More Efficient Shuffle Argument from Unique Factorization

Toomas Krips[1] and Helger Lipmaa[1,2]

[1] University of Tartu, Estonia
[2] Simula UiB, Norway

Abstract. Efficient shuffle arguments are essential in mixnet-based e-voting solutions. Terelius and Wikström (TW) proposed a 5-round shuffle argument based on unique factorization in polynomial rings. Their argument is available as the `Verificatum` software solution for real-world developers, and has been used in real-world elections. It is also the fastest non-patented shuffle argument. We will use the same basic idea as TW but significantly optimize their approach. We generalize the TW characterization of permutation matrices; this enables us to reduce the communication without adding too much to the computation. We make the TW shuffle argument computationally more efficient by using Groth's coefficient-product argument (JOC, 2010). Additionally, we use batching techniques. The resulting shuffle argument is the fastest known ≤ 5-message shuffle argument, and, depending on the implementation, can be faster than Groth's argument (the fastest 7-message shuffle argument).

Keywords: Mix-net, shuffle argument, unique factorization

1 Introduction

A (zero knowledge [20]) shuffle argument enables a prover to convince a verifier that given two lists of ciphertexts (encrypted by using a suitable homomorphic, blindable public-key cryptosystem like Elgamal [11]) $[\boldsymbol{w}] = ([\boldsymbol{w}_1], \ldots, [\boldsymbol{w}_N])$ and $[\widehat{\boldsymbol{w}}] = ([\widehat{\boldsymbol{w}}_1], \ldots, [\widehat{\boldsymbol{w}}_N])$, she knows a permutation $\pi \in S_N$ and a vector of randomizers $\boldsymbol{s} = (s_1, \ldots, s_N)$, such that $[\widehat{\boldsymbol{w}}_i] = [\boldsymbol{w}_{\pi^{-1}(i)}] + \mathsf{Enc}_{\mathsf{pk}}(0; s_{\pi^{-1}(i)})$ for $i = 1, \ldots, N$.[3] On top of satisfying the intuitively clear soundness requirement, it is required that the verifier obtains no additional information except the truth of the statement; that is, a shuffle argument should be zero knowledge [20]. In particular, shuffle arguments are important in e-voting applications, [9], allowing one to anonymize encrypted ballots while preventing one from cheating, e.g., by

[3] Here, the computation is done in an *additive* cyclic group $\mathbb{G}$ of prime order q, and for a fixed generator $P = [1]$ of $\mathbb{G}$, we denote $xP = x[1]$ by $[x]$. We generalize this notation to matrices as in $([a_{ij}]) = [(a_{ij})] = (a_{ij}P)$. Nevertheless, when discussing efficiency, we use multiplicative terminology by (say) computing the number of exponentiations instead of the number of scalar multiplications. Finally, recall that an Elgamal encryption belongs to $\mathbb{G}^2$.

changing some of the ballots. Shuffle arguments and mix-nets have several other prominent applications, see [25] for further references. As an important recent emerging application, shuffle arguments have become popular in cryptocurrencies, [15].

Since the prover may not know any of the corresponding plaintexts, the construction of efficient shuffle arguments is nontrivial. While contemporary shuffle arguments are relatively efficient, they are conceptually quite complicated, often relying on (say) a novel characterization of permutation matrices. In particular, computationally most efficient shuffle arguments either offer less security (for example, the argument of [18] is not zero-knowledge) or rely on the CRS-model [6] and require a large number of rounds (unless one relies on the random oracle model [4] to make the argument non-interactive by using the Fiat-Shamir heuristic [16]). On the other hand, the proposed random oracle-less CRS-model non-interactive shuffle arguments [23,27,12,14,13] are computationally considerably less efficient. While the random oracle model and the Fiat-Shamir heuristic are dubious from the security viewpoint [8,19], there are no known attacks on random oracle-model shuffle arguments. Moreover, the most efficient random oracle-less shuffle arguments [27,12,14,13] are only proven to be sound in the generic group model that is also known to be problematic. For the sake of efficiency, we only consider interactive shuffle arguments with the implicit understanding that they can be made non-interactive by using the Fiat-Shamir heuristic. Nevertheless, it is preferable to minimize the number of rounds.

We recall three main paradigms used in the known *computationally* most efficient shuffle arguments. Other approaches are known, but they have, up to now, resulted in significantly less computation-efficient shuffle arguments. For example, an orthogonal direction is to minimize the communication and the verifier's computation at the cost of possibly larger prover's computation and the number of rounds; see [1].

First, the approach of Furukawa and Sako [18] relies on a specific characterization of permutation matrices. Namely, a matrix $\boldsymbol{M}$ is a permutation matrix if $\langle \boldsymbol{M}^{(i)}, \boldsymbol{M}^{(j)} \rangle = \delta_{ij}$ and $\langle \boldsymbol{M}^{(i)}, \boldsymbol{M}^{(j)} \odot \boldsymbol{M}^{(k)} \rangle = \delta_{ijk}$, where $\boldsymbol{M}^{(i)}$ is the ith column vector of $\boldsymbol{M}$, $\delta_{ij} = [i = j]$ is the Kronecker delta, $\delta_{ijk} = \delta_{ij}\delta_{ik}$, $\odot$ denotes the element-wise multiplication, and $\langle , \rangle$ denotes the scalar product. The Furukawa-Sako argument satisfies a privacy requirement that is weaker than zero knowledge. Later, Furukawa [17] made it more efficient and zero-knowledge. Importantly, shuffle arguments of this approach have only 3 messages.

Second, the approach of Neff [28] uses the fact that permuting the roots of polynomial results in the same polynomial. Groth [21] optimized Neff's argument, and the resulting argument is the most computationally efficient known shuffle argument. Unfortunately, the arguments that follow Neff's approach require 7 messages.

Terelius and Wikström (TW, [32]) proposed the third approach that uses the fact that $\mathbb{Z}_q[\boldsymbol{X}]$ is a unique factorization domain. This approach is based on another characterization of permutation matrices: namely, $\boldsymbol{M} \in \mathbb{Z}_q^{N \times N}$ is a permutation matrix iff

(a) $\prod_{i=1}^{N} \langle \boldsymbol{M}^{(i)}, X_i \rangle = \prod_{i=1}^{N} X_i$, where X_i are independent random variables, and
(b) $\boldsymbol{M} \cdot \mathbf{1}_N = \mathbf{1}_N$.

The TW approach results in shuffle arguments of the intermediate number of messages (namely, 5). However, up to now, it has resulted in somewhat higher computational complexity than the first two approaches; see Table 1 for an efficiency comparison.

Notably, the TW approach has some benefits that are important in practical applications. First, the first two approaches are patented, while the TW approach is not. Second, the TW approach is backed up by an available open-source software package[4] that has been used in several real-life electronic elections. Therefore, we see it as an important open problem to optimize the TW approach to the level of the first two approaches, if not above.

Our Contributions. We propose a more efficient version of the shuffle argument of Terelius and Wikström [32] (that is, the TW approach). It provides better computational and communication complexity than the argument of Terelius and Wikström as described in [32,34], due to the new characterization of permutation matrices, the use of Groth's coefficient-product argument, and more precise security analysis. Computationally, the resulting shuffle argument is the most efficient known ≤ 5-message shuffle argument, and comparable to the most efficient 7-message shuffle argument by Groth [21], see Table 1.

We also note that [32] only has a sketch of the security proof (although their main reduction is precise), while [34] lacks any security proofs. Thus, for the first time, we give full security proof of a unique-factorization-based shuffle argument.

In some aspects, the new argument is very similar to the 7-message argument of Groth [21] that corresponds to the second approach. A precise efficiency comparison between the new argument and Groth's argument is up to implementation since we replaced some multi-exponentiations with the same number of fixed-base exponentiations. See Table 1 for an efficiency comparison.

The new shuffle argument is broadly based on the TW shuffle argument, with three main technical changes that range from a generalization of the TW's characterization of permutation matrices to using a protocol from Groth's shuffle argument to using batch verification techniques.

First. In Section 4, we generalize the permutation matrix characterization of Terelius-Wikström. Namely, we call a family of non-zero polynomials $\psi_i(\boldsymbol{X})$, $1 \leq i \leq N$, *PM-evidential*, iff neither any non-linear sum of ψ_i nor any ψ_i^2 divides their product. Generalizing a result from [32], we prove that $\boldsymbol{M} \in \mathbb{Z}_q^{N \times N}$ is a permutation matrix iff the following holds:

(i) $\prod_{i=1}^{N} \langle \boldsymbol{M}^{(i)}, \psi_i(\boldsymbol{X}) \rangle = \prod_{i=1}^{N} \psi_i(\boldsymbol{X})$ for some PM-evidential polynomial family ψ_i, and

[4] `http://www.verificatum.org`

	Prover			Verifier			Prover + Ver.			♯round
	♯(N-wide m.e.)	$\frac{\sharp\text{exp.}}{N}$	♯(N-wide f.b.e.)	♯(N-wide m.e.)	$\frac{\sharp\text{exp.}}{N}$	♯(N-wide f.b..e.)	♯(N-wide m.e.)	$\frac{\sharp\text{exp.}}{N}$	♯(N-wide f.b.e.)	
Furukawa [17]	3	1	5	6	—	—	9	1	6	3
Verificatum [34]	9	1	—	9	—	—	18	1	—	5
Current paper	5	—	1	6	—	—	11	—	1	5
Groth [21]	6	—	—	6	—	—	12	—	—	7

Table 1: The complexity of some known interactive shuffle arguments, sorted by the number of rounds. Multi-exponentiations and fixed-based exponentiations are much more efficient than usual exponentiations.

(ii) $M \cdot \mathbf{1}_N = \mathbf{1}_N$.

In particular, we show that one can choose $\psi_i(X_1, X_2) = X_1^i + X_2$. The use of the novel characterization allows us to minimize communication: in one round of the communication, the TW shuffle verifier returns N new random variables X_i. In our case, 2 variables suffice.[5]

Second. In a *coefficient-product argument*, the prover proves that the product of the coefficients of a committed vector is equal to a publicly known integer. Almost all (non-multi-)exponentiations in the TW shuffle argument are executed during the coefficient-product argument. Instead of the coefficient-product argument of [32], we use a more efficient coefficient-product argument by Groth [21]. Together with batch verification techniques [2], this is the main technique that helps us to decrease the *computational* complexity of the TW argument.

Third. We use batch verification techniques to speed up the verifier. Intuitively, batch verification means that instead of checking two (or more) verification equations, one checks whether a random linear combination of them holds. Depending on the equations, this allows us to save some verifier's computation.

Discussion. We compare efficiency in Table 1, see Section 6 for more information. Note that Table 1 is not completely precise since many of the single exponentiations come as part of (say) 2-wide multi-exponentiations. In the new

[5] We note that one can use a PRG to generate N random variables from a random seed, and thus if this method is applicable, one does not have to use our optimization. However, there might be situations where one does not want to or cannot rely on a PRG.

shuffle argument, the prover executes four ($\approx N$)-wide multi-exponentiations and in addition approximately N fixed-base exponentiations. Since an N-wide multi-exponentiation is much more efficient than N (non fixed-base) exponentiations, the new argument is significantly faster than the arguments of the first approach or the argument of Terelius and Wikström [32].

The comparison of the new shuffle argument with the best arguments following Neff's approach like [21] is more complicated. Since Neff's approach does not use permutation matrices, it only uses multi-exponentiations. However, in the new shuffle argument, the only non-multi exponentiations are fixed-base exponentiations, and the prover has only to execute N of them. By using a version Straus's algorithm [31], the computation of N fixed-base exponentiations requires one to execute approximately $2N \log_2 q / \log_2(N \log_2 q)$ squarings or multiplications. This is comparable to the time required by Straus's multi-exponentiation algorithm, where one has to use approximately the same number of multiplications.

However, when one uses a parallel computation model (like a modern GPU), an N-wide multi-exponentiation induces a latency (at least $\Theta(\log N)$, to combine all N individual exponentiations) while N fixed-base exponentiations can be computed independently. Hence, a precise comparison depends on the used hardware platform. Moreover, the bit complexity of (multi-)exponentiations depends heavily on the bit-length of exponents. In the current paper, we provide a precise analysis of the size of exponents; such analysis in the case of the Groth's argument from [21] is still missing.

2 Preliminaries

Let q be a large prime. All arithmetic expressions with integers (for example, Eqs. (1) to (3)) are in $\mathbb{Z}_q$ by default, while commitments and ciphertexts belong to an order q cyclic additive group $\mathbb{G}$ of order q. For a fixed generator $P = [1]$ of $\mathbb{G}$, we denote $xP = x[1]$ by $[x]$. We generalize this notation to matrices as in $([a_{ij}]) = [(a_{ij})] = (a_{ij}P)$; e.g., $[a, b] = ([a], [b])$. We assume that $\mathsf{p} \leftarrow (\mathbb{G}, q, [1])$ is generated by a public algorithm $\mathsf{Pgen}(1^\lambda)$.

Implicitly, all vectors are column vectors. For a vector $\boldsymbol{a}$, let $wt(\boldsymbol{a})$ be the number of its non-zero coefficients. The dimension of all vectors is by default N. Let $\langle \boldsymbol{a}, \mathbf{b} \rangle = \sum_{i=1}^{N} a_i b_i$ be the scalar product of vectors $\boldsymbol{a}$ and $\mathbf{b}$ (in $\mathbb{Z}_q$, as usual). We denote the vertical concatenation of two vectors $\boldsymbol{a}$ and $\mathbf{b}$ by $\boldsymbol{a} \,/\!/\, \mathbf{b}$. Let $\boldsymbol{e}_i$ be the ith unit vector. For a matrix $\boldsymbol{M} = (M_{ij})_{ij} \in \mathbb{Z}_q^{N \times N}$, let $\boldsymbol{M}_i$ be its ith row vector, and $\boldsymbol{M}^{(j)}$ be its jth column vector. Let S_N be the symmetric group of N elements. For a permutation $\pi \in S_N$, the corresponding permutation matrix $\boldsymbol{M} = \boldsymbol{M}_\pi \in \mathbb{Z}_2^{N \times N}$ is defined by $M_{ij} = 1$ iff $i = \pi(j)$. Thus, a Boolean matrix $\boldsymbol{M}$ is a permutation matrix if it has a single 1 in every row and column. When $\mathcal{A}$ is a randomized algorithm, we denote by $a \leftarrow \mathcal{A}(\mathtt{inp}; r)$ the output of A on input $\mathtt{inp}$ given the random tape r.

Let λ be a security parameter; the complexity of algorithms is computed as a function of λ. Let $\mathsf{poly}(\lambda)$ / $\mathsf{negl}(\lambda)$ be an arbitrary polynomial / negligible function, and let $f(X) \approx_c g(X)$ iff $|f(X) - g(X)| = \mathsf{negl}(\lambda)$.

Lemma 1 (Schwartz-Zippel [35,30]). *Let $f \in \mathbb{F}[X_1, \ldots, X_n]$ be a non-zero polynomial of total degree $d \geq 0$ over a field $\mathbb{F}$. Let S be a finite subset of $\mathbb{F}$, and let $\mathsf{x}_1, \ldots, \mathsf{x}_n$ be selected at random independently and uniformly from S. Then* $\Pr[f(\mathsf{x}_1, \ldots, \mathsf{x}_n) = 0] \leq d/|S|$.

Cryptography. Let λ be the security parameter. For two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, $\mathcal{D}_1 \approx_c \mathcal{D}_2$ means that they are computationally indistinguishable by any non-uniform probabilistic polynomial-time adversary.

In the CRS model [6], a trusted third party generates a CRS crs together with a trapdoor. The CRS is published and given to all participants. The trapdoor is however kept secret, and only used in the proof of zero-knowledge to generate a simulated transcript of the protocol.

Commitment Schemes. A commitment scheme $\Gamma = (\mathsf{Pgen}, \mathsf{KGen}, \mathsf{Com})$ in the CRS model consists of a key generation algorithm KGen, that generates a commitment key ck (the CRS), and a commitment algorithm $\mathsf{Com}_{\mathsf{ck}}(a; \cdot)$ that first samples a random r and then uses it to commit to a message a. We also assume that ck implicitly contains p. Γ is *perfectly hiding* if a random commitment is statistically independent from the message. Γ is (T, ε)-*binding*, if for any probabilistic T-time adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{binding}}_{\Gamma,\mathcal{A}}(\lambda) \leq \varepsilon$, where

$$\mathsf{Adv}^{\mathrm{binding}}_{\Gamma,\mathcal{A}}(\lambda) := \Pr\begin{bmatrix}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{KGen}(\mathsf{p}); (\boldsymbol{a}_0, \boldsymbol{a}_1, r_0, r_1) \leftarrow \mathcal{A}(\mathsf{ck}); \\ \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{a}_0; r_0) = \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{a}_1; r_1) \wedge \boldsymbol{a}_0 \neq \boldsymbol{a}_1\end{bmatrix} .$$

We say that Γ is computationally binding if it is $(\mathsf{poly}(\lambda), \mathsf{negl}(\lambda))$-binding.

The extended Pedersen commitment scheme Γ_p [29] is defined as follows. Let $\mathsf{p} = (\mathbb{G}, q, [1])$. KGen samples $\mathsf{ck} = ([h_1], \ldots, [h_N]) \leftarrow\!\!\$\, \mathbb{G}^N$ uniformly at random. For $\boldsymbol{a} \in \mathbb{Z}_q^N$, let $\mathsf{Com}_{\mathsf{ck}}(\boldsymbol{a}; r) := r[1] + \langle \boldsymbol{a}, [\boldsymbol{h}] \rangle = r[1] + \sum_{i=1}^N a_i[h_i]$. Γ_p is computationally binding under the discrete logarithm assumption [7]. It is perfectly hiding since the distribution of $\mathsf{Com}_{\mathsf{ck}}(\boldsymbol{a}; r)$, $r \leftarrow\!\!\$\, \mathbb{Z}_q$, is uniform in $\mathbb{G}$.

The extended Pedersen commitment scheme is additively homomorphic, $\mathsf{Com}_{\mathsf{ck}}(\boldsymbol{a_0} + \boldsymbol{a_1}; r_0 + r_1) = \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{a_0}; r_0) + \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{a_1}; r_1)$. Clearly, from this it follows that for any integer n, $\mathsf{Com}_{\mathsf{ck}}(n\boldsymbol{a}; nr) = n \cdot \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{a}; r)$.

Cryptosystems. An public-key cryptosystem $\Pi = (\mathsf{Pgen}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of a key generation algorithm KGen, that generates a public key pk and a secret key sk, an encryption algorithm $[\boldsymbol{w}] \leftarrow \mathsf{Enc}_{\mathsf{pk}}(a; \cdot)$ that first samples a random r and then uses it to encrypt a message a, and a decryption algorithm $\mathsf{Dec}_{\mathsf{ck}}([\boldsymbol{w}])$ that uses sk to decrypt $[\boldsymbol{w}]$. Obviously, it is required that for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{p})$, $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(a; r)) = a$ for any a, r.

In the Elgamal cryptosystem [11], one samples $\mathsf{sk} \leftarrow\!\$\, \mathbb{Z}_q$ and then defines the public key as $\mathsf{pk} = [\begin{smallmatrix}1\\h\end{smallmatrix}]$, where $[h] \leftarrow \mathsf{sk}\cdot[1]$. Let $\mathsf{Enc}_{\mathsf{pk}}([m]; r) = [\begin{smallmatrix}c_1\\c_2\end{smallmatrix}] = \binom{[m]+r[h]}{r[1]}$. To decrypt a ciphertext, one computes $[c_1] - \mathsf{sk}[c_2] = [m]$. The Elgamal cryptosystem is IND-CPA secure, assuming that the Decisional Diffie-Hellman problem is hard. It is group-homomorphic, with $\mathsf{Enc}_{\mathsf{pk}}([m]; r) + \mathsf{Enc}_{\mathsf{pk}}([m]'; r') = \mathsf{Enc}_{\mathsf{pk}}([m] + [m]'; r + r')$.

In the lifted Elgamal, the plaintext m belongs to $\mathbb{Z}_q$, and the ciphertext is $\binom{m[1]+r[h]}{r[1]}$. Lifted Elgamal is $\mathbb{Z}_q$-homomorphic, with $\mathsf{Enc}_{\mathsf{pk}}(m; r) + \mathsf{Enc}_{\mathsf{pk}}(m'; r') = \mathsf{Enc}_{\mathsf{pk}}(m + m'; r + r')$.

The new shuffle argument allows to use both lifted and non-lifted Elgamal. Moreover, it allows one to work with a tuple of plaintexts $[\boldsymbol{m}] = ([m_1], \ldots, [m_M])^\top$, for some $M \geq 1$, that is encrypted as $\mathsf{Enc}_{\mathsf{pk}}([\boldsymbol{m}]; \boldsymbol{r}) = (\mathsf{Enc}_{\mathsf{pk}}([m_1]; r_1) \,/\!/\, \ldots \,/\!/\, \mathsf{Enc}_{\mathsf{pk}}([m_M]; r_M))$. For the sake of simplicity, we concentrate the exposition on the case of (non-lifted) Elgamal and $M = 1$. However, the case $M > 1$ is important in practice, e.g., in the case of complicated ballots.

Arguments. Let $\mathcal{R} = \{(\mathtt{inp}, \mathtt{wit})\}$, with $|\mathtt{wit}| = \mathsf{poly}(|\mathtt{inp}|)$, be a polynomial-time verifiable relation. Let $\mathcal{L}_{\mathcal{R}} = \{\mathtt{inp} : \exists \mathtt{wit}, (\mathtt{inp}, \mathtt{wit}) \in \mathcal{R}\}$. We allow $\mathcal{R}$ and $\mathcal{L}$ to depend on the common reference string (CRS) crs that is generated by a trusted third party by using an algorithm KGen; crs can say contain the description p of a group and the commitment key $\mathsf{ck} = [\{h_i\}]$ of the extended Pedersen commitment scheme. In this case, we denote $\mathcal{R}$ by $\mathcal{R}_{\mathsf{crs}}$ and $\mathcal{L}_{\mathcal{R}}$ by $\mathcal{L}_{\mathcal{R}:\mathsf{crs}}$. Thus, $(\mathsf{crs}, \mathtt{inp}, \mathtt{wit}) \in \mathcal{R}$ iff $(\mathtt{inp}, \mathtt{wit}) \in \mathcal{R}_{\mathsf{crs}}$.

For a randomized two-party protocol $(\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ between the prover P and the verifier V in the CRS model, let σ_p be the number of random choices the prover can make (thus, $\lceil \log_2 \sigma_p \rceil$ is the bit-length of the prover's random tape) and σ_v the number of random choices the verifier can make (thus, $\lceil \log_2 \sigma_v \rceil$ is the bit-length of the verifier's random tape). We require the protocol to be public-coin, at the end of which the verifier will either accept (outputs acc) or reject (outputs rej). We allow the acceptance to be probabilistic, and assume that the verifier does also output the coins he used to decide acceptance. For a protocol between prover P and verifier V in the CRS model, let $\langle \mathsf{P}(\mathsf{crs}, \mathtt{inp}, \mathtt{wit}), \mathsf{V}(\mathsf{crs}, \mathtt{inp}) \rangle$ be the whole transcript between P and V, given crs, common input $\mathtt{inp}$ and prover's private input (witness) $\mathtt{wit}$. We sometimes write $\langle \mathsf{P}(\mathsf{crs}, \mathtt{inp}, \mathtt{wit}), \mathsf{V}(\mathsf{crs}, \mathtt{inp}) \rangle = \mathsf{acc}$ to denote that the verifier accepts, and rej if V rejects. The concrete meaning will be clear from the context.

Then, $(\mathsf{Pgen}, \mathsf{KGen}, \mathsf{P}, \mathsf{V})$ is an *argument* for relation $\mathcal{R}$ if for all non-uniform probabilistic polynomial-time stateful adversaries $\mathcal{A}$,

Completeness: $\Pr[\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{KGen}(\mathsf{p}); (\mathtt{inp}, \mathtt{wit}) \leftarrow \mathcal{A}(\mathsf{crs}) : (\mathtt{inp}, \mathtt{wit}) \notin \mathcal{R}_{\mathsf{crs}} \vee \langle \mathsf{P}(\mathsf{crs}, \mathtt{inp}, \mathtt{wit}), \mathsf{V}(\mathsf{crs}, \mathtt{inp}) \rangle = \mathsf{acc}] \approx_c 1$.

Soundness: $\Pr[\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{KGen}(\mathsf{p}); \mathtt{inp} \leftarrow \mathcal{A}(\mathsf{crs}) : \mathtt{inp} \notin \mathcal{L}_{\mathcal{R}:\mathsf{crs}} \wedge \langle \mathcal{A}(\mathsf{crs}, \mathtt{inp}), \mathsf{V}(\mathsf{crs}, \mathtt{inp}) \rangle = \mathsf{acc}] \approx_c 0$.

We call $(\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ a *proof* if soundness holds against unbounded adversaries. An argument is *public coin* if the verifier's subsequent messages correspond to

subsequent bit-strings from her random tape. In particular, they do not depend on the prover's messages.

As noted in [21], the standard definition of a proof of knowledge [3] does not work in such a setting since the adversary may have non-zero probability of computing some trapdoor pertaining to the common reference string and use that information in the argument. In this case, it is possible that there exists a prover with 100% probability of making a convincing argument, where we nonetheless cannot extract a witness. We prove security by using witness-extended emulation [26], the CRS-model version of which was defined by Groth, [21]. Intuitively, an argument has witness-extended emulation, if for every prover P there exists an emulator Emul, such that if P makes V accept, then almost always Emul makes V accept *and* outputs P^*'s witness.

Definition 1. *A public-coin argument* $\Pi = (\mathsf{Pgen}, \mathsf{KGen}, \mathsf{P}, \mathsf{V})$ *has* witness-extended emulation *if for all deterministic polynomial-time provers* P^* *there exists an expected polynomial-time emulator* Emul*, such that for all non-uniform probabilistic polynomial-time adversaries* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathrm{sound}}_{\Pi,\mathcal{A}}(\lambda) \approx_c \mathsf{Adv}^{\mathrm{emul}}_{\Pi,\mathcal{A},\mathsf{Emul}}(\lambda)$*, where*

$$\mathsf{Adv}^{\mathrm{sound}}_{\Pi,\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{KGen}(\mathsf{p}); \omega_{\mathsf{P}} \leftarrow\$ R; (\mathtt{inp}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{crs}; \omega_{\mathsf{P}}); \\ \mathtt{tr} \leftarrow \langle \mathsf{P}^*(\mathsf{crs}, \mathtt{inp}; \mathsf{st}), \mathsf{V}(\mathsf{crs}, \mathtt{inp})\rangle : \mathcal{A}(\mathtt{tr}) = \mathsf{acc}\end{array}\right],$$

$$\mathsf{Adv}^{\mathrm{emul}}_{\Pi,\mathcal{A},\mathsf{Emul}}(\lambda) := \Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{KGen}(\mathsf{p}); \omega_{\mathsf{P}} \leftarrow\$ R; (\mathtt{inp}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{crs}; \omega_{\mathsf{P}}); \\ (\mathtt{tr}, \mathtt{wit}) \leftarrow \mathsf{Emul}^{\langle \mathsf{P}^*(\mathsf{crs}, \mathtt{inp}; \mathsf{st}), \mathsf{V}(\mathsf{crs}, \mathtt{inp})\rangle}(\mathsf{crs}, \mathtt{inp}) : \\ \mathcal{A}(\mathtt{tr}) = \mathsf{acc} \wedge (\mathtt{tr}\ \textit{is accepting} \Rightarrow (\mathtt{inp}, \mathtt{wit}) \in \mathcal{R}_{\mathsf{crs}})\end{array}\right].$$

Here, Emul *can rewind the transcript oracle* $\langle \mathsf{P}^*(\mathsf{crs}, \mathtt{inp}; \mathsf{st}), \mathsf{V}(\mathsf{crs}, \mathtt{inp})\rangle$ *to any particular round with the verifier choosing fresh random coins,* R *is the space of the random coins for* $\mathcal{A}$*,* ω_{P} *is the random tape of* $\mathcal{A}$*, and* st *is the state of* P^* *that also contains his random coins (*P *is deterministic in inputs* $(\mathsf{crs}, \mathtt{inp}, \mathsf{st})$*).*

The verifier's randomness is a part of the transcript (we recall that this includes also the random coins used to probabilistically accept the transcript), while P^* is a deterministic function of $(\mathsf{crs}, \mathtt{inp}; \mathsf{st})$. Thus combining $(\mathsf{crs}, \mathtt{inp}; \mathsf{st})$ with the emulated transcript gives us the view of both the prover and the verifier, and at the same time gives us the witness.

Then, we have an argument of knowledge in the sense that the emulator is able to extract the witness whenever P^* makes a convincing argument. Hence, this definition implies soundness.

Honest-Verifier Zero Knowledge (HVZK, [10]). An argument $\Pi = (\mathsf{Pgen}, \mathsf{KGen}, \mathsf{P}, \mathsf{V})$ is *honest-verifier zero knowledge* if there exists a probabilistic polynomial-time simulator Sim, such that for any non-uniform probabilistic

polynomial-time adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{KGen}(\mathsf{p}); (\mathtt{inp}, \mathtt{wit}, \omega_\mathsf{V}) \leftarrow \mathcal{A}(\mathsf{crs}); \\ \mathtt{tr} \leftarrow \langle \mathsf{P}(\mathsf{crs}, \mathtt{inp}, \mathtt{wit}), \mathsf{V}(\mathsf{crs}, \mathtt{inp}; \omega_\mathsf{V})\rangle : (\mathtt{inp}, \mathtt{wit}) \in \mathcal{R}_\mathsf{crs} \wedge \mathcal{A}(\mathtt{tr}) = \mathsf{acc}\end{array}\right]$$
$$\approx_c \Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{crs} \leftarrow \mathsf{KGen}(\mathsf{p}); (\mathtt{inp}, \mathtt{wit}, \omega_\mathsf{V}) \leftarrow \mathcal{A}(\mathsf{crs}); \\ \mathtt{tr} \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathtt{inp}; \omega_\mathsf{V}) : (\mathtt{inp}, \mathtt{wit}) \in \mathcal{R}_\mathsf{crs} \wedge \mathcal{A}(\mathtt{tr}) = \mathsf{acc}\end{array}\right] .$$

That is, given that the verifier is honest, there exists a simulator that can simulate the view of the prover without knowing the witness. To compensate for this, Sim is allowed to create the messages of the transcript out-of-order.

3 Coefficient-Product Argument

In the shuffle argument, we need a *coefficient-product* argument, where the prover P shows that given a CRS $\mathsf{ck} = ([h_1], \ldots, [h_N])$ (ck for the Pedersen commitment scheme), a public commitment $[c_\mathsf{t}] \in \mathbb{G}$ and a public value $\gamma \in \mathbb{Z}_q$, he knows how to open $[c_\mathsf{t}]$ to a vector $\mathbf{t}$, $[c_\mathsf{t}] = \mathsf{Com}_\mathsf{ck}(\mathbf{t}; r_\mathsf{t})$, so that $\prod_{i=1}^N \mathsf{t}_i = \gamma$. Formally, it is an argument for the ck-dependent relation

$$\mathcal{R}_\mathsf{ck}^{cpa} := \left\{ (([c_\mathsf{t}], \gamma), (\mathbf{t}, r_\mathsf{t})) : [c_\mathsf{t}] = \mathsf{Com}_\mathsf{ck}(\mathbf{t}; r_\mathsf{t}) \wedge \textstyle\prod_{i=1}^N \mathsf{t}_i = \gamma \right\} .$$

Next, we outline the coefficient-product argument that is closely based on the coefficient-product argument from [21]. (More precisely, in [21] it was used as a subargument of a shuffle of known contents. See, for example, [22] for another prior implicit use of the following argument.) We give a formulation of this argument as a separate argument of its own worth.

In the coefficient-product argument, P first proves he knows the message in the commitment $[c_\mathsf{t}] = \mathsf{Com}_\mathsf{ck}(\mathbf{t}; r_\mathsf{t})$, by using the following standard Σ-protocol. Here, the verifier's message comes from a set of σ_y elements for some $\sigma_\mathsf{y} \geq 2^\lambda$.

1. The prover samples $\boldsymbol{\tau} \leftarrow\!\!\$\, \mathbb{Z}_q^N$, $\varrho_\mathsf{t} \leftarrow\!\!\$\, \mathbb{Z}_q$, and sends $[c_\tau] \leftarrow \mathsf{Com}_\mathsf{ck}(\boldsymbol{\tau}; \varrho_\mathsf{t})$ to the verifier.
2. The verifier samples $\mathsf{y} \leftarrow\!\!\$\, \{1, \ldots, \sigma_\mathsf{y}\}$ and sends it to the prover.
3. The prover sends
$$\mathbf{t}^* \leftarrow \mathsf{y}\mathbf{t} + \boldsymbol{\tau} \tag{1}$$
and $r_\mathsf{t}^* \leftarrow \mathsf{y} r_\mathsf{t} + \varrho_\mathsf{t}$ to the verifier.
4. The verifier accepts iff $\mathsf{y}[c_\mathsf{t}] + [c_\tau] =^? \mathsf{Com}_\mathsf{ck}(\mathbf{t}^*; r_\mathsf{t}^*)$.

(As always, the prover's third message elements are computed modulo q.)

Note that $\prod_{i=1}^N \mathsf{t}_i^* = \prod_{i=1}^N (\mathsf{y}\mathsf{t}_i + \tau_i) = \mathsf{y}^N \prod_{i=1}^N \mathsf{t}_i + p(\mathsf{y})$, where $p(Y)$ is some degree $\leq N-1$ polynomial. To finish up the coefficient-vector argument, the prover now only has to demonstrate the knowledge of $p(\mathsf{y})$, for some degree $\leq N-1$ polynomial $p(Y)$, such that $\prod_{i=1}^N \mathsf{t}_i^* - p(\mathsf{y}) = \mathsf{y}^N \gamma$.

For this, using a technique from [21], the prover does the following. Let

$$Q_i \leftarrow \mathsf{y} \prod_{j=1}^{i} \mathsf{t}_j + \Delta_i \ , \tag{2}$$

and Δ_i were chosen before y was sent. (Thus, Δ_i does not depend on y and Q_i is a linear polynomial on y.) In particular, set $Q_1 \leftarrow \mathsf{t}_1^* = \mathsf{y}\mathsf{t}_1 + \tau_1$; thus, $\Delta_1 = \tau_1$.

Also, choose $\Delta_N \leftarrow 0$, thus

$$Q_N = \mathsf{y} \prod_{i=1}^{N} \mathsf{t}_i = \mathsf{y}\gamma \ , \tag{3}$$

which can be tested by the verifier. (The verifier can recompute Q_N, as we will see shortly.) The prover samples the rest of $\Delta_i \leftarrow\!\!\$\, \mathbb{Z}_q$ randomly. Now,

$$\begin{aligned}
\mathsf{y}(Q_{i+1} - \Delta_{i+1}) \overset{(2)}{=} & \mathsf{y}\mathsf{t}_{i+1}(Q_i - \Delta_i) \overset{(1)}{=} \mathsf{t}_{i+1}^* Q_i - \tau_{i+1} Q_i - \mathsf{y}\mathsf{t}_{i+1}\Delta_i \\
\overset{(2)}{=} & \mathsf{t}_{i+1}^* Q_i - \tau_{i+1}\left(\mathsf{y}\prod_{j=1}^{i}\mathsf{t}_j + \Delta_i\right) - \mathsf{y}\mathsf{t}_{i+1}\Delta_i \\
= & \mathsf{t}_{i+1}^* Q_i - \mathsf{y}\left(\mathsf{t}_{i+1}\Delta_i + \tau_{i+1}\prod_{j=1}^{i}\mathsf{t}_j\right) - \tau_{i+1}\Delta_i \ .
\end{aligned}$$

Define $\mathsf{b}_i \leftarrow \Delta_{i+1} - \mathsf{t}_{i+1}\Delta_i - \tau_{i+1}\prod_{j=1}^{i}\mathsf{t}_j$, $\beta_i \leftarrow -\tau_{i+1}\Delta_i$, and $\mathsf{b}_i^* \leftarrow \mathsf{y}\mathsf{b}_i + \beta_i$ as on Fig. 1. Thus, for $i = 1, \ldots, N-1$,

$$\mathsf{y}Q_{i+1} = \mathsf{t}_{i+1}^* Q_i + \mathsf{y}\mathsf{b}_i + \beta_i = \mathsf{t}_{i+1}^* Q_i + \mathsf{b}_i^* \ . \tag{4}$$

The verifier can recompute Q_i by using the definition of Q_1, and Eq. (4), see Fig. 1.

Since b_i^* is linear in y,

$$\begin{aligned}
\mathsf{y}^N\gamma \overset{(3)}{=} & \mathsf{y}^{N-1}Q_N \overset{(4)}{=} \mathsf{y}^{N-2}(\mathsf{t}_N^* Q_{N-1} + \mathsf{b}_{N-1}^*) \overset{(4)}{=} \ldots \overset{(4)}{=} \prod_{i=1}^{N}\mathsf{t}_i^* + p(\mathsf{y}) \\
= & \mathsf{y}^N \prod_{i=1}^{N}\mathsf{t}_i + p'(\mathsf{y}) \ ,
\end{aligned} \tag{5}$$

where $p(X)$ and $p'(X)$ are degree $\leq N-1$ polynomials. This is equivalent to $\mathsf{y}^N(\gamma - \prod_{i=1}^{N}\mathsf{t}_i) + p'(\mathsf{y}) = 0$. As y was chosen randomly, due to Schwartz-Zippel lemma Eq. (6), with overwhelming probability this implies that $\mathsf{y}^N(\gamma - \prod_{i=1}^{N}\mathsf{t}_i) + p'(\mathsf{y})$ is a zero polynomial and thus $\gamma = \prod_{i=1}^{N}\mathsf{t}_i$. Hence, we are done.

Construction. The full coefficient-product argument Π_{cpa} is depicted by Fig. 1. Note that the verifier chooses $\mathsf{y} \neq 0$ to avoid division by 0 on the penultimate line of Fig. 1.

Security. We prove that this argument is perfectly complete, has witness-extended emulation, and is perfectly special honest-verifier zero knowledge. We state the security of Π_{cpa} in Theorem 1. Π_{cpa} is essentially the same as a sub-argument in [21]; it only includes batch verification as an additional step of optimization, and moreover, [21] did not formalize it as a separate argument.

CRS: $\mathsf{crs} = \mathsf{ck} = ([h_1], \ldots, [h_N])$ as in the extended Pedersen.
Common inputs: $\mathtt{inp} = ([c_\mathsf{t}] = \mathsf{Com}_\mathsf{ck}(\mathbf{t}; r_\mathsf{t}), \gamma)$.
Witness: $\mathtt{wit} = (\mathbf{t}, r_\mathsf{t})$.

1. The prover $\mathsf{P}(\mathsf{crs}, \mathtt{inp}, \mathtt{wit})$ does:
   (a) Sample $\boldsymbol{\tau} \leftarrow\$ \mathbb{Z}_q^N$.
   (b) Let $\Delta_1 \leftarrow \tau_1$ and $\Delta_N \leftarrow 0$.
   (c) For $i = 2, \ldots, N-1$: sample $\Delta_i \leftarrow\$ \mathbb{Z}_q$.
   (d) $X_0 \leftarrow 1$; For $i = 1, \ldots, N-1$:
   – Set $X_i \leftarrow X_{i-1}\mathsf{t}_i$, $\mathsf{b}_i \leftarrow \Delta_{i+1} - \mathsf{t}_{i+1}\Delta_i - \tau_{i+1}X_i$, $\beta_i \leftarrow -\tau_{i+1}\Delta_i$.
   (e) Denote $\mathbf{b} = (\mathsf{b}_1, \ldots, \mathsf{b}_{N-1})^\top$, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{N-1})^\top$.
   (f) Sample $\varrho_\mathsf{t}, r_\mathsf{b}, \varrho_b \leftarrow\$ \mathbb{Z}_q$.
   (g) $[c_\tau] \leftarrow \boxed{\mathsf{Com}_\mathsf{ck}(\boldsymbol{\tau}; \varrho_\mathsf{t})}$; $[c_b] \leftarrow \boxed{\mathsf{Com}_\mathsf{ck}(\mathbf{b} \,/\!/\, 0; r_\mathsf{b})}$;
   (h) $[c_\beta] \leftarrow \boxed{\mathsf{Com}_\mathsf{ck}(\boldsymbol{\beta} \,/\!/\, 0; \varrho_b)}$;
   Send $[c_\tau, c_b, c_\beta]$ to the verifier.
2. The verifier samples $\mathsf{y} \leftarrow\$ \{1, \ldots, \sigma_\mathsf{y}\}$, and sends y to the prover.
3. Prover does:
   (a) Set $\mathbf{t}^* \leftarrow \mathsf{y}\mathbf{t} + \boldsymbol{\tau}$, $r_\mathsf{t}^* \leftarrow \mathsf{y}r_\mathsf{t} + \varrho_\mathsf{t}$, $\mathbf{b}^* \leftarrow \mathsf{y}\mathbf{b} + \boldsymbol{\beta} \in \mathbb{Z}_q^{N-1}$, $r_\mathsf{b}^* \leftarrow \mathsf{y}r_\mathsf{b} + \varrho_b$.
   (b) Send $(\mathbf{t}^*, r_\mathsf{t}^*, \mathbf{b}^*, r_\mathsf{b}^*)$ to the verifier.
4. The verifier samples $\mathsf{z} \leftarrow\$ \{0, \ldots, \sigma_\mathsf{z} - 1\}$. He checks that
$$\mathsf{y}[c_\mathsf{t}] + [c_\tau] + \mathsf{z}(\mathsf{y}[c_b] + [c_\beta]) =^? \boxed{\mathsf{Com}_\mathsf{ck}(\mathbf{t}^* + \mathsf{z}(\mathbf{b}^* \,/\!/\, 0); r_\mathsf{t}^* + \mathsf{z}r_\mathsf{b}^*)} \quad . \tag{6}$$
   Set $Q_1 \leftarrow \mathsf{t}_1^*$, $Q_2 \leftarrow (\mathsf{t}_2^* Q_1 + \mathsf{b}_1^*)/\mathsf{y}$, $\ldots$, $Q_N \leftarrow (\mathsf{t}_N^* Q_{N-1} + \mathsf{b}_{N-1}^*)/\mathsf{y}$. Check that $Q_N =^? \mathsf{y}\gamma$.
   The verifier outputs $(\mathsf{acc}, \mathsf{z})$ iff both checks succeed, and $(\mathsf{rej}, \mathsf{z})$ otherwise.

Fig. 1: The coefficient-product argument. Dotted formulas correspond to expensive (that is, $\Omega(N)$) computations.

First, note that if either of the following equations does not hold,

$$\mathsf{y}[c_\mathsf{t}] + [c_\tau] =^? \mathsf{Com}_\mathsf{ck}(\mathbf{t}^*; r_\mathsf{t}^*) \;\; , \mathsf{y}[c_b] + [c_\beta] =^? \mathsf{Com}_\mathsf{ck}(\mathbf{b}^* \,/\!/\, 0; r_\mathsf{b}^*) \;\; , \tag{7}$$

then according to Schwartz-Zippel lemma, Eq. (6) holds for a random z at most with probability $1/\sigma_\mathsf{z}$. To simplify the proofs of both following theorem, we define another coefficient-product argument Π'_{cpa} that differs from Π_{cpa} only by removing the batch verification: namely, the verifier checks Eq. (7) instead of Eq. (6). Clearly, if an adversary succeeds with probability ε against Π_{cpa}, then it succeeds with probability not larger than ε', $\varepsilon \geq \varepsilon' \geq \varepsilon - 1/\sigma_\mathsf{z}$, against Π'_{cpa}.

Fix $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$ and $\mathsf{crs} \leftarrow \mathsf{KGen}(\mathsf{p})$. Let us denote with ω_P the random coins of the adversary and with ω_V the random coins of the verifier in Π'_{cpa}. Since Π'_{cpa} is a public-coin protocol, each ω_V corresponds to a different value verifier's challenge y. Let $\boldsymbol{V} = \boldsymbol{V}_\mathsf{crs}$ be a crs-dependent matrix with an entry $\boldsymbol{V}_{\omega_\mathsf{P}, \omega_\mathsf{V}} =$

$\boldsymbol{V}(\omega_{\mathsf{P}}, \omega_{\mathsf{V}}) = 1$ if for $(\mathtt{inp}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{crs}; \omega_{\mathsf{P}})$, $\langle \mathsf{P}^*(\mathsf{crs}, \mathtt{inp}; \mathsf{st}), \mathsf{V}(\mathsf{crs}, \mathtt{inp}; \omega_{\mathsf{V}}) \rangle = \mathsf{acc}$ and $\boldsymbol{V}(\omega_{\mathsf{P}}, \omega_{\mathsf{V}}) = 0$ otherwise.

Theorem 1. *Π_{cpa} is a three-message public-coin argument for $[c_{\mathsf{t}}]$ being a commitment to a message $\mathbf{t}$ such that $\prod_{i=1}^{N} \mathsf{t}_i = \gamma$. It is perfectly complete and perfectly HVZK. It has witness-extended emulation, assuming the commitment scheme is binding.*

Proof. Completeness: obvious.

Witness-extended emulation: Fix $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$ and $\mathsf{crs} \leftarrow \mathsf{KGen}(\mathsf{p})$. Let V' be the verifier of Π'_{cpa}. Let P^* be a prover that makes V' to accept a false statement with some non-negligible probability $\varepsilon'_{\mathsf{crs}}$.

To extract a witness, Emul rewinds a runs $\langle \mathsf{P}^*, \mathsf{V}' \rangle$ on the same challenge x until it gets another acceptable argument. Let $tr_j = (\mathtt{inp}; [c_\tau, c_b, c_\beta]; \mathsf{y}_j; \mathbf{t}^*_j, r^*_{\mathsf{t}:j}, \mathbf{b}^*_j, r^*_{b:j})$, $j \in \{1, 2\}$, be the two acceptable arguments. Emul uses the last two transcripts to open the commitments. Since $\mathsf{y}_1 \neq \mathsf{y}_2$ and Eq. (7) holds, from Eq. (7) (left) it follows that $[c_{\mathsf{t}}] = \mathsf{Com}_{\mathsf{ck}}(\mathbf{t}; r_{\mathsf{t}})$, where $\mathbf{t} \leftarrow (\mathbf{t}^*_1 - \mathbf{t}^*_2)/(\mathsf{y}_1 - \mathsf{y}_2)$ and $r_{\mathsf{t}} \leftarrow (r^*_{\mathsf{t}:1} - r^*_{\mathsf{t}:2})/(\mathsf{y}_1 - \mathsf{y}_2)$. Thus, Emul has succeeded in extracting an opening of $[c_{\mathsf{t}}]$.

Following Theorem 1 in [21], we can argue that Emul runs in expected polynomial time. If we are in a situation where P^* can make the verifier to accept with probability $\varepsilon > 0$ on challenge x, then the expected number of rewindings to get an acceptable transcript is $1/\varepsilon$. If P^* fails then we do not have to rewind at all, and thus the number of expected queries to $\langle \mathsf{P}^*, \mathsf{V}' \rangle$ is 2. Since Emul does an expected polynomial number of queries, there is only negligible probability of ending in a run where $\mathsf{y} = \mathsf{y}'$ or some other unlikely event (e.g., breaking the binding of the commitment scheme) occurs. Hence, with an overwhelming probability, either P^* did not succeed or Emul succeeded in extraction.

Next, we argue that the probability for extracting an opening of $[c_{\mathsf{t}}]$, such that $\prod_{i=1}^{N} \mathsf{t}_i \neq \gamma$, is negligible. Assume that P^* has a non-negligible success probability $1/f(\lambda)$, for a polynomial $f(X)$, to produce an acceptable argument. We now run P^* and rewind to get three random challenges $\mathsf{y}_1, \mathsf{y}_2, \mathsf{y}_3$. With probability at least $1/f(\lambda)^3$, P^* succeeds in creating accepting arguments for all three challenges. Since $\mathsf{y}_1 \neq \mathsf{y}_2$, with an overwhelming probability, and Eq. (7) holds, Emul can open the following commitments from the first two transcripts (with an overwhelming probability).

(1) From Eq. (7) (left) it follows that $[c_{\mathsf{t}}] = \mathsf{Com}_{\mathsf{ck}}(\mathbf{t}; r_{\mathsf{t}})$, where $\mathbf{t} \leftarrow (\mathbf{t}^*_1 - \mathbf{t}^*_2)/(\mathsf{y}_1 - \mathsf{y}_2)$; $r_{\mathsf{t}} \leftarrow (r^*_{\mathsf{t}:1} - r^*_{\mathsf{t}:2})/(\mathsf{y}_1 - \mathsf{y}_2)$.
(2) Since it knows $\mathbf{t}^*_1$, $\mathbf{t}$, $r^*_{\mathsf{t}:1}$ and r_{t}, Emul can compute $\boldsymbol{\tau} \leftarrow \mathbf{t}^*_1 - \mathsf{y}_1\mathbf{t}$; $\varrho_{\mathsf{t}} \leftarrow r^*_{\mathsf{t}:1} - \mathsf{y}_1 r_{\mathsf{t}}$; thus $[c_\tau] = \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{\tau}; \varrho_{\mathsf{t}})$.
(3) From Eq. (7) (right) it follows that $[c_b] = \mathsf{Com}_{\mathsf{ck}}(\mathbf{b} \,/\!/\, 0; r_{\mathsf{b}})$, where $\mathbf{b} \leftarrow (\mathbf{b}^*_1 - \mathbf{b}^*_2)/(\mathsf{y}_1 - \mathsf{y}_2)$; $r_{\mathsf{b}} \leftarrow (r^*_{\mathsf{b}:1} - r^*_{\mathsf{b}:2})/(\mathsf{y}_1 - \mathsf{y}_2)$.
(4) Since it knows $\mathbf{b}^*_1$, $\mathbf{b}$, $r^*_{\mathsf{b}:1}$ and r_{b}, Emul can compute $\boldsymbol{\beta} \leftarrow \mathbf{b}^*_1 - \mathsf{y}_1\mathbf{b}$; $\varrho_b \leftarrow r^*_{\mathsf{b}:1} - \mathsf{y}_1 r_{\mathsf{b}}$; thus $[c_\beta] = \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{\beta} \,/\!/\, 0; \varrho_b)$.

Sample $\mathbf{t}^* \leftarrow\$ \mathbb{Z}_q^N$, $r_{\mathsf{t}}^* \leftarrow\$ \mathbb{Z}_q$;
Set $[c_\tau] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{t}^*; r_{\mathsf{t}}^*) - \mathsf{y}[c_{\mathsf{t}}]$;
Set $Q_1 \leftarrow \mathsf{t}_1^*$, $Q_i \leftarrow\$ \mathbb{Z}_q$ for $i \in \{2, \ldots, N-1\}$, and $Q_N \leftarrow \mathsf{y}\gamma$;
Sample $r_{\mathsf{b}} \leftarrow\$ \mathbb{Z}_q$;
1 Set $[c_b] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{0}_N; r_{\mathsf{b}})$;
Set $\mathsf{b}_1^* \leftarrow \mathsf{y}Q_2 - \mathsf{t}_2^* Q_1$, $\ldots$, $\mathsf{b}_{N-1}^* \leftarrow \mathsf{y}Q_N - \mathsf{t}_N^* Q_{N-1}$;
Sample $r_{\mathsf{b}}^* \leftarrow\$ \mathbb{Z}_q$;
Set $[c_\beta] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{b}^* \,//\, 0; r_{\mathsf{b}}^*) - \mathsf{y}[c_b]$;
Return $([c_\tau, c_b, c_\beta]; \mathsf{y}; \mathbf{t}^*, r_{\mathsf{t}}^*, \mathbf{b}^*, r_{\mathsf{b}}^*; \mathsf{acc}, \mathsf{z})$;

Fig. 2: Simulator $\mathsf{Sim}(\mathsf{ck}, \mathtt{inp}, (\mathsf{y}, \mathsf{z}))$ of the coefficient-product argument

Thus, Emul has extracted $\mathbf{t}$, r_{t}, $\boldsymbol{\tau}$, ϱ_{t}, $\mathbf{b}$, r_{b}, $\boldsymbol{\beta}$, ϱ_b, and thus also $\mathbf{t}^* \leftarrow \mathsf{y}\mathbf{t} + \boldsymbol{\tau}$ and $\mathbf{b}^* \leftarrow \mathsf{y}\mathbf{b} + \boldsymbol{\beta}$.

Consider now the third transcript with y_3. Since $Q_N = \mathsf{y}\gamma$, we obtain from Eq. (5) that $P(\mathsf{y}_3) := \mathsf{y}_3^N(\gamma - \prod_{i=1}^N \mathsf{t}_i) - p'(\mathsf{y}_3) = 0$, where $p'(Y)$ is some degree $\leq N-1$ polynomial. Since the emulator knows $\mathbf{t}^*$ and $\mathbf{b}^*$, it knows $P(Y)$. Since a non-zero degree-($\leq N$) polynomial $P(Y)$ has $\leq N$ roots, then the probability that $P(\mathsf{y}_3) = 0$ and $P(Y) \neq 0$ is at most N/σ_{y}. Since $P(Y) = 0$ implies that $\gamma = \prod_{i=1}^N \mathsf{t}_i$, Emul has retrieved a witness $\mathtt{wit} = (\mathbf{t}, r_{\mathsf{t}})$, such that $[c_{\mathsf{t}}] = \mathsf{Com}_{\mathsf{ck}}(\mathbf{t}; r_{\mathsf{t}})$ and $\gamma = \prod_{i=1}^N \mathsf{t}_i$, with an overwhelming probability. Thus, this argument has witness-extended emulation.

Perfect SHVZK: We construct the following simulator Sim. (This part of the proof follows [21] quite closely.) The simulator is depicted in Fig. 2. Since Sim's output does not depend on $\mathbf{t}$ or r_{t}, it reveals no information about the witness. Clearly, Sim's output will be accepted by the verifier.

We use the same idea as [21] to show that this simulator provides output from the correct distribution. First, consider the following simulator $\mathsf{Sim}_{\mathsf{t}}$ that otherwise outputs the same thing as Sim, except that it uses the knowledge of $\mathbf{t}$ and r_{b} to construct $[c_b]$ as $[c_b] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{b} \,//\, 0; r_{\mathsf{b}})$. More precisely, it works as Sim, except that Line 1 in Fig. 2 is replaced with the following three steps:

for $i \in [1 \,..\, N]$ **do** $\Delta_i \leftarrow Q_i - \mathsf{y} \prod_{j=1}^i \mathsf{t}_i$
for $i \in [1 \,..\, N-1]$ **do** $\mathsf{b}_i \leftarrow \Delta_{i+1} - \mathsf{t}_{i+1}\Delta_i - \tau_{i+1} \prod_{j=1}^i \mathsf{t}_j$
$[c_b] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{b} \,//\, 0; r_{\mathsf{b}})$;

Clearly, the output of $\mathsf{Sim}_{\mathsf{t}}$ comes from the same distribution as the output of Sim, but it has constructed $[c_b]$ as needed due to the knowledge of $\mathbf{t}$ and r_{t}.

Finally, clearly $\mathsf{Sim}_{\mathsf{t}}$ chooses the values from the same distribution as the real prover (given that y and z are chosen randomly), but in a different order. □

Complexity. Clearly, the prover's computational complexity is dominated by three $\approx N$-wide multi-exponentiations, while the verifier's computational complexity is dominated by one $\approx N$-wide multi-exponentiation. The communication complexity is dominated by $\approx 2N$ elements of $\mathbb{Z}_q$.

4 A Characterization of Permutation Matrices

Next, we prove the following theorem that generalizes a result from [32]. Namely, [32] let the verifier to choose N random values $X_1, \ldots, X_N$. To reduce communication (by avoiding sending all N variables to the prover), they used a pseudo-random number generator to obtain N values X_i out of a short seed X_0. Our result shows that the pseudo-random number generator can be replaced by a what we call *PM-evidential* family of multivariate polynomials. On top of the efficiency aspect, we obtain a novel mathematical characterization of permutation matrices of independent interest.

Definition 2 (PM-evidential). *Let $\mathbb{F}$ be a field. A family of N-degree ν-variate polynomials $\{\psi_i(\boldsymbol{X})\}_{i=1}^{N}$, $\psi_i \in \mathbb{F}[X_1, \ldots, X_\nu]$, is* PM-evidential over $\mathbb{F}$, *if for $\Psi(\boldsymbol{X}) := \prod_{i=1}^{N} \psi_i(\boldsymbol{X})$,*

(i) $\psi_i(\boldsymbol{X}) \neq 0$ for each i,
(ii) for each $\boldsymbol{a} \in \mathbb{F}^n$ with $wt(\boldsymbol{a}) > 1$, $\left(\sum_{i=1}^{N} a_i \psi_i(\boldsymbol{X})\right) \nmid \Psi(\boldsymbol{X})$, and
(iii) $(\psi_i(\boldsymbol{X}))^2 \nmid \Psi(\boldsymbol{X})$ for each i.

One can use PM-evidential polynomials to efficiently check whether a matrix is a permutation matrix, as explained by the following result.

Lemma 2. *Let $\mathbb{F}$ be a field. Let $\boldsymbol{M}$ be an $N \times N$ matrix over $\mathbb{F}$, let $\boldsymbol{X} = (X_1, \ldots, X_\nu)$ be a vector of $\nu \leq N$ indeterminates, and let $\{\psi_i(\boldsymbol{X})\}_{i=1}^{N}$ be PM-evidential over $\mathbb{F}$. Let $\boldsymbol{\psi}(\boldsymbol{X}) := (\psi_1(\boldsymbol{X}), \ldots, \psi_N(\boldsymbol{X}))^\top$, $\Psi_{\boldsymbol{M}}(\boldsymbol{X}) := \prod_{i=1}^{N} \langle \boldsymbol{M}_i^\top, \boldsymbol{\psi}(\boldsymbol{X}) \rangle$, and $\Psi(\boldsymbol{X}) := \prod_{i=1}^{N} \psi_i(\boldsymbol{X})$ in $\mathbb{F}[\boldsymbol{X}]$. Then $\boldsymbol{M}$ is a permutation matrix iff $\Psi_{\boldsymbol{M}}(\boldsymbol{X}) = \Psi(\boldsymbol{X})$ and $\boldsymbol{M} \cdot \boldsymbol{1}_N = \boldsymbol{1}_N$.*

Proof. **($\Rightarrow$)** Assume $\boldsymbol{M}$ is a permutation matrix. Then clearly, $\Psi_{\boldsymbol{M}}(\boldsymbol{X}) = \Psi(\boldsymbol{X})$ (since then $\Psi_{\boldsymbol{M}}(\boldsymbol{X})$ is a product of ψ_i-s in a permuted order) and $\boldsymbol{M} \cdot \boldsymbol{1}_N = \boldsymbol{1}_N$.
($\Leftarrow$) Assume that $\Psi_{\boldsymbol{M}}(\boldsymbol{X}) = \Psi(\boldsymbol{X})$. Consider the following three cases. First, if some row $\boldsymbol{M}_i$ is a zero vector, then $\Psi_{\boldsymbol{M}}$ is a zero polynomial, and thus $\Psi(\boldsymbol{X}) = 0$, a contradiction to Item i in Definition 2. Second, if the ith row $\boldsymbol{M}_i$ contains more than one non-zero element, then $(\sum M_{ij}\psi_j(\boldsymbol{X})) \mid \Psi_{\boldsymbol{M}}(\boldsymbol{X})$, where $wt(\boldsymbol{M}_i) > 1$. A contradiction with Item ii in Definition 2. Third, if the jth column $\boldsymbol{M}^{(j)}$ contains more than one non-zero element, then — since each row contains exactly one non-zero element — $\psi_j^2(\boldsymbol{X}) \mid \Psi_{\boldsymbol{M}}(\boldsymbol{X})$, contradicting Item iii in Definition 2).

Hence, each row and column of $\boldsymbol{M}$ have exactly one non-zero element. Finally, since $\boldsymbol{M} \cdot \boldsymbol{1}_N = \boldsymbol{1}_N$, the non-zero elements of $\boldsymbol{M}$ must equal one. □

Intuitively, Terelius and Wikström proved that the family $\{\psi_i(\boldsymbol{X}) = X_i\}$ is PM-evidential. Next, we construct a simple family of PM-evidential polynomials, where the number of indeterminates is just two. More precisely, we show that the family of polynomials $\{\psi_k(X, Y) = X^k + Y\}_{k=0}^{N}$ in $\mathbb{F}[X, Y]$ is PM-evidential.

Lemma 3. *The family of polynomials $\{\psi_k\}_{k=1}^{N}$, where $\psi_k(X, Y) = X^{k-1} + Y$, is PM-evidential.*

To prove Lemma 3, we need to show that the polynomials $X^k + Y$ are irreducible. For the latter, we will use the well-known Eisenstein's criterion.

Proposition 1 (Eisenstein's Criterion [24]). *Let $\mathbb{V}$ be a unique factorization domain. Let $p(X) = \sum_{i=0}^{k} a_i X^i \in \mathbb{V}[X]$. Then $p(X)$ is irreducible in $\mathbb{V}[X]$ if there exists an irreducible element $s \in \mathbb{V}$ such that the three conditions hold:*

(i) $s \mid a_i$ *for every* $i \in [0\,..\,k-1]$,
(ii) $s \nmid a_k$,
(iii) $s^2 \nmid a_0$

Proof (Proof of Lemma 3). Item i of Definition 2 holds trivially.

To show that Items ii and iii hold, we first use Eisenstein's criterion to show $X^k + Y$ is irreducible, for $k \in [0\,..\,N-1]$. Think of $X^k + Y$ as a polynomial in $\mathbb{V}[X]$, where $\mathbb{V} = \mathbb{F}[Y]$. Then, $a_0 = Y$, $a_i = 0$ for $i \in [1\,..\,k-1]$ and $a_k = 1$. Taking $s = Y$, it is easy to see that the three conditions of the Eisenstein's criterion are satisfied. Thus, $X^k + Y$ is irreducible.

To see that Item ii holds, suppose that

$$\textstyle\sum_{i=1}^{N-1} a_i \psi_i(X,Y) = L(X,Y) \ , \tag{8}$$

where $L(X,Y) \mid \Psi(X,Y)$. Since $\mathbb{F}[X,Y]$ is a unique factorization domain and ψ_k are all irreducible, we get that $L(X,Y)$ is a product of some of ψ_i. Since on the left hand side of Eq. (8), the degree of Y is 1, then also also the degree of Y on the right hand is also 1. Thus, $L(X,Y) = \psi_j(X,Y)$ for some j, and hence, $\sum_{i=1}^{N-1} a_i \psi_i(X,Y) = \psi_j(X,Y)$. Because they have distinct degrees, $\psi_i(X,Y)$ are linearly independent. Thus we get that $a_i = 0$ if $i \neq j$ and $a_j = 1$. Thus, $wt(\boldsymbol{a}) = 1$ and Item ii holds.

Finally, Item iii follows from the fact that $\psi_i(X,Y)$ are irreducible and distinct. □

While we believe that the proposed solution is close to optimal, we leave it as an open question of whether some other families give even better communication and computational complexity for the final shuffle argument.

Additionally, in the proof of the new shuffle argument of Section 5, we will need to invert a certain matrix $(\psi_i(x_j))_{i,j}$ obtained by rewinding the argument. We need to show that the probability that this matrix is invertible is overwhelming. For that analysis, we give the following definition.

Definition 3. *Define*

$$\mathsf{ni}(\boldsymbol{\psi}, \sigma_{\mathsf{x}}, q) := \max_{\boldsymbol{x_0},\ldots,\boldsymbol{x_{N-2}}} \left\{ \Pr_{\boldsymbol{x_{N-1}}} \left[(\psi_i(\boldsymbol{x_j}))_{i,j} \text{ is not invertible} \right] \mid \begin{array}{c} x_{0,i}, \ldots, x_{N-1,i} \text{ are} \\ \text{pairwise different} \\ \text{for every } i \end{array} \right\} .$$

Here, we assume that $\boldsymbol{x_0}, \ldots, \boldsymbol{x_{N-1}} \in \{0, \ldots, \sigma_{\mathsf{x}} - 1\}^{\nu}$.

For good values in the proof, we want to make this value as small as possible.

Lemma 4. *Let $x_1, \ldots, x_N$ be distinct random elements of $\{0, \ldots, \sigma_{\mathsf{x}} - 1\}$, and let $y_1, \ldots, y_N$ be distinct random elements of $\{0, \ldots, \sigma_{\mathsf{x}} - 1\}$. Let $\boldsymbol{M}$ be the matrix with elements $(x_i^{j-1} + y_i)_{i,j=1}^N$ in $\mathbb{Z}_q$. Then, $\det(\boldsymbol{M}) = 0$ with probability at most $1/q$. Additionally, for fixed distinct $x_1, \ldots, x_N$ and fixed distinct $y_1, \ldots, y_{N-1}$, there exists at most one y_N, such that $\boldsymbol{M}$ is not invertible.*

Proof. Let the determinant of $\boldsymbol{M}$ be D. Denote $\boldsymbol{x} = (x_1, \ldots, x_N)$ and $S := [1\,..\,N]$. For a subset $S' \subseteq S$, let $\boldsymbol{M}_S$ be the matrix, where the ith row is $(x_i^{j-1})_{j=1}^N$, if $i \in S'$, and $(y_i)_{j=1}^N$, if $i \notin S'$. Since $\det\left(\begin{smallmatrix} \boldsymbol{A} \\ \boldsymbol{b}+\boldsymbol{c} \\ \boldsymbol{D} \end{smallmatrix}\right) = \det\left(\begin{smallmatrix} \boldsymbol{A} \\ \boldsymbol{b} \\ \boldsymbol{D} \end{smallmatrix}\right) + \det\left(\begin{smallmatrix} \boldsymbol{A} \\ \boldsymbol{c} \\ \boldsymbol{D} \end{smallmatrix}\right)$, we get by induction that $D = \sum_{S' \subseteq S} \det(\boldsymbol{M}_{S'})$.

Moreover, if $|S'| < N-1$, then $\det(\boldsymbol{M}_{S'}) = 0$. Really, in this case, there exist at least two rows i and j, where the elements are just y_i and y_j. The determinant of every 2×2-submatrix from these two rows is 0. Thus, by the cofactor expansion of a determinant, $\det(\boldsymbol{M}_{S'}) = 0$. Thus, $D = \det(\boldsymbol{M}_S) + \sum_{i=1}^N \det(\boldsymbol{M}_{S\setminus\{i\}})$. Now, $\boldsymbol{M}_S$ is a Vandermonde matrix and thus $\det(\boldsymbol{M}_S) = \prod_{1 \le i < j \le N} (x_j - x_i) \neq 0$.

On the other hand, observe that $\det(\boldsymbol{M}_{S\setminus\{i\}})$ is equal to y_i times $P_i(\boldsymbol{x})$ for some polynomial P_i. There are two possible cases. Either $P_i(\boldsymbol{x}) = 0$, for all i, or at least one $P_i(\boldsymbol{x})$ is nonzero. In the first case, $D = \det(\boldsymbol{M}_S) \neq 0$. Note that then, there exists no y_I such that $\det(\boldsymbol{M}_S) = 0$. In the second case, say $P_I(\boldsymbol{x}) \neq 0$ holds for a concrete I. Then, $D = 0$ iff $y_I = -(\det(\boldsymbol{M}_S) + \sum_{i \neq I} y_i P_i(\boldsymbol{x}))/P_I(\boldsymbol{x})$, which is true for precisely one value of y_I.

Thus, presuming that $x_1, \ldots, x_N$ are pairwise different and taking the probability over the choice y_N,

$$\Pr[D = 0] = 0 \cdot \Pr[\forall i \in S, P_i(\boldsymbol{x}) = 0] + \tfrac{1}{q} \cdot (1 - \Pr[\forall i \in S, P_i(\boldsymbol{x}) = 0]) \le \tfrac{1}{q} \ .$$

This proves the claim. □

5 Shuffle Argument

Let N be the number of shuffled ciphertexts, let $[\boldsymbol{w}]$ and $[\widehat{\boldsymbol{w}}]$ be two tuples of the ciphertexts. Assume that pk is the public key of an additively homomorphic (in our case, the lifted Elgamal [11]) cryptosystem. In a *shuffle argument*, the prover aims to convince the verifier that, for a fixed pk, he knows a permutation $\pi \in S_N$ and a vector of randomizers $\boldsymbol{s} \in \mathbb{Z}_q^N$, such that $[\widehat{\boldsymbol{w}}_i] = [\boldsymbol{w}_{\pi^{-1}(i)}] + \mathsf{Enc}_{\mathsf{pk}}(0; s_{\pi^{-1}(i)})$. Formally, it is an argument for the relation

$$\mathcal{R}_{\mathsf{pk}} := \left\{ \begin{array}{l} (([\boldsymbol{w}], [\widehat{\boldsymbol{w}}]), (\pi, \boldsymbol{s}) \in S_N \times \mathbb{Z}_q^N) : \\ \forall i \in [1\,..\,N], [\widehat{\boldsymbol{w}}_i] = [\boldsymbol{w}_{\pi^{-1}(i)}] + \mathsf{Enc}_{\mathsf{pk}}(0; s_{\pi^{-1}(i)}) \end{array} \right\} .$$

Before going on, we recall that if for some matrix $\boldsymbol{M} \in \mathbb{Z}_q^{N \times N}$, $[u_i] = \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{M}^{(i)}; \hat{r}_i)$, then for any $\mathbf{t} \in \mathbb{Z}_q^N$,

$$\langle [\boldsymbol{u}], \mathbf{t} \rangle = \sum_{i=1}^{N} \mathsf{t}_i [u_i] = \mathsf{Com}_{\mathsf{ck}} \left(\sum_{i=1}^{N} \boldsymbol{M}^{(i)} \mathsf{t}_i ; \sum_{i=1}^{N} \hat{r}_i \mathsf{t}_i \right) = \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{M}\mathbf{t}; \langle \hat{\boldsymbol{r}}, \mathbf{t} \rangle) \ . \quad (9)$$

Next, we will give a short explanation of the argument on Fig. 3. The prover P and the verifier V do the following:

(1) P commits to the permutation matrix $\boldsymbol{M}$, where $[u_i]$ is a commitment to $\boldsymbol{M}^{(i)}$ for $i < N$, and $[u_N]$ is homomorphically computed as a commitment to $\mathbf{1}_N - \sum_{i=1}^{N-1} \boldsymbol{M}^{(i)}$. (This guarantees that $\sum_{i=1}^{N} \boldsymbol{M}^{(i)} = \mathbf{1}_N$.)
(2) V chooses $\mathbf{x}$ after P committed to $\boldsymbol{M}$. Let $\mathbf{t} = \boldsymbol{\psi}(\mathbf{x})$. Both parties compute a permuted vector $\hat{\mathbf{t}}$, $\hat{\mathsf{t}}_i = \mathsf{t}_{\pi^{-1}(i)}$, of the vector $\mathbf{t}$.
(3) P proves that he knows how to open $\langle \mathbf{t}, [\boldsymbol{u}] \rangle$ as a commitment of $\boldsymbol{M}\mathbf{t}$.
(4) P proves, by using the coefficient-product argument, that $\prod_{i=1}^{N} \hat{\mathsf{t}}_i = \prod_{i=1}^{N} \mathsf{t}_i$. Hence, the verifier is convinced (via Lemma 2) that $\boldsymbol{M}$ is a permutation matrix. Thus, $\hat{\mathbf{t}}$ is a permutation of $\mathbf{t}$.
The coefficient-product argument is interleaved with the main argument for efficiency reasons. For easier readability, we have added the symbol (🙈) to the lines in Fig. 3 that contain the coefficient-product argument.
(5) Finally, P proves that he used the same matrix $\boldsymbol{M}$ (together with some additional randomness) to form $[\widehat{\boldsymbol{w}}]$ from $[\boldsymbol{w}]$. Since $\boldsymbol{M}$ is a permutation matrix, the shuffle argument is sound (more precisely, has witness-extended emulation).

Construction. The new shuffle argument Π_{sh} is depicted by Fig. 3. Here, we assume $\sigma_{\mathsf{x}}, \sigma_{\mathsf{y}}, \sigma_{\mathsf{z}} \geq 2^{\lambda}$, $\mathbb{G}$ is a group of order q, and N is the number of ciphertexts with $N < 2^{0.5\lambda}$ Fix a family of PM-evidential polynomials $\psi_i \in \mathbb{Z}[X_1, \ldots, X_\nu]$ for some $\nu \leq N$, such that $\mathsf{ni}(\boldsymbol{\psi}, \sigma_{\mathsf{x}}, q)$ is negligible. The common inputs of the prover and the verifier are the ciphertext tuples $[\boldsymbol{w}] = ([\boldsymbol{w}_1], \ldots, [\boldsymbol{w}_N])$ and $[\widehat{\boldsymbol{w}}] = ([\widehat{\boldsymbol{w}}_1], \ldots, [\widehat{\boldsymbol{w}}_N])$, with $[\widehat{\boldsymbol{w}}_i] = [\boldsymbol{w}_{\pi^{-1}(i)}] + \mathsf{Enc}_{\mathsf{pk}}(0; s_{\pi^{-1}(i)})$. The prover's witness is $(\pi, \boldsymbol{s}) \in S_N \times \mathbb{Z}_q^N$. The CRS consists of $\mathsf{pk} = [1 \,/\!/\, h]$, and $\mathsf{ck} = [h_1, \ldots, h_N]$ as in the extended Pedersen.

We will use the terms and algorithms from the coefficient-product argument and we will add a subscript G to them to distinguish them from the analogous terms in the main argument.

Theorem 2 (Security of the shuffle argument). *Π_{sh} is perfectly complete and perfectly SHVZK. If the commitment scheme is computationally binding then this shuffle argument has witness-extended emulation. More precisely, let $d = \max_i \deg \psi_i(\boldsymbol{X})$ and $d_{sum} = \sum_{i=1}^{N} \deg \psi_i(\boldsymbol{X})$. Let $\varepsilon \geq \zeta + 1/\sigma_{\mathsf{z}}$ be the success probability of P^* to make V to accept, $\varepsilon \geq \varepsilon' \geq \varepsilon - 1/\sigma_{\mathsf{z}}$, and $\zeta = \mathsf{ni} + \frac{N(N-1)}{2\sigma_{\mathsf{x}}^{\nu}} + \frac{N}{\sigma_{\mathsf{y}}}$. The emulator either recovers the witness or breaks the binding property of the commitment scheme with probability at least $1 - d_{sum}/\sigma_{\mathsf{x}} - N/\sigma_{\mathsf{y}} - 1/\sigma_{\mathsf{z}}$ and makes $\leq (2N+1)/(\varepsilon' - \zeta)$ queries to $\boldsymbol{V}$, where ζ is defined as above.*

1. The prover $\mathsf{P}(\mathsf{crs}, ([\boldsymbol{w}], [\widehat{\boldsymbol{w}}]), (\pi, \boldsymbol{s}))$ does:
   (a) For $i \in [1 \,..\, N-1]$: $r_i \leftarrow\$ \mathbb{Z}_q$; $\boxed{[u_i] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{e}_{\pi(i)}; r_{\pi(i)})}$; // $= [h_{\pi(i)}] + r_\pi(i)[1]$
   (b) $[u_N] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{1}_N; 0) - \sum_{i=1}^{N-1} [u_i]$.
   (c) Sample $\boldsymbol{\tau} \leftarrow\$ \mathbb{Z}_q^N$, $\varrho_{\mathsf{t}} \leftarrow\$ \mathbb{Z}_q$, $\varrho_f \leftarrow\$ \mathbb{Z}_q$.(\$)
   (d) Set $\boxed{[c_\tau] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{\tau}; \varrho_{\mathsf{t}})}$. (\$)
   (e) Set $\Delta_1 \leftarrow \tau_1$, $\Delta_N \leftarrow 0$. For $i \in [2 \,..\, N-1]$: sample $\Delta_i \leftarrow\$ \mathbb{Z}_q$.(\$)
   (f) For $i \in [1 \,..\, N-1]$: set $\beta_i \leftarrow -\tau_{i+1} \Delta_i$.(\$)
   (g) Sample $\varrho_b \leftarrow\$ \mathbb{Z}_q$. Denote $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{N-1})^\top$.
   (h) Compute $\boxed{[c_\beta] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{\beta} \,//\, 0; \varrho_b)}$.(\$)
   (i) Set $[\boldsymbol{F}_{\boldsymbol{\omega}}] \leftarrow \boxed{\sum_{i=1}^{N} \tau_i [\widehat{\boldsymbol{w}}_i]} + \mathsf{Enc}_{\mathsf{pk}}(0; -\varrho_f)$. // The only online step
   (j) Send $([u_1], \ldots, [u_{N-1}], [c_\tau], [c_\beta], [\boldsymbol{F}_{\boldsymbol{\omega}}])$ to the verifier.
2. The verifier generates random $\mathsf{x} \leftarrow \{0, \ldots, \sigma_{\mathsf{x}} - 1\}^\nu$. He sends x to the prover. For $i \in [1 \,..\, N]$: set $\mathsf{t}_i \leftarrow \psi_i(\mathsf{x})$.
3. The prover does the following:
   (a) For $i \in [1 \,..\, N]$: set $\mathsf{t}_i \leftarrow \psi_i(\mathsf{x})$.
   (b) For $i \in [1 \,..\, N]$: set $\hat{\mathsf{t}}_i \leftarrow \mathsf{t}_{\pi^{-1}(i)}$.
   (c) $X_0 \leftarrow 1$. For $i = [1 \,..\, N-1]$: $X_i \leftarrow X_{i-1} \hat{\mathsf{t}}_i$, $\mathsf{b}_i \leftarrow \Delta_{i+1} - \hat{\mathsf{t}}_{i+1} \Delta_i - \tau_{i+1} X_i$.(\$)
   (d) Let $\mathbf{b} := (\mathsf{b}_1, \ldots, \mathsf{b}_{N-1})^\top$. Sample $r_{\mathsf{b}} \leftarrow\$ \mathbb{Z}_q$. Set $\boxed{[c_b] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{b} \,//\, 0; r_{\mathsf{b}})}$.
   (e) Send $[c_b]$ to the verifier.
4. The verifier generates random $\mathsf{y} \leftarrow \{1, \ldots, \sigma_{\mathsf{y}}\}$. He sends y to the prover.
5. The prover does:
   (a) Set $r_{\mathsf{t}} \leftarrow \langle \hat{\mathbf{t}}, \boldsymbol{r} \rangle$, $r_f \leftarrow \langle \mathbf{t}, \boldsymbol{s} \rangle$.
   (b) Compute $\mathbf{t}^* \leftarrow \mathsf{y} \hat{\mathbf{t}} + \boldsymbol{\tau}$, $r_{\mathsf{t}}^* \leftarrow \mathsf{y} r_{\mathsf{t}} + \varrho_{\mathsf{t}}$, $r_f^* \leftarrow \mathsf{y} r_f + \varrho_f$.(\$)
   (c) Set $\mathbf{b}^* \leftarrow \mathsf{y}\mathbf{b} + \boldsymbol{\beta} \in \mathbb{Z}_q^{N-1}$, $r_{\mathsf{b}}^* \leftarrow \mathsf{y} r_{\mathsf{b}} + \varrho_b$.(\$)
   (d) Send $(\mathbf{t}^*, r_{\mathsf{t}}^*, r_f^*, \mathbf{b}^*, r_{\mathsf{b}}^*)$ to the verifier.
6. The verifier sets $\gamma \leftarrow \prod_{i=1}^{N} \mathsf{t}_i$, $[u_N] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{1}_N; 0) - \sum_{i=1}^{N-1} [u_i]$,

$$[\hat{c}_{\mathsf{t}}] \leftarrow \boxed{\langle \mathbf{t}, [\boldsymbol{u}] \rangle} \quad , \quad [\boldsymbol{F}] \leftarrow \boxed{\boxed{\langle \mathbf{t}, [\boldsymbol{w}] \rangle}} \quad . \tag{10}$$

   He generates a random $\mathsf{z} \leftarrow \{0, \ldots, \sigma_{\mathsf{z}} - 1\}$ for batch verification. He checks:

$$\mathsf{y}[\boldsymbol{F}] + [\boldsymbol{F}_{\boldsymbol{\omega}}] =^? \boxed{\sum_{i=1}^{N} \mathsf{t}_i^* [\widehat{\boldsymbol{w}}_i]} + \mathsf{Enc}_{\mathsf{pk}}(0; -r_f^*) \ , \tag{11}$$

$$\mathsf{y}[\hat{c}_{\mathsf{t}}] + [c_\tau] + \mathsf{z}(\mathsf{y}[c_b] + [c_\beta]) =^? \boxed{\mathsf{Com}_{\mathsf{ck}}(\mathbf{t}^* + \mathsf{z}(\mathbf{b}^* \,//\, 0); r_{\mathsf{t}}^* + \mathsf{z} r_{\mathsf{b}}^*)} \ .(\$) \tag{12}$$

   Set $Q_1 \leftarrow \mathsf{t}_1^*$, $Q_2 \leftarrow (\mathsf{t}_2^* Q_1 + \mathsf{b}_1^*)/\mathsf{y}$, $\ldots$, $Q_N \leftarrow (\mathsf{t}_N^* Q_{N-1} + \mathsf{b}_{N-1}^*)/\mathsf{y}$. He checks that $Q_N =^? \mathsf{y}\gamma$.(\$)
   Return $(\mathsf{acc}, \mathsf{z})$ iff all checks accept. Otherwise, return $(\mathsf{rej}, \mathsf{z})$.

Fig. 3: The new shuffle argument Π_{sh}. Dashed/Dotted formulas correspond to expensive ($\approx N$ fixed-base/multi exponentiations) computations only. A twice boxed formula signifies $\approx 2N$ operations.

Proof. First, if $[u_i]$ are honestly generated, then from Eq. (9) we get that $[\hat{c}_\mathsf{t}] = \langle [\boldsymbol{u}], \mathbf{t} \rangle = \mathsf{Com}_\mathsf{ck}(\sum_{i=1}^N \mathsf{t}_i \boldsymbol{e}_{\pi(i)}; \sum_{i=1}^N \mathsf{t}_i r_{\pi(i)}) = \mathsf{Com}_\mathsf{ck}(\sum \mathsf{t}_{\pi^{-1}(i)} \boldsymbol{e}_i; \sum \mathsf{t}_i r_{\pi(i)}) = \mathsf{Com}_\mathsf{ck}(\hat{\mathbf{t}}; r_\mathsf{t})$ for $\hat{\mathbf{t}}, r_\mathsf{t}$ defined as in Items 3b and 5a in Fig. 3.

Completeness: assume that $[\boldsymbol{w}_i] = \mathsf{Enc}_\mathsf{pk}(m_i; R_i)$ and $[\widehat{\boldsymbol{w}}_i] = \mathsf{Enc}_\mathsf{pk}(m_{\pi^{-1}(i)}; R_{\pi^{-1}(i)} + s_{\pi^{-1}(i)})$ for some (possibly unknown) m_i, R_i, and s_i. According to Eq. (10), $[\boldsymbol{F}] = \langle \mathbf{t}, [\boldsymbol{w}] \rangle = \mathsf{Enc}_\mathsf{pk}(\langle \mathbf{t}, \boldsymbol{m} \rangle; \langle \mathbf{t}, \boldsymbol{R} \rangle)$, and according to Item 1i,

$$\begin{aligned}
[\boldsymbol{F}_{\boldsymbol{\omega}}] &= \textstyle\sum_{i=1}^N \tau_i [\widehat{\boldsymbol{w}}_i] + \mathsf{Enc}_\mathsf{pk}(0; -\varrho_f) \\
&= \textstyle\sum_{i=1}^N \tau_i \cdot \mathsf{Enc}_\mathsf{pk}(m_{\pi^{-1}(i)}; R_{\pi^{-1}(i)} + s_{\pi^{-1}(i)}) + \mathsf{Enc}_\mathsf{pk}(0; -\varrho_f) \\
&= \mathsf{Enc}_\mathsf{pk}(\langle \boldsymbol{\tau}, \hat{\boldsymbol{m}} \rangle; \langle \boldsymbol{\tau}, \widehat{\boldsymbol{R}} + \hat{\boldsymbol{s}} \rangle - \varrho_f) \ ,
\end{aligned}$$

where $\hat{s}_i = s_{\pi^{-1}(i)}$ and $\widehat{R}_i = R_{\pi^{-1}(i)}$. Denoting $\hat{m}_i = m_{\pi^{-1}(i)}$, $\mathsf{y}[\boldsymbol{F}] + [\boldsymbol{F}_{\boldsymbol{\omega}}] + \mathsf{Enc}_\mathsf{pk}(0; r_f^*) = \mathsf{Enc}_\mathsf{pk}(\mathsf{y}\langle \mathbf{t}, \boldsymbol{m} \rangle + \langle \boldsymbol{\tau}, \hat{\boldsymbol{m}} \rangle; \mathsf{y}\langle \mathbf{t}, \boldsymbol{R} \rangle + \langle \boldsymbol{\tau}, \widehat{\boldsymbol{R}} + \hat{\boldsymbol{s}} \rangle - \varrho_f + \varrho_f) = \mathsf{Enc}_\mathsf{pk}(\langle \mathbf{t}^*, \hat{\boldsymbol{m}} \rangle; \langle \mathbf{t}^*, \widehat{\boldsymbol{R}} + \hat{\boldsymbol{s}} \rangle)$, which is equal to $\langle \mathbf{t}^*, [\widehat{\boldsymbol{w}}] \rangle$ as required by Eq. (11).

Next, if $\boldsymbol{M}$ is the permutation matrix corresponding to the permutation π, then $M_{ij} = 1$ iff $i = \pi(j)$. Hence, $\boldsymbol{M}^{(j)} = \boldsymbol{e}_{\pi(j)}$ and $[u_i] = \mathsf{Com}_\mathsf{ck}(\boldsymbol{e}_{\pi(j)}; \hat{r}_i) = \mathsf{Com}_\mathsf{ck}(\boldsymbol{M}^{(i)}; \hat{r}_i)$. According to Eq. (10), $[\hat{c}_\mathsf{t}] = \langle \mathbf{t}, [\boldsymbol{u}] \rangle = \mathsf{Com}_\mathsf{ck}(\boldsymbol{M}\mathbf{t}; \langle \hat{\boldsymbol{r}}, \mathbf{t} \rangle) = \mathsf{Com}_\mathsf{ck}(\hat{\mathbf{t}}; \langle \hat{\boldsymbol{r}}, \mathbf{t} \rangle)$, where $\hat{\mathsf{t}}_i = \mathsf{t}_{\pi^{-1}(i)}$. According to Items 1d and 5b, $\mathsf{y}[\hat{c}_\mathsf{t}] + [c_\tau] = \mathsf{Com}_\mathsf{ck}(\mathsf{y}\hat{\mathbf{t}} + \boldsymbol{\tau}; \mathsf{y}r_\mathsf{t} + \varrho_\mathsf{t}) = \mathsf{Com}_\mathsf{ck}(\mathbf{t}^*; r_\mathsf{t}^*)$. According to Items 1h, 3d and 5c, $\mathsf{y}[c_b] + [c_\beta] = \mathsf{Com}_\mathsf{ck}((\mathsf{y}\mathbf{b} + \boldsymbol{\beta}) /\!/ 0; \mathsf{y}r_\mathsf{b} + \varrho_b) = \mathsf{Com}_\mathsf{ck}(\mathbf{b}^* /\!/ 0; r_\mathsf{b}^*)$. Thus, $\mathsf{y}[\hat{c}_\mathsf{t}] + [c_\tau] + \mathsf{z}(\mathsf{y}[c_b] + [c_\beta]) = \mathsf{Com}_\mathsf{ck}(\mathbf{t}^* + \mathsf{z}(\mathbf{b}^* /\!/ 0); r_\mathsf{t}^* + \mathsf{z}r_\mathsf{b}^*)$. Hence, Eq. (12) holds.

Perfect HVZK: Assume that we are given $\mathbf{t}$ and y. For fixed $\mathbf{t}$, y, z, the simulator is depicted in Fig. 4. Clearly, the verifier of Fig. 3 will accept the simulated proof (this is taken care of by Lines 2, 3 and 5, together with the definition of b_i^* and Q_N).

Similarly to the proof of Theorem 1, we can argue that the output of the simulator Sim follows the correct distribution (we first change Line 4, obtaining a hybrid simulator Sim_t, and then argue that the outputs of Sim and Sim_t come from the same distribution, and the output of Sim_t and the real protocol come from the same distribution). Thus, we get a simulator for the shuffle argument.

Witness-extended emulation: As in the proof of Theorem 1, from Eq. (12) it follows — since this is a standard batch verification, [2] — that with probability $1 - 1/\sigma_\mathsf{z}$, Eq. (7) holds. Hence, as in the case of the coefficient-product argument, we define another shuffle argument Π'_sh that is exactly the same as Π_sh, except that the verifier accepts not when Eq. (12) holds but Eq. (7) holds (that is, we do not use batch verification.) It is clear that if an adversary succeeds with probability ε against Π_sh, then it succeeds with probability not larger than ε', $\varepsilon \geq \varepsilon' \geq \varepsilon - 1/\sigma_\mathsf{z}$, against Π'_sh. Let V' be the verifier of Π'_sh.

Let P^* be a prover that makes V' to accept with probability ε'. We first run the argument and obtain one accepting transcript tr. If P^* fails to produce an acceptable transcript, then we reject. Assume now that the transcript is acceptable. In this case we need to extract a witness $(\pi, \boldsymbol{s})$ that $([\widehat{\boldsymbol{w}}_i])$ is a shuffle of $([\boldsymbol{w}_i])$. For this, we rewind the argument to get more transcripts with randomly

Compute $(\mathbf{x}, \mathsf{y}, \mathsf{z})$ from ω_{V};
Set $\mathsf{t}_i \leftarrow \psi_i(\mathbf{x})$; $\gamma \leftarrow \prod_{i=1}^{N} \mathsf{t}_i$;
for $i \in [1\,..\,N-1]$ **do** sample $r_i \leftarrow\$\, \mathbb{Z}_q$; set $[u_i] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{e}_i; r_i)$;
Set $[u_N] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{1}_N; 0) - \sum_{i=1}^{N-1}[u_i]$; Set $[\hat{c}_{\mathsf{t}}] \leftarrow \langle \mathbf{t}, [\boldsymbol{u}] \rangle$;
Sample $\mathbf{t}^* \leftarrow\$\, \mathbb{Z}_q^N$, $r_{\mathsf{t}}^* \leftarrow\$\, \mathbb{Z}_q$;
2 Set $[c_\tau] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{t}^*; r_{\mathsf{t}}^*) - \mathsf{y}[\hat{c}_{\mathsf{t}}]$; // need: $\mathsf{y}[\hat{c}_{\mathsf{t}}] + [c_\tau] =^{?} \mathsf{Com}_{\mathsf{ck}}(\mathbf{t}^*; r_{\mathsf{t}}^*)$
Set $[\boldsymbol{F}] \leftarrow \langle \mathbf{t}, [\boldsymbol{w}] \rangle$; Sample $r_f^* \leftarrow\$\, \mathbb{Z}_q$;
3 Set $[\boldsymbol{F}_{\boldsymbol{\omega}}] \leftarrow \langle \mathbf{t}^*, [\widehat{\boldsymbol{w}}] \rangle + \mathsf{Enc}_{\mathsf{pk}}(0; -r_f^*) - \mathsf{y}[\boldsymbol{F}]$;
// need: $\mathsf{y}[\boldsymbol{F}] + [\boldsymbol{F}_{\boldsymbol{\omega}}] =^{?} \langle \mathbf{t}^*, [\widehat{\boldsymbol{w}}] \rangle + \mathsf{Enc}_{\mathsf{pk}}(0; -r_f^*)$;
Set $Q_1 \leftarrow \mathsf{t}_1^*$, $Q_i \leftarrow\$\, \mathbb{Z}_q$ for $i \in \{2, \ldots, N-1\}$, and $Q_N \leftarrow \mathsf{y}\gamma$;
Sample $\mathsf{b}_1^* \leftarrow\$\, \mathbb{Z}_q$; Set $\mathsf{b}_2^* \leftarrow \mathsf{y}Q_2 - \mathsf{t}_2^* Q_1$, $\ldots$, $\mathsf{b}_N^* \leftarrow \mathsf{y}Q_N - \mathsf{t}_N^* Q_{N-1}$;
Sample $r_{\mathsf{b}}^* \leftarrow\$\, \mathbb{Z}_q$; Sample $r_{\mathsf{b}} \leftarrow\$\, \mathbb{Z}_q$;
4 Set $[c_b] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{0}_N; r_{\mathsf{b}})$;
5 Set $[c_\beta] \leftarrow \mathsf{Com}_{\mathsf{ck}}(\mathbf{b}^* \,//\, 0; r_{\mathsf{b}}^*) - \mathsf{y}[c_b]$ // need: $\mathsf{y}[c_b] + [c_\beta] =^{?} \mathsf{Com}_{\mathsf{ck}}(\mathbf{b}^* \,//\, 0; r_{\mathsf{b}}^*)$;
Return $([u_1], \ldots, [u_{N-1}], [c_\tau], [c_\beta], [\boldsymbol{F}_{\boldsymbol{\omega}}]; \mathbf{x}; [c_b]; \mathsf{y}; \mathbf{t}^*, r_{\mathsf{t}}^*, r_f^*, \mathbf{b}^*, r_{\mathsf{b}}^*; \mathsf{acc}, \mathsf{z})$;

Fig. 4: The simulator $\mathsf{Sim}(\mathsf{crs}, \mathtt{inp} = ([\boldsymbol{w}], [\widehat{\boldsymbol{w}}]); \omega_{\mathsf{V}})$ in the proof of Theorem 2.

chosen challenges $\mathbf{x}, \mathsf{y}$, and use the witness-extended emulator of Π'_{cpa} to get openings of $[c_{\mathsf{t}}]$. We repeat this until we obtain $N+1$ acceptable transcripts. Let $tr_1 = tr$, and let the additional transcripts be tr_j, $j > 1$, where $tr_j =$

$$(([\boldsymbol{w}, \widehat{\boldsymbol{w}}])_{i=1}^{N}; ([u_i])_{i=1}^{N-1}; [c_\tau, c_\beta, F_\omega]; \mathbf{x}_j; [c_{\mathsf{b}:j}]; \mathsf{y}_j; \mathbf{t}_j^*, r_{\mathsf{t}:j}^*, r_{f:j}^*, \mathbf{b}_j^*, r_{\mathsf{b}:j}^*; \mathsf{acc}, \mathsf{z}_j) \ .$$

Clearly, the expected number of rewindings for this is N/ε. However, since we only need to extract a witness when the transcript is acceptable, the expected number of rewindings is only N. (As in [21], one can argue that combining expected polynomial-time algorithms results in an expected polynomial-time argument.) Since the emulator uses an expected polynomial number of rewindings, with an overwhelming probability it is the case that either (1) the argument is not acceptable, or (2) the argument is acceptable, but no event with negligible probability (like breaking the binding property of the commitment scheme or having collisions among randomly chosen challenges) occurs. Assume from now on that either (2) holds.

Let us show that Emul obtains the witness. Let $\mathbf{t}_j = \boldsymbol{\psi}(\mathbf{x}_j)$ and $[\hat{c}_{\mathsf{t}:j}] = \langle \mathbf{t}_j, [\boldsymbol{u}] \rangle$. From Eq. (7) (left) and the jth transcript, we get

$$\mathsf{y}_j \mathbf{t}_j^\top [\boldsymbol{u}] + [c_\tau] = \mathsf{y}_j \langle \mathbf{t}_j, [\boldsymbol{u}] \rangle + [c_\tau] = \mathsf{y}_j [\hat{c}_{\mathsf{t}:j}] + [c_\tau] = \langle \mathbf{t}_j^*, [\boldsymbol{h}] \rangle + r_{\mathsf{t}:j}^*[1] \ .$$

Hence,

$$(\mathsf{y}_j \mathbf{t}_j^\top - \mathsf{y}_1 \mathbf{t}_1^\top)[\boldsymbol{u}] = \langle \mathbf{t}_j^*, [\boldsymbol{h}] \rangle + r_{\mathsf{t}:j}^*[1] \ .$$

Denote $\boldsymbol{T}_{\mathsf{y}} = (\mathsf{y}_2 \mathbf{t}_2 - \mathsf{y}_1 \mathbf{t}_1 \| \ldots \| \mathsf{y}_{N+1} \mathbf{t}_{N+1} - \mathsf{y}_1 \mathbf{t}_1)^\top$ and $\boldsymbol{T}^* = (\mathsf{y}_2 \mathbf{t}_2^* - \mathsf{y}_1 \mathbf{t}_1^* \| \ldots \| \mathsf{y}_{N+1} \mathbf{t}_{N+1}^* - \mathsf{y}_1 \mathbf{t}_1^*)^\top$. Since $\boldsymbol{T} = (\mathbf{t}_1 \| \ldots \| \mathbf{t}_N)$ is invertible with (this follows from Lemma 4), then also $\boldsymbol{T}_{\mathsf{y}}$ is invertible with overwhelming probability. Define $\boldsymbol{M} := \boldsymbol{T}_{\mathsf{y}}^{-1} \boldsymbol{T}^*$. Thus, $\boldsymbol{T}_{\mathsf{y}}[\boldsymbol{u}] = \boldsymbol{T}^*[\boldsymbol{h}] + (\boldsymbol{r}_{\mathsf{t}}^* - r_1^* \mathbf{1}_N)[1]$, and thus

$$[\boldsymbol{u}] = \boldsymbol{M}[\boldsymbol{h}] + \boldsymbol{T}_{\mathsf{y}}^{-1}(\boldsymbol{r}_{\mathsf{t}}^* - r_1^* \mathbf{1}_N)[1] \ .$$

Denote $[\boldsymbol{E}_j] := \mathsf{Enc}_{\mathsf{pk}}(0; r^*_{f:j})$. From Eq. (11) (left), we have $\mathsf{y}_j \mathbf{t}_j^\top [\boldsymbol{w}] + [\boldsymbol{F}_{\boldsymbol{\omega}}] = (\mathbf{t}^*_j)^\top [\widehat{\boldsymbol{w}}] + [\boldsymbol{E}_j]$, and thus $(\mathsf{y}_j \mathbf{t}_j - \mathsf{y}_1 \mathbf{t}_j)^\top [\boldsymbol{w}] = (\mathbf{t}^*_j - \mathbf{t}^*_1)^\top [\widehat{\boldsymbol{w}}] + [\boldsymbol{E}_j - \boldsymbol{E}_1]$. Thus, $\boldsymbol{T}_{\mathsf{y}}[\boldsymbol{w}] = \boldsymbol{T}^*[\widehat{\boldsymbol{w}}] + [\boldsymbol{E}] - \mathbf{1}_N[\boldsymbol{E}_1]$ and thus

$$[\boldsymbol{w}] = \boldsymbol{M}[\widehat{\boldsymbol{w}}] + \boldsymbol{T}_{\mathsf{y}}^{-1}([\boldsymbol{E}] - \mathbf{1}_N[\boldsymbol{E}_1]) \ .$$

Hence, assuming $\boldsymbol{M}$ is a permutation matrix, we have recovered a permutation π and a randomness $\boldsymbol{s}$, such that $[\widehat{\boldsymbol{w}}_i] = [\boldsymbol{w}_{\pi^{-1}(i)}] + \mathsf{Enc}_{\mathsf{pk}}(0; s_{\pi^{-1}(i)})$.

Next, we argue that $\boldsymbol{M}$ is a permutation matrix. Assume that P^* has a non-negligible success probability $1/f(\lambda)$, for a polynomial $f(X)$, to produce an acceptable argument. We run P^* and rewind to get $N+2$ random challenges. We extract $\boldsymbol{M}$ and other values from the first $N+1$ transcripts as above.

Consider the $(N+2)$th argument. Define $\hat{\mathbf{t}}_{N+2} := \boldsymbol{M}^\top \mathbf{t}_{N+2}$. Thus, $\prod_{i=1}^N \langle \boldsymbol{M}^{(i)}, \boldsymbol{\psi}(\boldsymbol{x}_{N+2}) \rangle = \prod_{i=1}^N \langle \boldsymbol{M}^{(i)}, \mathbf{t}_{N+2} \rangle = \prod_{i=1}^N \hat{\mathsf{t}}_{N+2:i}$. Since Π_{cpa} has witness-extended emulation, then its emulator returns an opening of $[\hat{c}_{\mathsf{t}:N+2}]$ whose coefficient-product is equal to $\gamma_{N+2} := \prod_{i=1}^N \mathsf{t}_{N+2:i}$. Since the commitment scheme is binding and $[\hat{c}_{\mathsf{t}:N+2}] = \mathbf{t}_{N+2}^\top [\boldsymbol{u}]$, the opening is equal to $\boldsymbol{M}^\top \mathbf{t}_{N+2} = \hat{\mathbf{t}}_{N+2}$. Thus, by the soundness of the coefficient-product emulation, $\prod_{i=1}^N \hat{\mathsf{t}}_{N+2:i} = \prod_{i=1}^N \mathsf{t}_{N+2:i}$. Hence, $\prod_{i=1}^N \langle \boldsymbol{M}^{(i)}, \boldsymbol{\psi}(\mathbf{x}_{N+2}) \rangle = \prod_{i=1}^N \mathsf{t}_{N+2:i} = \prod_{i=1}^N \psi_i(\mathbf{x}_{N+2:i})$. Due to the Schwartz-Zippel lemma, from this it follows with an overwhelming probability that $\prod_{i=1}^N \langle \boldsymbol{M}^{(i)}, \boldsymbol{\psi}(\boldsymbol{X}_{N+2}) \rangle = \prod_{i=1}^N \psi_i(\boldsymbol{X}_{N+2:i})$ as a polynomial.

The ith row of $\boldsymbol{M} \cdot \mathbf{1}_N$ is $\sum_{j=1}^N M_{ij} = 1$ due to the choice of $[u_N]$, and thus $\hat{\boldsymbol{M}} \cdot \mathbf{1}_N = \mathbf{1}_N$. It follows now from Lemma 2 that $\hat{\boldsymbol{M}}$ is a permutation matrix. Thus, with an overwhelming probability, the emulator has extracted $\pi \in S_N$, the permutation corresponding to $\hat{\boldsymbol{M}}$, such that $\hat{\mathsf{t}}_i = \mathsf{t}_{\pi^{-1}(i)}$. □

6 Efficiency

Recall that one N-wide multi-exponentiation and N fixed-base exponentiations by ℓ-bit exponent can be done significantly faster than N arbitrary exponentiations. Importantly, in the new shuffle argument, neither the prover or the verifier has to execute the latter.

Clearly, the prover's computation in the shuffle argument of Fig. 3 is dominated by four ($\approx N$)-wide multi-exponentiations and N fixed-base exponentiations. The verifier's computation is dominated by six $\approx N$-wide multi-exponentiations. The communication is dominated by $([u_1], \ldots, [u_{N-1}], \mathbf{b}^*)$, that is, by $(\ell_{\mathbb{G}} + \log q)N + O(\lambda)$ bits, where $\ell_{\mathbb{G}}$ is the number of bits it takes to represent an element of $\mathbb{G}$. In practice, we can assume $\log q = 128$ and $\ell_{\mathbb{G}} = 256$, in this case the communication is dominated by $388N$ bits. (Note that in the introduction, we already gave an extensive comparison with other shuffles.)

Online Computation. As remarked in [33], online computational complexity (i.e., computation done after the input data — in this case, the ciphertexts —

has arrived) is an important separate measure of the shuffle arguments. In the online phase of the protocol on Fig. 3, the prover's computation is dominated by two ($\approx N$)-wide multi-exponentiations (computation of $[\boldsymbol{F}_{\boldsymbol{\omega}}]$), and the verifier's computation is dominated by four ($\approx N$)-wide multi-exponentiations (the computation of $[\boldsymbol{F}]$ and the verification of $[\boldsymbol{F}_{\boldsymbol{\omega}}]$).

The case of larger ciphertexts. We assumed that each ciphertext $[\boldsymbol{w}_i]$ / $[\widehat{\boldsymbol{w}}_i]$ corresponds to one Elgamal ciphertext. However, in practice it might be the case — say, if the ballot is complex — that each $[\boldsymbol{w}_i]$ / $[\widehat{\boldsymbol{w}}_i]$ corresponds to $m > 1$ Elgamal ciphertexts. This only changes the "type" of $[\boldsymbol{w}_i]$ / $[\widehat{\boldsymbol{w}}_i]$ in Fig. 3. Efficiency-wise, the prover then has to perform $2m + 2$ ($\approx N$)-wide multi-exponentiations and N fixed-base exponentiations, while the verifier has to perform $4m + 2$ ($\approx N$)-wide multi-exponentiations.

7 Discussions

Comparison to Bayer-Groth. All shuffle arguments mentioned in Table 1 have linear argument size. Bayer and Groth [1] proposed a shuffle argument that achieves sublinear argument size but pays with higher computation. While sublinear argument size is an excellent property to have, its influence is decreased because the storage of ciphertexts makes the communication and storage requirements linear anyhow. Computation-wise, Bayer and Groth [1] include a comparison with Verificatum [32], claiming that the total computation of the prover and the verifier in Verificatum is $20N$ exponentiations, and in Bayer-Groth it is $16N$ exponentiations. [1] does not distinguish fixed-base exponentiations, multi-exponentiations, and "usual" exponentiations, Hence, we expect it to be slower than both [21] and the new shuffle, especially since the latter shuffles do not include any "usual" exponentiations. Finally, the optimized version of the Bayer and Groth shuffle takes nine rounds, compared to the five rounds in the new shuffle.

PM-Evidential Polynomials and Random Oracle Model. In practice, one would use the Fiat-Shamir heuristic to modify the shuffle argument to be non-interactive, which results in the security proof being in the random oracle model. A natural question that may arise is the necessity of minimizing the verifier's communication in that case since one would use a random oracle to generate the verifier's response. In a setting like in [32], the standard approach is to generate N random strings by applying the random oracle N times. In the new shuffle, one only has to apply the random oracle twice instead of N times.

Bellare and Rogaway [5] argue that it is better to rely less on random oracles. Quoting [5],

> *But there may remain some lingering fear that the concrete hash function instantiates the random oracle differs from a random function in some significant way. So it is good to try to limit reliance on random oracles.*

We refer to [5] for more discussion.

Acknowledgments. We thank Douglas Wikström and Janno Siim for helpful discussions. The authors were partially supported by the Estonian Research Council grant (PRG49).

References

1. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280
2. Bellare, M., Garay, J.A., Rabin, T.: Batch Verification with Applications to Cryptography and Checking. In: LATIN 1998. LNCS, vol. 1380, pp. 170–191
3. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: CRYPTO'92. LNCS, vol. 740, pp. 390–420
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 93, pp. 62–73
5. Bellare, M., Rogaway, P.: Minimizing the use of random oracles in authenticated encryption schemes. In: ICICS 97. LNCS, vol. 1334, pp. 1–16
6. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: 20th ACM STOC, pp. 103–112
7. Brands, S.: Rapid demonstration of linear relations connected by Boolean operators. In: EUROCRYPT'97. LNCS, vol. 1233, pp. 318–333
8. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th ACM STOC, pp. 209–218
9. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM **24**(2) (1981) pp. 84–88
10. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: CRYPTO'94. LNCS, vol. 839, pp. 174–187
11. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: CRYPTO'84. LNCS, vol. 196, pp. 10–18
12. Fauzi, P., Lipmaa, H.: Efficient culpably sound NIZK shuffle argument without random oracles. In: CT-RSA 2016. LNCS, vol. 9610, pp. 200–216
13. Fauzi, P., Lipmaa, H., Siim, J., Zajac, M.: An efficient pairing-based shuffle argument. In: ASIACRYPT 2017, Part II. LNCS, vol. 10625, pp. 97–127
14. Fauzi, P., Lipmaa, H., Zajac, M.: A shuffle argument secure in the generic model. In: ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 841–872
15. Fauzi, P., Meiklejohn, S., Mercer, R., Orlandi, C.: Quisquis: A new design for anonymous cryptocurrencies. In: ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 649–678
16. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO'86. LNCS, vol. 263, pp. 186–194
17. Furukawa, J.: Efficient and Verifiable Shuffling and Shuffle-Decryption. IEICE Transactions **88-A**(1) (2005) pp. 172–188
18. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: CRYPTO 2001. LNCS, vol. 2139, pp. 368–387
19. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: 44th FOCS, pp. 102–115
20. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC, pp. 291–304

21. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. Journal of Cryptology **23**(4) (2010) pp. 546–579
22. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 253–280
23. Groth, J., Lu, S.: A non-interactive shuffle with pairing based verifiability. In: ASIACRYPT 2007. LNCS, vol. 4833, pp. 51–67
24. Hungerford, T.W.: Algebra. 8 edn. Graduate Texts in Mathematics, vol. 73. Springer-Verlag (1980)
25. Khazaei, S., Moran, T., Wikström, D.: A mix-net from any CCA2 secure cryptosystem. In: ASIACRYPT 2012. LNCS, vol. 7658, pp. 607–625
26. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. In: CRYPTO 2001. LNCS, vol. 2139, pp. 171–189
27. Lipmaa, H., Zhang, B.: A more efficient computationally sound non-interactive zero-knowledge shuffle argument. In: SCN 12. LNCS, vol. 7485, pp. 477–502
28. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: ACM CCS 2001, pp. 116–125
29. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: CRYPTO'91. LNCS, vol. 576, pp. 129–140
30. Schwartz, J.T.: Fast Probabilistic Algorithms for Verification of Polynomial Identities. Journal of the ACM **27**(4) (1980) pp. 701–717
31. Straus, E.G.: Addition Chains of Vectors. Amer. Math. Monthly **70** (1964) pp. 806–808
32. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: AFRICACRYPT 10. LNCS, vol. 6055, pp. 100–113
33. Wikström, D.: A Commitment-Consistent Proof of a Shuffle. In: ACISP 2009. LNCS, vol. 5594, pp. 4007–421
34. Wikström, D.: How to Implement a Stand-alone Verifier for the Verificatum Mix-Net. Version 1.4.1, available from `http://www.verificatum.org` (2015)
35. Zippel, R.: Probabilistic Algorithms for Sparse Polynomials. In: EUROSM 1979. LNCS, vol. 72, pp. 216–226