$

Here C_a is the same commitment used in the permutation argument, while C_ρ fixes the rerandomization vector before the positional challenge. A random challenge w combines the two ciphertext equations. We write $\mathbf{h}_w := \widetilde{\boldsymbol{E}} \circ \widetilde{\boldsymbol{F}}^{\,w}$, $\mathsf{u}_w := \mathsf{g} Y^w$, and $V_w := V_E V_F^w$.

At a high level, the prover samples mask vectors $\boldsymbol{s}_a, \boldsymbol{s}_\rho \in \mathbb{F}^N$. After the verifier samples u, the protocol forms $\widehat{\boldsymbol{a}} := \boldsymbol{a} + u\boldsymbol{s}_a$ and $\widehat{\boldsymbol{\rho}} := \boldsymbol{\rho} + u\boldsymbol{s}_\rho$, together with three group elements $P_a, P_\rho, P_{\mathrm{ct}}$. These elements represent the masked opening of C_a, the masked opening of C_ρ, and the masked ciphertext equation, respectively. After they are fixed, the prover sends Q, which binds the same two masked vectors across these checks. The challenges r_1, r_2, r_3 then combine $Q, P_a, P_\rho, P_{\mathrm{ct}}$ into one IPA instance.

5.1 Protocol Construction

Protocol Π_{cons}. The prover $\mathcal{P}(\mathsf{ck}, Y, V_E, V_F, C_a, C_\rho, \widetilde{\boldsymbol{E}}, \widetilde{\boldsymbol{F}}, \boldsymbol{a}, \boldsymbol{\rho}, r_{a,0}, r_{a,1}, r_\rho)$ and the verifier $\mathcal{V}(\mathsf{ck}, Y, V_E, V_F, C_a, C_\rho, \widetilde{\boldsymbol{E}}, \widetilde{\boldsymbol{F}})$ interact as follows.

1. $\mathcal{P} \to \mathcal{V}$: sample random mask vectors $\boldsymbol{s}_a, \boldsymbol{s}_\rho \xleftarrow{\$} \mathbb{F}^N$ and blinding randomness $r_{s_a,0}, r_{s_a,1}, r_{s_\rho} \xleftarrow{\$} \mathbb{F}$. Send

$$M_a := \mathbf{g}^{\boldsymbol{s}_a} \mathsf{u}_1^{r_{s_a,0}} \mathsf{u}_2^{r_{s_a,1}}, \qquad M_\rho := \mathbf{k}^{-\boldsymbol{s}_\rho} \mathsf{u}_\rho^{r_{s_\rho}}.$$

2. $\mathcal{V} \to \mathcal{P}$: sample a random challenge $w \xleftarrow{\$} \mathbb{F}$ and a fresh independent group element $\mathsf{s} \xleftarrow{\$} \mathbb{G}_1 \setminus \{1\}$, and send (w, s). Both parties combine the two ciphertext equations by setting

$$\mathsf{h}_w := \widetilde{\boldsymbol{E}} \circ \widetilde{\boldsymbol{F}}^{\,w}, \quad \mathsf{u}_w := \mathsf{g} Y^w, \quad V_w := V_E V_F^w.$$

3. $\mathcal{P} \to \mathcal{V}$: write $\langle \boldsymbol{a} + X\boldsymbol{s}_a, -(\boldsymbol{\rho} + X\boldsymbol{s}_\rho)\rangle = t_0 + t_1 X + t_2 X^2$, where $t_0 := \langle \boldsymbol{a}, -\boldsymbol{\rho}\rangle$, $t_1 := \langle \boldsymbol{a}, -\boldsymbol{s}_\rho\rangle + \langle \boldsymbol{s}_a, -\boldsymbol{\rho}\rangle$, and $t_2 := \langle \boldsymbol{s}_a, -\boldsymbol{s}_\rho\rangle$. Sample $\delta_1, \delta_2 \xleftarrow{\$} \mathbb{F}$ and send

$$T_1 := \mathsf{h}_w^{\boldsymbol{s}_a} \mathsf{u}_w^{t_1} \mathsf{s}^{\delta_1}, \qquad T_2 := \mathsf{u}_w^{t_2} \mathsf{s}^{\delta_2}.$$

4. $\mathcal{V} \to \mathcal{P}$: sample $u \xleftarrow{\$} \mathbb{F}^*$ and send it.

5. $\mathcal{P} \to \mathcal{V}$: compute $\widehat{\boldsymbol{a}} := \boldsymbol{a} + u\boldsymbol{s}_a$ and $\widehat{\boldsymbol{\rho}} := \boldsymbol{\rho} + u\boldsymbol{s}_\rho$, and send

$$\mu_{a,0} := r_{a,0} + u r_{s_a,0}, \mu_{a,1} := r_{a,1} + u r_{s_a,1}, \mu_\rho := r_\rho - u r_{s_\rho}, \widehat{\delta} := u\delta_1 + u^2\delta_2.$$

6. The verifier computes

$$P_a := C_a M_a^u \mathsf{u}_1^{-\mu_{a,0}} \mathsf{u}_2^{-\mu_{a,1}}, \quad P_\rho := C_\rho^{-1} M_\rho^u \mathsf{u}_\rho^{\mu_\rho}, \quad P_{\mathrm{ct}} := V_w T_1^u T_2^{u^2} \mathsf{s}^{-\widehat{\delta}}.$$

For an honest prover,

$$P_a = \mathbf{g}^{\widehat{\boldsymbol{a}}}, \quad P_\rho = \mathsf{k}^{-\widehat{\boldsymbol{\rho}}}, \quad P_{\mathrm{ct}} = \mathsf{h}_w^{\widehat{\boldsymbol{a}}} \mathsf{u}_w^{-\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{\rho}}\rangle}.$$

7. $\mathcal{V} \to \mathcal{P}$: sample $\sigma \xleftarrow{\$} \{0,1\}^\lambda$ and send it. Both parties append σ to the current transcript tr and set $\mathbf{b} := \mathsf{Gen}(\mathrm{tr}, N)$.

8. $\mathcal{P} \to \mathcal{V}$: send $Q := \mathbf{b}^{\widehat{\boldsymbol{a}}} \mathbf{g}^{\widehat{\boldsymbol{\rho}}}$.

9. $\mathcal{V} \to \mathcal{P}$: sample random aggregation challenges $r_1, r_2, r_3 \xleftarrow{\$} \mathbb{F}^*$ and send them. Both parties set

$$P_{\mathrm{IPA}} := Q P_a^{r_1} P_\rho^{r_2} P_{\mathrm{ct}}^{r_3}, \mathbf{G}_{\mathrm{IPA}} := \mathbf{b} \circ \mathbf{g}^{r_1} \circ \mathsf{h}_w^{r_3}, \mathbf{H}_{\mathrm{IPA}} := \mathbf{g} \circ \mathsf{k}^{-r_2}, \mathsf{U}_{\mathrm{IPA}} := \mathsf{u}_w^{-r_3}.$$

10. The prover and verifier run one IPA on $(P_{\mathrm{IPA}}, \mathbf{G}_{\mathrm{IPA}}, \mathbf{H}_{\mathrm{IPA}}, \mathsf{U}_{\mathrm{IPA}})$, proving

$$P_{\mathrm{IPA}} = \mathbf{G}_{\mathrm{IPA}}^{\widehat{\boldsymbol{a}}} \mathbf{H}_{\mathrm{IPA}}^{\widehat{\boldsymbol{\rho}}} \mathsf{U}_{\mathrm{IPA}}^{\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{\rho}}\rangle}.$$

The verifier accepts iff the IPA verifier accepts.

Fig. 3. Protocol Π_{cons}.

Theorem 5. *In the algebraic group and random-oracle models, under the discrete logarithm assumption in $\mathbb{G}_1$ for the commitment bases derived in Section 4 and under the $(N+3)$-DLOG assumption for KZG binding of C_a, Protocol Π_{cons} is perfectly complete, has perfect special honest-verifier zero knowledge, and has witness-extended emulation for $\mathcal{R}_{\mathrm{cons}}$.*

The proof is given in Appendix A.3.

6 Efficiency Analysis and Implementation

6.1 Complexity Analysis

Proof size We count the prover-to-verifier communication, equivalently the proof published after applying the Fiat–Shamir transform. Verifier challenges are therefore not included. The proof contains six group elements outside Π_{cons}, namely $C_p, C_\rho, C_a, C_z, C_q, C_W$, and two field elements $d_v, z_{\omega v}$. The consistency argument contributes the five group elements M_a, M_ρ, T_1, T_2, Q. It also contributes the four field elements $\mu_{a,0}, \mu_{a,1}, \mu_\rho, \widehat{\delta}$ and one IPA transcript of $2\log N$ group elements and two field elements. Hence the total proof size is

$$(2\log N + 11)\mathbb{G}_1 + 8\mathbb{F}.$$

With compressed BLS12-381 encodings, using 48 bytes for a $\mathbb{G}_1$ element and 32 bytes for a field element, this is $96\log N + 784$ bytes, namely 1.42 KiB for $N = 2^7$ and 2.64 KiB for $N = 2^{20}$.

Computation convention We use Straus-style joint exponentiation for paired terms $A^a B^b$. Under this convention, the prover and verifier costs are $18N$ and $7N$, respectively, compared with ungrouped counts of $21N$ and $8N$. The prior-work counts in Table 1 are reproduced as reported.

Prover computation. Under this convention, the prover performs $18N$ group exponentiations and $\mathcal{O}(N\log N)$ field operations. The leading group-operation term consists of $6N$ exponentiations for the commitments, $4N$ for the consistency messages, and $8N$ for the IPA.

Verifier computation. The verifier performs mainly $7N$ group exponentiations and three pairings. Computing V_E and V_F contributes $2N$, constructing $\mathbf{h}_w$ contributes N, while the final consistency and IPA check contributes $4N$.

Both parties additionally perform N transcript-bound hash-to-curve evaluations per proof. At $N = 2^{12}$, a separate measurement showed that these evaluations accounted for only 2.8% of the prover time.

6.2 Implementation and Benchmarks

We implement the protocol in Rust[1], using `dusk-bls12_381`[2] and `dusk-plonk`[3] for BLS12-381 field arithmetic, group operations, and KZG commitments. Non-interactivity follows the Fiat–Shamir transcript schedule in Corollary 1. All benchmarks are run on a MacBook Pro with an Apple M3 Pro chip and 18 GB unified memory.

[1] Source code available at `https://github.com/yuxi16/verifiable-shuffle`.
[2] `https://crates.io/crates/dusk-bls12-381`
[3] `https://github.com/dusk-network/plonk`

Table 2 reports proof sizes only. For our scheme, the values are computed from $(2\log N + 11)\mathbb{G}_1 + 8\mathbb{F}$ using compressed BLS12-381 encodings. For prior logarithmic and linear shuffle arguments, the values are computed from the formulas in Table 1 using 32-byte group elements and 32-byte scalars, matching the setting used in those works.

Table 2. Proof-size comparison. All sizes are formula-based transcript-size estimates computed from Table 1. For our protocol, sizes use 48 B compressed $\mathbb{G}_1$ elements and 32 B field elements over BLS12-381. The three shaded baseline columns use 32 B group elements and 32 B scalars, matching the encoding assumptions used for the baseline estimates. Sizes are reported in KiB or MiB.

N	**Ours** (KiB)	[29] (KiB)	[6] (KiB)	[41]
2^7	1.42	1.69	6.75	28.0 KiB
2^8	1.52	1.88	8.00	56.0 KiB
2^9	1.61	2.06	13.5	112 KiB
2^{10}	1.70	2.25	16.0	224 KiB
2^{11}	1.80	2.44	27.0	448 KiB
2^{12}	1.89	2.63	32.0	896 KiB
2^{13}	1.98	2.81	54.0	1.75 MiB
2^{14}	2.08	3.00	64.0	3.50 MiB
2^{15}	2.17	3.19	108	7.00 MiB
2^{16}	2.27	3.38	128	14.0 MiB
2^{17}	2.36	3.56	216	28.0 MiB
2^{18}	2.45	3.75	256	56.0 MiB
2^{19}	2.55	3.94	432	112 MiB
2^{20}	2.64	4.13	512	224 MiB

Figure 4 plots the measured prover and verifier running time of the implementation. The plot uses a logarithmic scale on both axes to show the near-linear growth of the measured running time as the number of ciphertexts doubles.

Discussion. The measured running times exhibit the expected near-linear scaling. The proof size grows only through the IPA transcript, increasing from 1.42 KiB at $N = 2^7$ to 2.64 KiB at $N = 2^{20}$. As shown in Table 2, our proof is the smallest in the comparison for all reported sizes. At $N = 2^{20}$, it is about one third smaller than the logarithmic proof of Hoffmann et al. [29] and two to five orders of magnitude smaller than the square-root and linear-size proofs, despite using 48-byte compressed BLS12-381 $\mathbb{G}_1$ elements instead of 32-byte group elements.

The running times grow almost linearly with N, since both the prover and the verifier are dominated by linear-size MSMs and group exponentiations. The consistency argument, including the IPA, accounts for most of the prover time because it repeatedly processes changing bases across the IPA rounds, while the permutation argument uses a smaller number of large batched MSMs. The

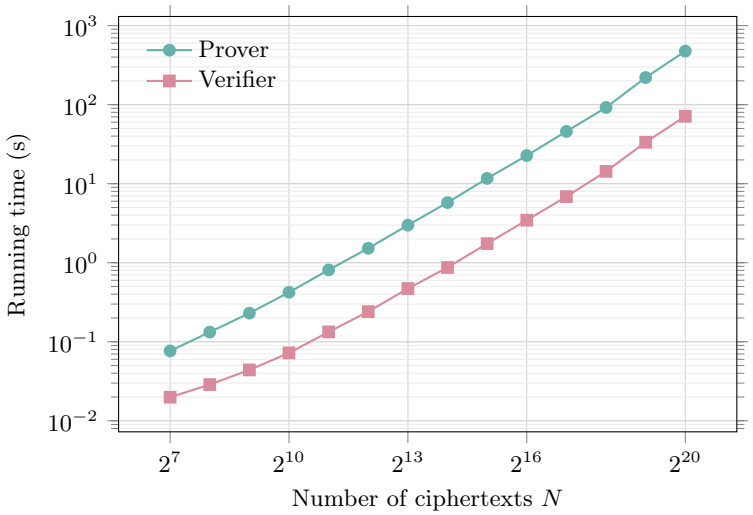

Fig. 4. Measured prover and verifier running time.

verifier is substantially faster because it performs fewer linear group operations and only three pairings for the batched KZG check.

7 Application to Verifiable Mix-Net Voting

In a shuffle-then-decrypt election, each mix server secretly permutes and rerandomizes the encrypted ballots and posts a proof for $\mathcal{R}_{\mathsf{vs}}$. Thus Π_{vs} can serve directly as the mixing-stage proof of a standard verifiable mix-net. Ballot-validity and decryption proofs are supplied by the surrounding voting system and lie outside our construction.

We recall the standard shuffle-then-decrypt architecture. It involves voters, each casting one encrypted ballot, a public append-only bulletin board BB that stores all ballots and proofs, a sequence of mix servers $M_1, \ldots, M_k$, a set of decryption trustees holding shares of the election secret key, and auditors who read BB and check the posted proofs. The election key is a threshold ElGamal key $Y = \mathsf{g}^x$, with x shared among the trustees by distributed key generation [34]. Privacy of the mixing stage assumes at least one honest mix server. Secrecy before tallying assumes fewer corrupted trustees than the decryption threshold. The bulletin board is assumed to provide a globally consistent and append-only public view, as in deployed verifiable mix-nets [43]. None of these components is introduced by us. We only specify the proof that each mix server posts.

The reusable commitment key ck is generated once according to the setup procedure of Section 4 and supports all elections and mixing rounds up to the configured maximum batch size. A voter with ballot $m \in \mathbb{G}_1$ samples $\rho \xleftarrow{\$} \mathbb{F}$, forms $\mathsf{C} = \mathsf{Enc}_Y(m, \rho)$, and posts it to BB together with a standard proof that

Table 3. Total shuffle proof size for a $k = 4$ server mix-net, as a function of the ballot count N. Each value is the four shuffle proofs posted for the mixing stage. Our values use compressed BLS12-381 encodings. The baselines use the proof-size formulas of Table 1 with 32-byte group elements and scalars.

N	This work	[29]	[6]	[41]
2^{10}	6.81 KiB	9.00 KiB	64.0 KiB	896 KiB
2^{16}	9.06 KiB	13.5 KiB	512 KiB	56.0 MiB
2^{20}	10.56 KiB	16.5 KiB	2.00 MiB	896 MiB

the ciphertext is well formed and encodes an admissible ballot. Let the accepted ballots be $\mathbf{C}^{(0)} = (\mathsf{C}_1, \ldots, \mathsf{C}_N)$. For $j = 1, \ldots, k$, server M_j reads $\mathbf{C}^{(j-1)}$, samples a fresh permutation $\pi_j \in \Sigma_N$ and rerandomization factors $\boldsymbol{\rho}_j \in \mathbb{F}^N$, and posts the new batch $\mathbf{C}^{(j)}$ with

$$\mathsf{C}_i^{(j)} = \mathsf{C}_{\pi_j(i)}^{(j-1)}\, \mathsf{Enc}_Y(1, \rho_{j,i}) \quad \text{for all } i \in [N],$$

together with a proof ϖ_j from Π_{vs} for the statement $(\mathsf{ck}, Y, \mathbf{C}^{(j-1)}, \mathbf{C}^{(j)})$. The Fiat–Shamir transcript is initialized as specified in Corollary 1, with the election identifier and mixing-round index as its public context. It also absorbs ck, Y, and both the input and output ciphertext batches before the first challenge, preventing a proof from being reused across elections or mixing rounds. After the last server, the trustees decrypt $\mathbf{C}^{(k)}$ by threshold decryption and post each plaintext with a proof of correct decryption [10]. An auditor accepts the mixing stage if $\varpi_1, \ldots, \varpi_k$ all verify, using only the public batches and proofs on BB.

The shuffle proofs are the main proof artifact added by the mixing stage to BB. A k-server mix posts k proofs of size $(2 \log N + 11)\mathbb{G}_1 + 8\mathbb{F}$ each, which is $k\,(96 \log N + 784)$ bytes over BLS12-381. This grows only logarithmically in the ballot count, so larger elections leave the per-server proof almost unchanged. For $N = 2^{20}$ ballots and $k = 4$ servers, the four shuffle proofs together occupy about 10.6 KiB. This number refers only to the proof portion of the mixing data, not to the ciphertext batches themselves. The directly comparable logarithmic-size shuffle of Hoffmann et al. [29] would post about 16.5 KiB for the same four proofs, and linear-size arguments post from hundreds of kilobytes to hundreds of megabytes. Table 3 gives the comparison across ballot counts, using the proof-size formulas of Table 1. Verifying the mixing stage costs k runs of the Π_{vs} verifier, each with a dominant cost of $7N$ group exponentiations and three pairings.

References

1. Abdolmaleki, B., Fauzi, P., Krips, T., Siim, J.: Shuffle arguments based on subset-checking. In: International Conference on Security and Cryptography for Networks. pp. 345–366. Springer (2024)
2. Abdolmaleki, B., Lipmaa, H., Siim, J., Zając, M.: A non-interactive shuffle argument with low trust assumptions. In: Cryptographers' Track at the RSA Conference. pp. 667–692. Springer (2020)

3. Aranha, D.F., Baum, C., Gjøsteen, K., Silde, T.: Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. pp. 1467–1481 (2023)
4. Aranha, D.F., Baum, C., Gjøsteen, K., Silde, T., Tunge, T.: Lattice-based proof of shuffle and applications to electronic voting. In: Cryptographers' Track at the RSA Conference. pp. 227–251. Springer (2021)
5. Attema, T., Fehr, S., Klooß, M.: Fiat–Shamir transformation of multi-round interactive proofs. In: Theory of Cryptography Conference. pp. 113–142. Springer (2022)
6. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 263–280. Springer (2012). https://doi.org/10.1007/978-3-642-29011-4_17
7. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 327–357. Springer (2016)
8. Bowe, S., Gabizon, A., Miers, I.: Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive (2017)
9. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE symposium on security and privacy (SP). pp. 315–334. IEEE (2018)
10. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: CRYPTO. pp. 89–105 (1992)
11. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **24**(2), 84–90 (1981). https://doi.org/10.1145/358549.358563
12. Ethereum Foundation Cryptography Research Team: Curdleproofs: A shuffle argument protocol. https://github.com/asn-d6/curdleproofs/blob/main/doc/curdleproofs.pdf (2022)
13. Fauzi, P., Lipmaa, H.: Efficient culpably sound NIZK shuffle argument without random oracles. In: Cryptographers' Track at the RSA Conference. pp. 200–216. Springer (2016)
14. Fauzi, P., Lipmaa, H., Siim, J., Zając, M.: An efficient pairing-based shuffle argument. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 97–127. Springer (2017)
15. Fauzi, P., Lipmaa, H., Zając, M.: A shuffle argument secure in the generic model. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 841–872. Springer (2016)
16. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Annual International Cryptology Conference. pp. 33–62. Springer (2018)
17. Furukawa, J.: Efficient, verifiable shuffle decryption and its requirement of unlinkability. In: International Workshop on Public Key Cryptography. pp. 319–332. Springer (2004)
18. Furukawa, J.: Efficient and verifiable shuffling and shuffle-decryption. IEICE transactions on fundamentals of electronics, communications and computer sciences **88**(1), 172–188 (2005)
19. Furukawa, J., Miyauchi, H., Mori, K., Obana, S., Sako, K.: An implementation of a universally verifiable electronic voting scheme based on shuffling. In: International Conference on Financial Cryptography. pp. 16–30. Springer (2002)

20. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Annual International Cryptology Conference. pp. 368–387. Springer (2001)
21. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive (2019)
22. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: International Workshop on Public Key Cryptography. pp. 145–160. Springer (2003)
23. Groth, J.: Linear algebra with sub-linear zero-knowledge arguments. In: Annual International Cryptology Conference. pp. 192–208. Springer (2009)
24. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. Journal of Cryptology **23**(4), 546–579 (2010)
25. Groth, J.: On the size of pairing-based non-interactive arguments. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 305–326. Springer (2016)
26. Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 379–396. Springer (2008)
27. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 253–280. Springer (2015)
28. Groth, J., Lu, S.: A non-interactive shuffle with pairing based verifiability. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 51–67. Springer (2007)
29. Hoffmann, M., Klooß, M., Rupp, A.: Efficient zero-knowledge arguments in the discrete log setting, revisited. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 2093–2110 (2019)
30. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16. pp. 177–194. Springer (2010)
31. Krips, T., Lipmaa, H.: More efficient shuffle argument from unique factorization. In: Cryptographers' Track at the RSA Conference. pp. 252–275. Springer (2021)
32. Lipmaa, H., Zhang, B.: A more efficient computationally sound non-interactive zero-knowledge shuffle argument. In: International Conference on Security and Cryptography for Networks. pp. 477–502. Springer (2012)
33. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Proceedings of the 8th ACM Conference on Computer and Communications Security. pp. 116–125 (2001). https://doi.org/10.1145/501983.502000
34. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: EUROCRYPT. pp. 522–526 (1991)
35. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. Journal of the ACM (JACM) **27**(4), 701–717 (1980)
36. Sefranek, M.: How (not) to simulate PLONK. In: International Conference on Security and Cryptography for Networks. pp. 96–117. Springer (2024)
37. Seres, I.A., Nagy, D.A., Buckland, C., Burcsi, P.: MixEth: Efficient, trustless coin mixing service for Ethereum. In: International Conference on Blockchain Economics, Security and Protocols (Tokenomics). OASIcs, vol. 71, pp. 13:1–13:20. Schloss Dagstuhl (2020)

38. State Electoral Office of Estonia: Activities of i-voting and voting on paper compared. https://www.valimised.ee/en/internet-voting/more-about-i-voting/activities-i-voting-and-voting-paper-compared, accessed: Jul. 31, 2026
39. Swiss Federal Chancellery: E-voting. https://www.bk.admin.ch/en/e-voting, accessed: Jul. 31, 2026
40. Swiss Post: E-voting community: Frequently asked questions. https://swisspost-digital.ch/en/evoting-community/help-and-contact/faq, accessed: Jul. 31, 2026
41. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: Progress in Cryptology – AFRICACRYPT 2010. Lecture Notes in Computer Science, vol. 6055, pp. 100–113. Springer (2010). https://doi.org/10.1007/978-3-642-12678-9_7
42. Wikström, D.: A commitment-consistent proof of a shuffle. In: Australasian Conference on Information Security and Privacy. pp. 407–421. Springer (2009)
43. Wikström, D.: Verificatum mix-net. http://www.verificatum.org (2010)

A Security Analysis

AGM convention. All knowledge soundness and witness-extended emulation claims in this section are for algebraic provers, and the extractor (resp. emulator) receives the algebraic representation of each output group element with respect to the public elements fixed at that point. We compare verifier equations as identities in this coordinate module, modulo the public SRS relations and the failure probability of the corresponding discrete logarithm or $(N+3)$-DLOG binding assumption. In particular, g is not treated as independent of $\mathbf{g}$, since $\sum_i \mathcal{L}_i(X) = 1$ gives $\mathsf{g} = \prod_i \mathsf{g}_i$. In Π_{cons}, the verifier samples s as a fresh independent group handle together with w, after the preceding transcript prefix has been fixed. Under the discrete logarithm assumption in $\mathbb{G}_1$, no efficient algebraic prover knows a nontrivial relation tying s to the preceding handles, so its coordinate is well defined. Hence all group elements determined by that prefix and by w, including V_w, $\mathbf{h}_w$, and u_w, have zero s-coordinate.

Lemma 4 (Nested public coin extraction). *The rewindings used below can be composed in expected time* $\mathrm{poly}(N, \lambda, 1/\varepsilon)$*, where* ε *is the acceptance probability of the algebraic prover.*

Proof. Use the standard expected time tree extractor for public coin protocols. At each node, fix an accepting prefix and resample only the next public challenge. The nested forks follow the order w, u, (r_1, r_2, r_3), and the IPA challenges. They use two accepting values of w, three accepting values of u, a constant-size fork tree for (r_1, r_2, r_3), and the standard logarithmic-depth IPA emulator. The challenge x is handled by the Schwartz–Zippel argument after the relevant commitments are fixed and requires no rewinding. The usual good prefix averaging argument gives the stated expected time.

Lemma 5 (Reading C_a). *Let* $f_a(X) = \sum_{i=1}^{N} a_i \mathcal{L}_i(X) + (r_{a,0} + r_{a,1}X)Z_H(X)$. *Then* $\mathsf{g}^{f_a(\tau)} = \prod_{i=1}^{N} \mathsf{g}_i^{a_i} \mathsf{u}_1^{r_{a,0}} \mathsf{u}_2^{r_{a,1}}$. *Thus the same group element is a KZG com-*

mitment to f_a and a vector commitment to $\boldsymbol{a} = (f_a(\omega^i))_{i\in[N]}$. Under the $(N+3)$-DLOG based KZG binding assumption, this vector is unique except with negligible probability.

Proof. The identity follows from $\mathsf{g}_i = \mathsf{g}^{\mathcal{L}_i(\tau)}$, $\mathsf{u}_1 = \mathsf{g}^{Z_H(\tau)}$, and $\mathsf{u}_2 = \mathsf{g}^{\tau Z_H(\tau)}$. Since Z_H vanishes on H, we have $f_a(\omega^i) = a_i$. Uniqueness follows from KZG binding for the supported degree bound.

Lemma 6 (Batched KZG opening separation). *Fix C_p, C_a, C_z, C_q and the values $x, y, \alpha, \beta_b, v, d_v, z_{\omega v}$ before β_d, β_z are sampled. The prover may choose C_W after seeing β_d, β_z. If the final batched KZG check accepts with nonnegligible probability, then, except with probability at most $2/|\mathbb{F}|+1/(|\mathbb{F}|-1)$ over (β_d, β_z, y) and the stated binding and extraction failure probabilities, $d_v = d(v), z_{\omega v} = z(\omega v)$, and*

$$\begin{aligned}&(d(v)-\alpha)z(v) - (I(v)-\alpha)z(\omega v)\\ &\quad + \beta_b(z(v)-1)\mathcal{L}_1(v) - Z_H(v)q(v) = 0.\end{aligned}$$

Proof. Project the accepted pairing equation to any AGM coordinate outside the KZG subspace. The h^{τ^2} coefficient forces the corresponding coordinate of C_W to be zero. Comparing the h^{τ} and h coefficients, and using the fresh challenges β_d, β_z, forces the same coordinate to vanish for the committed elements opened in the batched equation. Since $C_d = C_p C_a^y$ and y is sampled from $\mathbb{F}^*$ after C_p, C_a are fixed, any nonzero coordinate of C_p or C_a outside the KZG subspace can cancel in C_d for at most one value of y. Hence, except with probability at most $1/(|\mathbb{F}|-1)$, both C_p and C_a lie in the KZG subspace whenever C_d does. Thus all commitments opened in the batched equation lie in the KZG subspace, apart from the stated AGM and binding failure events.

KZG binding now gives degree bounded polynomials d, z, q, W. The pairing equation is equivalent to

$$\begin{aligned}0 = {}&(X-v)(X-\omega v)W(X)\\ &- (X-\omega v)\big(R(X) + \beta_d(d(X)-d_v)\big)\\ &- (X-v)\beta_z(z(X)-z_{\omega v}),\end{aligned}$$

where

$$\begin{aligned}R(X) = {}&(d_v-\alpha)z(X) - (I(v)-\alpha)z_{\omega v}\\ &+ \beta_b(z(X)-1)\mathcal{L}_1(v) - Z_H(v)q(X).\end{aligned}$$

Evaluating at $X = \omega v$ gives $\beta_z(z(\omega v) - z_{\omega v}) = 0$. Hence $z_{\omega v} = z(\omega v)$, except with probability $1/|\mathbb{F}|$. Evaluating at $X = v$ then gives $R(v) + \beta_d(d(v) - d_v) = 0$. If either $d(v) \neq d_v$ or $R(v) \neq 0$, this affine equation holds for at most one value of the fresh β_d. Thus, except with a further probability $1/|\mathbb{F}|$, we have $d(v) = d_v$ and $R(v) = 0$, which gives the claimed residual.

Lemma 7 (Blinding and trapdoor programming). *In the trapdoor simulation setting for the KZG SRS, the permutation transcript can be statistically simulated from the verifier challenges.*

Proof. The only opened values outside H are $d(v)$, $z(\omega v)$, and the single batched quotient relation. The blinders in f_p, f_a, and z are multiples of $Z_H(X)$, so by Lemma 2 they make these off domain values uniform. Given τ, the simulator chooses the opened values, interpolates degree bounded blinded polynomials consistent with them, and commits as $\mathsf{g}^{f(\tau)}$. It then programs q and W so that the batched identity holds at τ. These are exactly the honest blinding degrees of freedom, so the simulated distribution matches the honest one conditioned on the verifier challenges.

A.1 Proof of Theorem 4

Proof. Perfect completeness is immediate. For an honest prover, $d(\omega^i) = p_i + ya_i = \pi(i) + yx^{\pi(i)}$, so the grand product check is satisfied. The shuffle check is complete because $a_i = x^{\pi(i)}$ and $\widetilde{\mathsf{C}}_i = \mathsf{C}_{\pi(i)}\mathsf{Enc}_Y(1, \rho_i)$.

For knowledge soundness, let an algebraic prover make the verifier accept with nonnegligible probability. By KZG binding and Lemma 6, the accepted transcript defines degree bounded polynomials f_p, f_a, z, q, with $p_i = f_p(\omega^i)$ and $a_i = f_a(\omega^i)$. Define

$$\begin{aligned} G_{\beta_b}(X) = (d(X) - \alpha)z(X) - (I(X) - \alpha)z(\omega X) \\ + \beta_b(z(X) - 1)\mathcal{L}_1(X) - Z_H(X)q(X). \end{aligned}$$

The final opening check gives $G_{\beta_b}(v) = 0$. A malicious prover may use the full degree-$N+3$ KZG SRS for each committed polynomial; without a separate low-degree test, the verifier can therefore rely only on $\deg G_{\beta_b} \le 2N + 6$. Since q is fixed before $v \xleftarrow{\$} \mathbb{F} \setminus (H \cup \{0\})$, a nonzero G_{β_b} passes with probability at most $(2N + 6)/(|\mathbb{F}| - N - 1)$. We henceforth condition on $G_{\beta_b}(X) \equiv 0$.

For $i = 2, \ldots, N$, evaluation at $X = \omega^i$ gives

$$\big(d(\omega^i) - \alpha\big)z(\omega^i) = \big(I(\omega^i) - \alpha\big)z(\omega^{i+1}).$$

At $X = \omega$, let

$$A_1 = (d(\omega) - \alpha)z(\omega) - (I(\omega) - \alpha)z(\omega^2).$$

Then $A_1 + \beta_b(z(\omega) - 1) = 0$. Since A_1 and $z(\omega)$ are fixed before $\beta_b \xleftarrow{\$} \mathbb{F}$, except with probability $1/|\mathbb{F}|$ we have $A_1 = 0$ and $z(\omega) = 1$. Thus the recurrence holds for all $i \in [N]$, where $z(\omega^{N+1}) = z(\omega)$.

Let $z_i = z(\omega^i)$ and $S_{x,y} = \{i + yx^i : i \in [N]\}$. Since $\alpha \notin S_{x,y}$, every $I(\omega^i) - \alpha$ is nonzero. If some $z_i = 0$, the recurrence forces $z_{i+1} = 0$, and iterating cyclically contradicts $z_1 = 1$. Hence all z_i are nonzero, and multiplying the recurrences gives

$$\textstyle\prod_i (p_i + ya_i - \alpha) = \prod_i (i + yx^i - \alpha).$$

For fixed x, y, define

$$D_y(T) = \textstyle\prod_i \big(T - (p_i + ya_i)\big) - \prod_i \big(T - (i + yx^i)\big).$$

If the compressed multisets differ, D_y is nonzero of degree at most $N-1$. Since α is uniform in $\mathbb{F} \setminus S_{x,y}$, this occurs with probability at most

$$\frac{N-1}{|\mathbb{F}| - |S_{x,y}|} \leq \frac{N-1}{|\mathbb{F}| - N}.$$

Now define

$$F_{\mathrm{pair}}(Z,T) = \textstyle\prod_i \big(T - (p_i + Z a_i)\big) - \prod_i \big(T - (i + Z x^i)\big).$$

If the two multisets of pairs differ, some coefficient of $F_{\mathrm{pair}}(Z,T)$, viewed as a polynomial in T, is a nonzero polynomial in Z of degree at most N. Since $y \xleftarrow{\$} \mathbb{F}^*$ after C_p, C_a are fixed, $F_{\mathrm{pair}}(y,T) \equiv 0$ occurs with probability at most $N/(|\mathbb{F}|-1)$. Outside this event,

$$\{(p_i, a_i) : i \in [N]\} = \{(i, x^i) : i \in [N]\}.$$

There is a permutation π^* such that $p_i = \pi^*(i)$ and $a_i = x^{\pi^*(i)}$. Since C_p is sent before the verifier samples x, the extracted π^* is fixed before x.

By Lemma 5, the vector opened by C_a is the same $\boldsymbol{a}$. By Theorem 5, the consistency argument extracts $\boldsymbol{\rho}$ such that

$$V_E = \widetilde{\boldsymbol{E}}^{\boldsymbol{a}} \mathsf{g}^{\langle \boldsymbol{a}, -\boldsymbol{\rho} \rangle}, \quad V_F = \widetilde{\boldsymbol{F}}^{\boldsymbol{a}} Y^{\langle \boldsymbol{a}, -\boldsymbol{\rho} \rangle}.$$

Substitute $V_E = \prod_j E_j^{x^j}$, $V_F = \prod_j F_j^{x^j}$, $\widetilde{\boldsymbol{E}} = (\widetilde{E}_1, \ldots, \widetilde{E}_N)$, $\widetilde{\boldsymbol{F}} = (\widetilde{F}_1, \ldots, \widetilde{F}_N)$, and $a_i = x^{\pi^*(i)}$. Reindexing by $j = \pi^*(i)$ gives

$$\textstyle\prod_{j=1}^N \left(E_j \widetilde{E}_{\pi^{*-1}(j)}^{-1} \mathsf{g}^{\rho_{\pi^{*-1}(j)}} \right)^{x^j} = 1, \quad \prod_{j=1}^N \left(F_j \widetilde{F}_{\pi^{*-1}(j)}^{-1} Y^{\rho_{\pi^{*-1}(j)}} \right)^{x^j} = 1.$$

The statement and π^* are fixed before x. Moreover, C_ρ is sent before x, and the extracted witness opens this commitment to $\boldsymbol{\rho}$. Its binding therefore fixes $\boldsymbol{\rho}$ before x. If any coefficient above is not the identity, projection to a nonzero AGM coordinate gives a nonzero polynomial in x of degree at most N. Since $x \xleftarrow{\$} \mathbb{F}^*$, this occurs with probability at most $N/(|\mathbb{F}|-1)$ for each ciphertext component. Therefore, outside a total error of $2N/(|\mathbb{F}|-1)$,

$$\widetilde{E}_i = E_{\pi^*(i)} \mathsf{g}^{\rho_i}, \quad \widetilde{F}_i = F_{\pi^*(i)} Y^{\rho_i}$$

for all i. Thus $\widetilde{\mathsf{C}}_i = \mathsf{C}_{\pi^*(i)} \mathsf{Enc}_Y(1, \rho_i)$, and the extractor outputs $(\pi^*, \boldsymbol{\rho}) \in \mathcal{R}_{\mathsf{vs}}$. Conditioned on the binding and extraction events, the public-coin tests above contribute at most

$$\frac{2N+6}{|\mathbb{F}| - N - 1} + \frac{4}{|\mathbb{F}|} + \frac{N-1}{|\mathbb{F}| - N} + \frac{3N+8}{|\mathbb{F}| - 1}$$

to the soundness error. Here the last numerator combines the pair-compression error $N/(|\mathbb{F}|-1)$, the final x-interpolation error $2N/(|\mathbb{F}|-1)$, the possible y-collision when separating $C_d = C_p C_a^y$, and the $7/(|\mathbb{F}|-1)$ public-coin failure

probability from Lemma 10. The term $4/|\mathbb{F}|$ additionally includes the event $\mathsf{u}_w = 1$. In particular, the remaining public-coin error is $O(N/|\mathbb{F}|)$, in addition to the negligible AGM, random-oracle freshness, discrete-logarithm, and $(N+3)$-DLOG failure probabilities.

Statistical zero knowledge with trapdoor simulation follows by composing Lemma 7 with the simulator of Theorem 5. The KZG trapdoor programs the permutation commitments, the off domain openings, and the batched quotient commitment. The consistency transcript is simulated perfectly by Theorem 5; the remaining statistical distance is that of Lemma 7.

A.2 Proof of Corollary 1

Proof. Perfect completeness and knowledge soundness follow from Theorem 4 and the multi-round Fiat–Shamir transformation of Attema et al. [5]. Statistical zero knowledge follows from the trapdoor simulator by programming the random oracle.

A.3 Proof of Theorem 5

Lemma 8 (Fresh coordinate block). *Consider the interactive protocol before the fresh basis seed σ is sampled. After σ is appended to the current transcript, let $\mathbf{b} := \mathsf{Gen}(\mathrm{tr}, N)$. Except with negligible probability that the adversary predicts σ or a derived handle collides with a previously seen handle, the elements $\mathsf{b}_1, \ldots, \mathsf{b}_N$ are fresh handles under our derived-generator convention and form a fresh AGM coordinate block. Consequently, every group element output by the prover, or computed by the verifier from such elements, before σ is sampled has zero $\mathbf{b}$-coordinates.*

Proof. Before σ is sampled, the input to Gen is unavailable except with negligible probability. Every element already output therefore has an algebraic representation using only the handles available at that time. By the derived-generator convention, conditioned on neither bad event, the derived generators extend the representation module by a fresh coordinate block, so earlier representations have zero coordinates in that block.

Lemma 8 applies to $P_a, P_\rho, P_{\mathrm{ct}}$, to $\mathbf{g}, \mathbf{k}$, and to $\mathbf{h}_w = \widetilde{\boldsymbol{E}} \circ \widetilde{\boldsymbol{F}}^{w}$ and $\mathsf{u}_w = \mathsf{g}Y^{w}$. It does not apply to Q. No independence between the statement and the SRS is assumed, and the encryption key Y may be adversarial.

Lemma 9 (Admissibility of the combined IPA bases). *Condition on Lemma 8 and on $\mathsf{u}_w \neq 1$. Fix the messages sent before r_2 is sampled. Except for at most one value of $r_2 \in \mathbb{F}^*$, there is no nontrivial known algebraic relation among $\mathbf{G}_{\mathrm{IPA}}$, $\mathbf{H}_{\mathrm{IPA}}$, and $\mathsf{U}_{\mathrm{IPA}}$. Thus the failure probability is at most $1/(|\mathbb{F}| - 1)$.*

Proof. Suppose

$$\prod_i \mathsf{G}_{\mathrm{IPA},i}^{c_i} \prod_i \mathsf{H}_{\mathrm{IPA},i}^{d_i} \mathsf{U}_{\mathrm{IPA}}^{e} = 1.$$

Projection to the fresh $\mathbf{b}$-block gives $\boldsymbol{c} = \mathbf{0}$. The remaining relation is

$$\mathbf{g}^{\boldsymbol{d}}\mathbf{k}^{-r_2\boldsymbol{d}}\mathsf{u}_w^{-r_3 e} = 1.$$

Write the fixed coordinates of u_w in the $\mathbf{g}$-, $\mathbf{k}$-, and remaining blocks as $\boldsymbol{u}_g, \boldsymbol{u}_k, \boldsymbol{u}_\perp$. Since $r_3 \neq 0$, projection gives

$$\boldsymbol{d} = r_3 e \boldsymbol{u}_g, \qquad -r_2\boldsymbol{d} = r_3 e \boldsymbol{u}_k, \qquad e\boldsymbol{u}_\perp = \mathbf{0}.$$

If $e = 0$, then $\boldsymbol{d} = \mathbf{0}$. If $e \neq 0$, then $\boldsymbol{u}_\perp = \mathbf{0}$ and $r_2\boldsymbol{u}_g + \boldsymbol{u}_k = \mathbf{0}$. Since $\mathsf{u}_w \neq 1$, this vector equation has at most one scalar solution r_2.

Lemma 10 (Separation for the combined IPA). *Fix all messages sent before (r_1, r_2, r_3), including σ and Q, and fix (w, u). If the combined IPA accepts with nonnegligible probability over (r_1, r_2, r_3) and the IPA challenges, then, except for the AGM/IPA extraction error, the random-oracle freshness error, the event $\mathsf{u}_w = 1$, and a public-coin failure probability of at most $7/(|\mathbb{F}| - 1)$, the emulator extracts vectors $(\boldsymbol{a}^{(u)}, \boldsymbol{\rho}^{(u)})$, independent of (r_1, r_2, r_3), such that*

$$\begin{aligned} P_a &= \mathbf{g}^{\boldsymbol{a}^{(u)}}, & P_\rho &= \mathbf{k}^{-\boldsymbol{\rho}^{(u)}}, \\ Q &= \mathbf{b}^{\boldsymbol{a}^{(u)}}\mathbf{g}^{\boldsymbol{\rho}^{(u)}}, & P_{\mathrm{ct}} &= \mathbf{h}_w^{\boldsymbol{a}^{(u)}}\mathsf{u}_w^{-\langle \boldsymbol{a}^{(u)}, \boldsymbol{\rho}^{(u)}\rangle}. \end{aligned}$$

For uniform $w \in \mathbb{F}$, the event $\mathsf{u}_w = 1$ has probability at most $1/|\mathbb{F}|$.

Proof. Fix a pre-challenge prefix whose conditional acceptance probability exceeds the total bad-set bound. Condition on the fresh-coordinate and combined-basis admissibility events. The recursive messages L_j, R_j are generated after $\mathbf{b}$ is known and may have nonzero fresh coordinates. We therefore first invoke the standard IPA witness-extended emulator, which absorbs the complete folding transcript and returns vectors $(\boldsymbol{a}_*, \boldsymbol{\rho}_*)$ satisfying the unfolded relation

$$\begin{aligned} &QP_a^{r_1}P_\rho^{r_2}P_{\mathrm{ct}}^{r_3} \\ &\quad = \left(\mathbf{b} \circ \mathbf{g}^{r_1} \circ \mathbf{h}_w^{r_3}\right)^{\boldsymbol{a}_*}\left(\mathbf{g} \circ \mathbf{k}^{-r_2}\right)^{\boldsymbol{\rho}_*}\mathsf{u}_w^{-r_3\zeta}. \end{aligned} \tag{1}$$

Here $\zeta := \langle \boldsymbol{a}_*, \boldsymbol{\rho}_*\rangle$. A priori the extracted vectors may depend on all three outer challenges. Each polynomial below has coefficients fixed by the selected prefix. If it were nonzero, Schwartz–Zippel would bound the conditional accepting fraction by its degree divided by $|\mathbb{F}| - 1$; hence, outside the stated failure events, it is a polynomial identity.

Projecting (1) to the fresh coordinate block gives

$$\boldsymbol{a}_* = \boldsymbol{\alpha} := \pi_{\mathbf{b}}(Q),$$

which is fixed before (r_1, r_2, r_3).

Let π_g, π_k be projection to the blocks indexed by $\mathbf{g}$ and $\mathbf{k}$. Define

$$\begin{aligned} Q_g &= \pi_g(Q), & A_g &= \pi_g(P_a), & C_g &= \pi_g(P_\rho), \\ D_g &= \pi_g(P_{\mathrm{ct}}), & \boldsymbol{u}_g &= \pi_g(\mathsf{u}_w), & & \\ Q_k &= \pi_k(Q), & A_k &= \pi_k(P_a), & C_k &= \pi_k(P_\rho), \\ D_k &= \pi_k(P_{\mathrm{ct}}), & \boldsymbol{u}_k &= \pi_k(\mathsf{u}_w). & & \end{aligned}$$

and let G_h, K_h have columns $\pi_g(\mathsf{h}_{w,i})$ and $\pi_k(\mathsf{h}_{w,i})$. No one of these fixed vectors or matrices is assumed to vanish. Projection of (1) gives

$$\boldsymbol{\rho}_* = Q_g + r_1(A_g - \boldsymbol{\alpha}) + r_2 C_g + r_3(D_g - G_h\boldsymbol{\alpha} + \zeta \boldsymbol{u}_g), \tag{2}$$
$$r_2\boldsymbol{\rho}_* = -Q_k - r_1 A_k - r_2 C_k + r_3(K_h\boldsymbol{\alpha} - \zeta \boldsymbol{u}_k - D_k). \tag{3}$$

Now $\zeta = \langle \boldsymbol{\alpha}, \boldsymbol{\rho}_* \rangle$ is linear in the sole unknown vector.

Set $m_g = \langle \boldsymbol{\alpha}, \boldsymbol{u}_g \rangle$ and $m_k = \langle \boldsymbol{\alpha}, \boldsymbol{u}_k \rangle$. Taking inner products with $\boldsymbol{\alpha}$ gives

$$\zeta(1 - r_3 m_g) = M_g(r_1, r_2, r_3),$$
$$\zeta(r_2 + r_3 m_k) = M_k(r_1, r_2, r_3).$$

where

$$\begin{aligned} M_g &= \langle \boldsymbol{\alpha}, Q_g \rangle + r_1(\langle \boldsymbol{\alpha}, A_g \rangle - \langle \boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle) + r_2 \langle \boldsymbol{\alpha}, C_g \rangle \\ &\quad + r_3(\langle \boldsymbol{\alpha}, D_g \rangle - \langle \boldsymbol{\alpha}, G_h\boldsymbol{\alpha} \rangle), \\ M_k &= -\langle \boldsymbol{\alpha}, Q_k \rangle - r_1 \langle \boldsymbol{\alpha}, A_k \rangle - r_2 \langle \boldsymbol{\alpha}, C_k \rangle \\ &\quad + r_3(\langle \boldsymbol{\alpha}, K_h\boldsymbol{\alpha} \rangle - \langle \boldsymbol{\alpha}, D_k \rangle). \end{aligned}$$

Eliminating ζ without division gives

$$M_g(r_2 + r_3 m_k) = M_k(1 - r_3 m_g).$$

This polynomial has total degree at most two. Except with probability $2/(|\mathbb{F}|-1)$, coefficient comparison gives

$$\langle \boldsymbol{\alpha}, Q_k \rangle = \langle \boldsymbol{\alpha}, A_k \rangle = \langle \boldsymbol{\alpha}, C_g \rangle = 0,$$
$$\langle \boldsymbol{\alpha}, A_g \rangle = \langle \boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle, \qquad \langle \boldsymbol{\alpha}, C_k \rangle = -\langle \boldsymbol{\alpha}, Q_g \rangle,$$

and

$$\langle \boldsymbol{\alpha}, D_g \rangle - \langle \boldsymbol{\alpha}, G_h\boldsymbol{\alpha} \rangle = -m_g \langle \boldsymbol{\alpha}, Q_g \rangle.$$

Thus $M_g = \langle \boldsymbol{\alpha}, Q_g \rangle(1 - r_3 m_g)$. The factor $1 - r_3 m_g$ vanishes with probability at most $1/(|\mathbb{F}| - 1)$; outside that event,

$$\zeta = \langle \boldsymbol{\alpha}, Q_g \rangle$$

is fixed.

Substitute this constant into (2) and then into (3). The resulting vector-valued polynomial identity is

$$\begin{aligned} &r_2 Q_g + r_1 r_2(A_g - \boldsymbol{\alpha}) + r_2^2 C_g + r_2 r_3(D_g - G_h\boldsymbol{\alpha} + \zeta \boldsymbol{u}_g) \\ &\quad = -Q_k - r_1 A_k - r_2 C_k + r_3(K_h\boldsymbol{\alpha} - \zeta \boldsymbol{u}_k - D_k). \end{aligned}$$

It has total degree at most two. Except with a further probability $2/(|\mathbb{F}| - 1)$, its coefficients give

$$Q_k = A_k = C_g = \mathbf{0}, \quad A_g = \boldsymbol{\alpha}, \quad C_k = -Q_g,$$

and

$$D_g = G_h\boldsymbol{\alpha} - \zeta \boldsymbol{u}_g, \qquad D_k = K_h\boldsymbol{\alpha} - \zeta \boldsymbol{u}_k.$$

Substitution into (2) yields $\boldsymbol{\rho}_* = \boldsymbol{\beta} := Q_g$, fixed before the outer challenges, and $\zeta = \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle$.

Finally, rearranging (1) gives

$$\begin{aligned}\left(Q\mathbf{b}^{-\boldsymbol{\alpha}}\mathbf{g}^{-\boldsymbol{\beta}}\right)\left(P_a\mathbf{g}^{-\boldsymbol{\alpha}}\right)^{r_1}\left(P_\rho\mathbf{k}^{\boldsymbol{\beta}}\right)^{r_2} \\ \cdot \left(P_{\mathrm{ct}}\mathbf{h}_w^{-\boldsymbol{\alpha}}\mathsf{u}_w^{\langle\boldsymbol{\alpha},\boldsymbol{\beta}\rangle}\right)^{r_3} = 1.\end{aligned}$$

The four residuals are fixed before (r_1, r_2, r_3). Except with probability $1/(|\mathbb{F}|-1)$, this degree-one vector-valued polynomial is identically zero, so all residuals are the identity.

The two degree-two tests, the possible zero of $1 - r_3 m_g$, the final degree-one test, and Lemma 9 contribute at most $7/(|\mathbb{F}| - 1)$. The random-oracle freshness and IPA extraction errors are negligible. Finally, writing $Y = \mathsf{g}^y$, the equation $\mathsf{u}_w = \mathsf{g}Y^w = 1$ has at most one solution in w, giving the additional $1/|\mathbb{F}|$ term.

Remark 1 (Why the basis is derived at this point). A basis included in the public parameters would already be available when an adversarial statement is chosen and could occur in the algebraic representation of $\mathbf{h}_w$. The projection used above would then contain an unknown matrix multiplying $\boldsymbol{a}_*$ and would not directly determine it. Deriving $\mathbf{b}$ after the statement and the preceding messages are fixed supplies a coordinate block in which $\mathbf{h}_w$ is absent. This explains the role of the derived basis.

Proof. Perfect completeness follows by direct expansion. For an honest witness, after w the prover sends

$$T_1 = \mathbf{h}_w^{\boldsymbol{s}_a}\mathsf{u}_w^{t_1}\mathsf{s}^{\delta_1}, \qquad T_2 = \mathsf{u}_w^{t_2}\mathsf{s}^{\delta_2}.$$

With $\widehat{\boldsymbol{a}} = \boldsymbol{a} + u\boldsymbol{s}_a$ and $\widehat{\boldsymbol{\rho}} = \boldsymbol{\rho} + u\boldsymbol{s}_\rho$, the verifier obtains

$$P_a = \mathbf{g}^{\widehat{\boldsymbol{a}}}, \qquad P_\rho = \mathbf{k}^{-\widehat{\boldsymbol{\rho}}},$$

and

$$\begin{aligned}P_{\mathrm{ct}} &= V_w T_1^u T_2^{u^2}\mathsf{s}^{-\widehat{\delta}} \\ &= \mathbf{h}_w^{\widehat{\boldsymbol{a}}}\mathsf{u}_w^{t_0+ut_1+u^2t_2} = \mathbf{h}_w^{\widehat{\boldsymbol{a}}}\mathsf{u}_w^{-\langle\widehat{\boldsymbol{a}},\widehat{\boldsymbol{\rho}}\rangle}.\end{aligned}$$

After the responses are fixed, the verifier samples σ and both parties derive $\mathbf{b}$. The prover sends

$$Q = \mathbf{b}^{\widehat{\boldsymbol{a}}}\mathbf{g}^{\widehat{\boldsymbol{\rho}}}.$$

The combined IPA relation follows from the definitions of its bases.

For perfect special honest-verifier zero knowledge, fix all verifier challenges, including σ, r_1, r_2, r_3 and the IPA challenges, with all required field challenges

nonzero. The simulator first samples $\widehat{\boldsymbol{a}}, \widehat{\boldsymbol{\rho}} \xleftarrow{\$} \mathbb{F}^N$, $\mu_{a,0}, \mu_{a,1}, \mu_\rho, \widehat{\delta} \xleftarrow{\$} \mathbb{F}$, and $T_2 \xleftarrow{\$} \mathbb{G}_1$. It sets

$$P_a^* = \mathbf{g}^{\widehat{\boldsymbol{a}}}, \qquad P_\rho^* = \mathbf{k}^{-\widehat{\boldsymbol{\rho}}},$$
$$P_{\mathrm{ct}}^* = \mathbf{h}_w^{\widehat{\boldsymbol{a}}} \mathsf{u}_w^{-\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{\rho}} \rangle}.$$

It programs

$$M_a = (P_a^* C_a^{-1} \mathsf{u}_1^{\mu_{a,0}} \mathsf{u}_2^{\mu_{a,1}})^{1/u},$$
$$M_\rho = (P_\rho^* C_\rho \mathsf{u}_\rho^{-\mu_\rho})^{1/u}, \qquad T_1 = (P_{\mathrm{ct}}^* V_w^{-1} T_2^{-u^2} \mathsf{s}^{\widehat{\delta}})^{1/u}.$$

These choices determine the entire transcript prefix before σ. The simulator then appends σ to tr, derives $\mathbf{b} := \mathsf{Gen}(\mathrm{tr}, N)$ and sets

$$Q^* = \mathbf{b}^{\widehat{\boldsymbol{a}}} \mathbf{g}^{\widehat{\boldsymbol{\rho}}},$$

forms P_{IPA}^*, and runs the honest IPA prover with witness $(\widehat{\boldsymbol{a}}, \widehat{\boldsymbol{\rho}})$. No random-oracle programming or additional blinder for Q is needed. In an honest execution, the two masked vectors and the three μ-responses are uniform. Moreover, for fixed masks and nonzero u, the map

$$(\delta_1, \delta_2) \longmapsto \left(\mathsf{u}_w^{t_2} \mathsf{s}^{\delta_2}, u\delta_1 + u^2\delta_2\right)$$

is a bijection from $\mathbb{F}^2$ to $\mathbb{G}_1 \times \mathbb{F}$. Thus $(T_2, \widehat{\delta})$ is uniform, while T_1 is uniquely determined by the verifier equation. The commitment Q is computed from the same uniform masked vectors in both executions. Hence its joint distribution with $P_a^*, P_\rho^*, P_{\mathrm{ct}}^*$, and therefore the complete simulated transcript distribution, is identical to the honest one.

For witness-extended emulation, Lemma 4 allows the rewindings below to be composed in expected polynomial time. Fix the messages sent before u, and rewind u together with all later challenges. For each accepting fork, Lemma 10 gives vectors $(\boldsymbol{a}^{(u)}, \boldsymbol{\rho}^{(u)})$ satisfying the four separate relations.

Choose two accepting forks $u_0 \neq u_1$, and write $\boldsymbol{a}_j = \boldsymbol{a}^{(u_j)}$ and $\mu_{a,\ell,j} = \mu_{a,\ell}^{(u_j)}$. Define

$$\boldsymbol{s}_a = \frac{\boldsymbol{a}_1 - \boldsymbol{a}_0}{u_1 - u_0}, \qquad \boldsymbol{a} = \boldsymbol{a}_0 - u_0 \boldsymbol{s}_a,$$

and, for $\ell \in \{0, 1\}$,

$$r_{s_a,\ell} = \frac{\mu_{a,\ell,1} - \mu_{a,\ell,0}}{u_1 - u_0}, \qquad r_{a,\ell} = \mu_{a,\ell,0} - u_0 r_{s_a,\ell}.$$

Comparing $P_a = C_a M_a^u \mathsf{u}_1^{-\mu_{a,0}} \mathsf{u}_2^{-\mu_{a,1}} = \mathbf{g}^{\boldsymbol{a}^{(u)}}$ in the two forks gives

$$C_a = \mathbf{g}^{\boldsymbol{a}} \mathsf{u}_1^{r_{a,0}} \mathsf{u}_2^{r_{a,1}}, \qquad M_a = \mathbf{g}^{\boldsymbol{s}_a} \mathsf{u}_1^{r_{s_a,0}} \mathsf{u}_2^{r_{s_a,1}}.$$

By KZG binding and Lemma 5, $\boldsymbol{a}$ is the vector opened by C_a.

For the same forks, write $\boldsymbol{\rho}_j = \boldsymbol{\rho}^{(u_j)}$ and $\mu_{\rho,j} = \mu_\rho^{(u_j)}$. Define

$$\boldsymbol{s}_\rho = \frac{\boldsymbol{\rho}_1 - \boldsymbol{\rho}_0}{u_1 - u_0}, \qquad \boldsymbol{\rho} = \boldsymbol{\rho}_0 - u_0 \boldsymbol{s}_\rho,$$

and

$$r_{s_\rho} = \frac{\mu_{\rho,0} - \mu_{\rho,1}}{u_1 - u_0}, \qquad r_\rho = \mu_{\rho,0} + u_0 r_{s_\rho}.$$

Comparing the two equations for P_ρ gives

$$C_\rho = \mathsf{k}^{\boldsymbol{\rho}} \mathsf{u}_\rho^{r_\rho}, \qquad M_\rho = \mathsf{k}^{-\boldsymbol{s}_\rho} \mathsf{u}_\rho^{r_{s_\rho}}.$$

Moreover, $\boldsymbol{\rho}^{(u)} = \boldsymbol{\rho} + u\boldsymbol{s}_\rho$ and $\mu_\rho = r_\rho - u r_{s_\rho}$.

The same affine openings hold in every further accepting u-fork. Indeed, C_a, M_a, C_ρ, M_ρ are fixed before u, and their algebraic representations are binding. Hence any additional accepting fork is forced to satisfy

$$\boldsymbol{a}^{(u)} = \boldsymbol{a} + u\boldsymbol{s}_a, \qquad \boldsymbol{\rho}^{(u)} = \boldsymbol{\rho} + u\boldsymbol{s}_\rho.$$

It remains to extract the ciphertext equations. The separated ciphertext relation gives

$$P_{\mathrm{ct}} = \mathsf{h}_w^{\boldsymbol{a}+u\boldsymbol{s}_a} \mathsf{u}_w^{-\langle \boldsymbol{a}+u\boldsymbol{s}_a, \boldsymbol{\rho}+u\boldsymbol{s}_\rho \rangle}.$$

The verifier also defines $P_{\mathrm{ct}} = V_w T_1^u T_2^{u^2} \mathsf{s}^{-\widehat{\delta}}$. Let δ_1 and δ_2 be the fixed s-coordinates of T_1 and T_2. Since s is sampled independently of the preceding transcript prefix, the elements V_w, $\mathbf{h}_w$, and u_w have zero s-coordinate. Thus $\widehat{\delta} = u\delta_1 + u^2\delta_2$ in every accepting fork. Define $\overline{T}_1 = T_1 \mathsf{s}^{-\delta_1}$ and $\overline{T}_2 = T_2 \mathsf{s}^{-\delta_2}$. The remaining identity is

$$V_w \overline{T}_1^u \overline{T}_2^{u^2} = \mathsf{h}_w^{\boldsymbol{a}+u\boldsymbol{s}_a} \mathsf{u}_w^{-\langle \boldsymbol{a}+u\boldsymbol{s}_a, \boldsymbol{\rho}+u\boldsymbol{s}_\rho \rangle}.$$

This is a group-coordinate polynomial of degree at most two in u, with the same fixed coefficients in every fork. Three accepting forks with distinct values of u therefore force its constant coefficient to vanish, giving

$$V_w = \mathsf{h}_w^{\boldsymbol{a}} \mathsf{u}_w^{-\langle \boldsymbol{a}, \boldsymbol{\rho} \rangle}.$$

Finally, $V_w = V_E V_F^w$, $\mathbf{h}_w = \widetilde{\boldsymbol{E}} \circ \widetilde{\boldsymbol{F}}^w$, and $\mathsf{u}_w = \mathsf{g} Y^w$. Rewinding in w gives

$$V_E = \widetilde{\boldsymbol{E}}^{\boldsymbol{a}} \mathsf{g}^{\langle \boldsymbol{a}, -\boldsymbol{\rho} \rangle}, \qquad V_F = \widetilde{\boldsymbol{F}}^{\boldsymbol{a}} Y^{\langle \boldsymbol{a}, -\boldsymbol{\rho} \rangle}.$$