An Efficient Pairing-Based Shuffle Argument

Prastudy Fauzi[1], Helger Lipmaa[2], Janno Siim[2,3], and Michał Zając[2]

[1] Aarhus University, Aarhus, Denmark
[2] University of Tartu, Tartu, Estonia
[3] STACC, Tartu, Estonia

Abstract. We construct the most efficient known pairing-based NIZK shuffle argument. It consists of three subarguments that were carefully chosen to obtain optimal efficiency of the shuffle argument:

1. A same-message argument based on the linear subspace QANIZK argument of Kiltz and Wee,
2. A (simplified) permutation matrix argument of Fauzi, Lipmaa, and Zając,
3. A (simplified) consistency argument of Groth and Lu.

We prove the knowledge-soundness of the first two subarguments in the generic bilinear group model, and the culpable soundness of the third subargument under a KerMDH assumption. This proves the soundness of the shuffle argument. We also discuss our partially optimized implementation that allows one to prove a shuffle of 100 000 ciphertexts in less than a minute and verify it in less than 1.5 minutes.

Keywords: Common reference string, generic group model, mix-net, shuffle argument, zero knowledge

1 Introduction

Consider the case of using mix-networks [9] in e-voting, where n voters individually encrypt their vote using a blindable public-key cryptosystem and send the encrypted votes to a bulletin board. After the vote casting period ends, the first mix-server gets all encrypted votes from the bulletin board. The mix-servers are ordered sequentially, creating a mix-network, and it is assumed that some of them are honest. The kth mix-server obtains input ciphertexts $(\mathfrak{M}_i)_{i=1}^n$, shuffles them, and sends the resulting ciphertext tuple $(\mathfrak{M}'_i)_{i=1}^n$ to the next mix-server. Shuffling means that the mix-server generates a random permutation $\sigma \leftarrow_r S_n$ and a vector $\boldsymbol{t}$ of randomizers, and sets $\mathfrak{M}'_i = \mathfrak{M}_{\sigma(i)} + \mathsf{Enc}_{\mathsf{pk}}(\mathfrak{o}; t_i)$. [4]

If at least one of the mix-servers behaves honestly, the link between a voter and his votes is completely hidden. However, in the malicious model, a corrupt mix-server can do an incorrect shuffle, resulting in a set of decrypted votes that

[4] Throughout this paper, we use additive notation combined with the bracket notation of [13]. We also denote group elements by using the Fraktur script as in $\mathfrak{M}_i$ or $\mathfrak{o}$. Thus, adding $\mathsf{Enc}_{\mathsf{pk}}(\mathfrak{o}; t_i)$ results in a blinded version of $\mathfrak{M}_{\sigma(i)}$.

do not reflect the original voters' votes. Hence there needs to be some additional steps to achieve security against corruption.

The cryptographically prudent way to proceed is to get each mix-server to prove in zero-knowledge [18] that her shuffle was done correctly. The resulting proof is known as a *(zero-knowledge) shuffle argument*. Based on earlier work [25,34], in CT-RSA 2016, Fauzi and Lipmaa (FL, [14]) proposed the then most efficient shuffle argument in the common reference string (CRS, [8]) model in terms of prover's computation.[5] Importantly, the FL shuffle argument is based on the standard Elgamal cryptosystem. The culpable soundness [25,26] of the FL shuffle argument is proven under a knowledge assumption [10] and three computational assumptions. Intuitively, culpable soundness means that if a cheating adversary produces an invalid shuffle (yet accepted by the verifier) together with the secret key, then one can break one of the underlying knowledge or computational assumptions.

While the FL shuffle argument is quite efficient for the prover, it is quite inefficient for the verifier. More precisely, while the prover's online cost is only dominated by $4n$ exponentiations in $\mathbb{G}_1$, the verifier's online cost is dominated by $8n$ pairings. (See Tbl. 1.) Depending on the concrete implementation, a pairing can be up to 8 times slower than a $\mathbb{G}_1$ exponentiation. Such a large gap is non-satisfactory since verification time is more important in practice.

In ACNS 2016, González and Ràfols [20] proposed a new shuffle argument that importantly relies only on standard (falsifiable) assumptions. However, they achieve this (and efficient computation) by allowing the CRS to be quadratically long in n. Since in e-voting applications one could presumably have $n > 2^{20}$, quadratic CRS length will not be acceptable in such applications.

In Asiacrypt 2016, Fauzi, Lipmaa and Zając (FLZ, [15]) improved on the efficiency of the FL shuffle argument by proving knowledge-soundness (not culpable soundness) in the generic bilinear group model (GBGM, [39,35]). By additionally using batching techniques, they sped up the verification time of the FL shuffle argument approximately 3.5 times and the online verification time approximately twice. Here, the precise constants depend on how efficient operations such pairings and exponentiations in different groups are relative to each other; they do not provide an implementation of their shuffle.

However, due to the construction of the FLZ argument, they need each message to be encrypted twice, in $\mathbb{G}_1$ and $\mathbb{G}_2$. Since Elgamal cannot guarantee security in such a case, they use the non-standard ILin cryptosystem of Escala *et al.* [13]. This means that each ciphertext in this case will consist of 6 group elements, which makes storing and transmitting them more burdensome.

This results in several concrete drawbacks. First, since the prover's online complexity includes shuffling the ciphertexts and uses a non-standard cryptosystem (amongst other things), the prover's online complexity in the FLZ shuffle

[5] Many random-oracle model shuffle arguments are known, such as [16,3,22]. We will not provide comparisons with such arguments or discussions about the benefits of the CRS vs the random oracle model. We remark that the CRS can be created by using multi-party computation, see, e.g., [5]

argument is more than 4 times worse (using the comparison given in [15]) than in the FL shuffle argument. Second, since plaintexts in this shuffle argument are elements of $\mathbb{Z}_q$, decryption requires computing a discrete logarithm, which means that plaintexts must be small elements of $\mathbb{Z}_q$ (e.g. only 40 bits long). This rules out voting mechanisms with sufficiently complex ballots, such as the single transferable vote. Third, the GBGM knowledge-soundness proof of this shuffle argument means that there must exist a generic adversary that knows the discrete logarithm of each ballot submitted by each individual voter. Such a version of soundness (named *white-box soundness* in [14]) seems to be more dubious than culpable soundness achieved by the Groth-Lu [25] and FL shuffle arguments. (See [14] for more thorough comparison of these two soundness definitions.) Fourth, the CRS of this shuffle argument has public keys in both $\mathbb{G}_1$ and $\mathbb{G}_2$, which makes it more difficult to design an efficient shuffle or to prove its soundness in the GBGM. Indeed, [15] used a computer algebra system to derive knowledge-soundness of their shuffle argument.

This brings us to the main question of this paper:

> *Is it possible to construct a NIZK shuffle argument that shares the best features of the FL and the FLZ shuffle arguments? That is, it would use standard Elgamal (and in only one group), be (non-whitebox) sound, have linear-length CRS, have prover as efficient or better than in the FL shuffle argument, and have verifier as efficient or better than in the FLZ shuffle argument. Moreover, can one simplify (significantly?) the soundness proof of the FLZ shuffle argument while not losing in efficiency?*

Our Constructions. We answer the main question positively, constructing a new pairing-based NIZK shuffle argument that is more efficient than prior work in essentially all parameters. As in [14], we use the Elgamal cryptosystem (with plaintexts in $\mathbb{G}_2$), which means that unlike [15], compatible voting mechanisms are not restricted by the size of the plaintext space. We construct more efficient subarguments, which sometimes leads to a significant efficiency gain. Since the CRS has very few elements from $\mathbb{G}_2$, the new shuffle has much simpler soundness proofs than in the case of the FLZ shuffle argument. Moreover, as in [25,14] (but not in [34,15]), we do not give the generic adversary in our soundness proof access to the discrete logarithms of encrypted messages. Our high-level approach in the shuffle argument is as follows; it is similar to the approach in the FL shuffle argument except that we use (significantly) more efficient subarguments.

We first let the prover choose a permutation matrix and commit separately to its every row. The prover then proves that the committed matrix is a permutation matrix, by proving that each row is a unit vector, including the last row which is computed explicitly, see Sect. 4.2. We construct a new unit vector argument based on the square span programs of Danezis *et al.* [11]; it is similar to but somewhat simpler than the 1-sparsity argument of [15]. Basically, to show that a vector $\boldsymbol{a}$ is unit vector, we choose polynomials $(P_i(X))_{i \in [0\,..\,n]}$ that interpolate a certain matrix (and a certain vector) connected to the definition of "unit vectorness", and then commit to $\boldsymbol{a}$ by using a version of the extended

Pedersen commitment scheme, $\mathfrak{c} = \sum_{i=1}^{n} a_i [P_i(\chi)]_1 + r[\varrho]_1$ for trapdoor values (χ, ϱ) and randomizer r. (This commitment scheme, though for different polynomials $P_i(X)$, was implicitly used first by Groth [24] in EUROCRYPT 2016, and then used in the FLZ shuffle argument; similar commitment schemes have been used before [19,23,32].) The new unit vector argument differs from the corresponding (1-sparsity) argument in [15] by a small optimization that makes it possible to decrease the number of trapdoor elements by one. If the unit vector argument for each row is accepting, it follows that the committed matrix is a permutation matrix [14]. The knowledge-soundness proof of the new unit vector argument is almost trivial, in contrast to the very complex machine-assisted knowledge-soundness proof in [15].

We then use the same high-level idea as previous NIZK shuffle arguments [25,34,14,15] to obtain a shuffle argument from a permutation matrix argument. Namely, we construct a verification equation that holds tautologically under a corresponding KerMDH [36] assumption. That is, if the permutation matrix argument is knowledge-sound, the mentioned verification equation holds, and the KerMDH assumption holds, then the prover has used his committed permutation matrix to also shuffle the ciphertexts.

However, as in [25,34,14,15], the resulting KerMDH assumption itself will not be secure if we use here the same commitment scheme as before. Intuitively, this is since the polynomials $P_i(X)$ were carefully chosen to make the permutation matrix argument as efficient as possible. Therefore, we define an alternative version of the extended Pedersen commitment scheme with the commitment computed as $\hat{\mathfrak{c}} = \sum a_i [\hat{P}_i(\chi)]_1 + r[\hat{\varrho}]_1$ for trapdoor values $(\chi, \hat{\varrho})$ and randomizer r. Here, $\hat{P}_i(X)$ are well-chosen polynomials that satisfy a small number of requirements, including that $\{P_i(X)\hat{P}_j(X)\}_{1 \le i,j \le n}$ is linearly independent.

Before going on, we obviously need an efficient argument (that we call, following [14], a *same-message argument*) to show that $\mathfrak{c}$ and $\hat{\mathfrak{c}}$ commit to the same vector $\boldsymbol{a}$ (and use the same randomness r) while using different shrinking commitment schemes. We first write down the objective of this argument as the requirement that $\left(\begin{smallmatrix} \boldsymbol{a} \\ r \end{smallmatrix}\right)$ belongs to a subspace generated by a certain matrix $\boldsymbol{M}$. After doing that, we use the quasi-adaptive NIZK (QANIZK, [28,29]) argument of Kiltz and Wee (EUROCRYPT 2015, [30]) for linear subspaces to construct an efficient same-message argument. Since we additionally need it to be knowledge-sound, we give a proof in GBGM.

The new consistency argument is similar to but again more efficient than the consistency arguments of previous pairing-based shuffles. Here, we crucially use the fact that neither the permutation matrix argument nor the same-message argument add "too many" $\mathbb{G}_2$ elements to the CRS. Hence, while the Groth-Lu and FL shuffle arguments require two consistency verification equations, for us it suffices to only have one. (The Lipmaa-Zhang [34] and FLZ shuffle arguments have only one consistency verification equation, but this was compensated by using a non-standard cryptosystem with ciphertexts of length 6.)

In fact, we generalize the consistency argument to prove that given a committed matrix $\boldsymbol{E}$ and two tuples of ciphertexts $\mathfrak{M}'$ and $\mathfrak{M}$, it holds that

$\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{M}') = \boldsymbol{E} \cdot \mathsf{Dec}_{\mathsf{sk}}(\mathfrak{M})$. Moreover, we prove that the consistency argument is culpably sound [25,26] under a suitable KerMDH [36] assumption, and we prove that the concrete KerMDH assumption holds in the GBGM.

Finally, we will give a standard (i.e., non-culpable) soundness proof for the full shuffle argument, assuming that the used commitment scheme is computationally binding, the same-message argument and the permutation matrix argument are knowledge-sound, and the consistency argument is culpably sound. Additionally, as in the FLZ shuffle argument, we use batching techniques [4] to speed up verification time. However, we use batching in a more aggressive manner than in the FLZ shuffle argument.

Efficiency Comparison. We provide the first implementation of a pairing-based shuffle argument. Our implementation is built on top of the freely available libsnark library, [6]. In fact, we implement two versions of the new shuffle argument, where in the second version we switch the roles of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. In the first case we get better overall prover's computation, while in the second case we get the most efficient online computation for both prover and verifier, and overall the most efficient verifier.

Tbl. 1 shows a comparison between both versions of the new shuffle argument and prior state of the art CRS-based shuffle arguments with either the best prover's computational complexity or best verifier's computational complexity. Hence, we for instance do not include in this comparison table prior work by Groth and Lu [25] or Lipmaa and Zhang [34], since their shuffle arguments are slower than [14] and [15] in both prover's and verifier's computation. We also do not include the shuffle argument of González and Ràfols [20] since it has quadratic CRS length. In each row, the argument with best efficiency or best security property is highlighted.

Units (the main parameter) are defined in Tbl. 3 in Sect. 7. One should compare the number of units, which is a weighted sum of different exponentiations and pairings, and hence takes into account the fact that (say) computations in $\mathbb{G}_1$ and $\mathbb{G}_2$ take different time. Moreover, this table counts separately the number of general exponentiations, multi-exponentiations, and fixed-base exponentiations, since the last two can be much more efficient than general exponentiations. We take this into account by using different unit values for these three types of exponentiations, see Tbl. 3 for the number of units required for each type of operation. Note that we use our implementation of each operation in libsnark to compute the number of units.

Tbl. 2 gives the running time of the new shuffle argument (without and with switching the groups) on our test machine. As seen from this table, our preliminary implementation enables one to prove a shuffle argument in less than 1 minute and verify it in less than 1.5 minutes for $n = 100\,000$. After switching the groups, the prover's online computation takes less than 15 seconds and online verification takes less than 3 minutes for $n = 300\,000$. This means that the new shuffle argument is actually ready to be used in practice. Sect. 7 provides more

Table 1. A comparison of the new NIZK shuffle argument and prior work by Fauzi and Lipmaa (FL, [14]), and Fauzi, Lipmaa and Zając (FLZ, [15]). We include shuffling itself to the efficiency analysis of communication and prover's computation.

	FL	FLZ	Current Work	Current Work ($\mathbb{G}_1/\mathbb{G}_2$ switched)
$\lvert\mathrm{CRS}\rvert$ in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$	$(6n+8, 2n+8, 1)$	$(2n+6, n+7, 1)$	$(4n+7, n+7, 1)$	$(n+7, 4n+7, 1)$
Communication	$(7n+2, 2n, 0)$	$(5n+1, 4n+2, 0)$	$(4n-1, 3n+1, 0)$	$(3n+1, 4n-1, 0)$
Prover's computation				
Exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(2n-1, 0)$	$(2n, 0)$	$(n, 0)$	$(0, n)$
Fb-exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(8n-2, 2n-2)$	$(4n-1, 4n-1)$	$(3n-1, 3n-1)$	$(3n-1, 3n-1)$
M. exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(6n+6, 2n+2)$	$(3n+3, 5n+5)$	$(n+1, 2n+2)$	$(2n+2, n+1)$
Units	4.84	5.34	2.87	4.25
Prover's online computation				
M. exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(2n+2, 0)$	$(3n+3, 3n+3)$	$(0, 2n+2)$	$(2n+2, 0)$
Units	0.26	1.2	0.54	0.26
Verifier's computation				
Exp. in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$	$(0, 0, 0)$	$(7n+6, 7, 1)$	$(n, 2n+3, 1)$	$(2n+3, n, 1)$
M. exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(0, 0)$	$(4n, 3n)$	$(4n-4, 0)$	$(0, 4n-4)$
Pairing product	$18n+6$	$3n+6$	$3n+6$	$3n+6$
Units	38.52	14.75	12.98	12.02
Verifier's online computation				
Exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(0, 0)$	$(6n+3, 3)$	$(0, 2n+1)$	$(2n+1, 0)$
M. exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(0, 0)$	$(3n, 3n)$	$(0, 0)$	$(0, 0)$
Pairing product	$8n+4$	$2n+3$	$2n+1$	$2n+1$
Units	17.12	11.48	9.32	6.28
Lifted encryption	No	Yes	No	No
Soundness	Culpable	White-box	Full	Full

information about implementation, including the definition of units and data about the test machine.

2 Preliminaries

Let S_n be the symmetric group on n elements, i.e., all elements of the group are permutations on n elements. All vectors will be by default column vectors. By $\boldsymbol{a} = (a_i)_{i=1}^n$ we denote column vectors and by $\boldsymbol{a} = (a_1, \dots, a_n)$ we denote row vectors. For a matrix $\boldsymbol{A}$, $\boldsymbol{A}_i$ is its ith row vector and $\boldsymbol{A}^{(i)}$ is its ith column vector. A Boolean $n \times n$ matrix $\boldsymbol{A}$ is a permutation matrix representing $\sigma \in S_n$, when $A_{ij} = 1$ iff $\sigma(i) = j$. Clearly, in this case $\boldsymbol{A}\boldsymbol{x} = (x_{\sigma(i)})_{i=1}^n$ for any vector $\boldsymbol{x}$.

For a field $\mathbb{F}$, let $\mathbb{F}[\boldsymbol{X}]$ be the ring of multivariate polynomials over $\mathbb{F}$, and let $\mathbb{F}[\boldsymbol{X}^{\pm 1}]$ be the ring of multivariate Laurent polynomials over $\mathbb{F}$. For any (Laurent) polynomials $f_i(X)$, $i \in [1\,..\,n]$, we denote $\boldsymbol{f}(X) = (f_i(X))_{i=1}^n$.

Let $(\omega_i)_{i=1}^{n+1}$ be $n+1$ different values in $\mathbb{Z}_q$. For example, one can define $\omega_i = i$. Define the following polynomials:

- $Z(X) = \prod_{i=1}^{n+1}(X - \omega_i)$: the unique degree $n+1$ monic polynomial such that $Z(\omega_i) = 0$ for all $i \in [1\,..\,n]$.

Table 2. Efficiency of the shuffle implementation (in minutes and seconds) using the libsnark library: the original argument (left) and the one with switched groups (right), for various values of n.

	10,000	100,000	300,000
CRS generation	1.7s	13.4s	37.0s
Prover	6.4s	56.7s	2m38.3s
Prover (online)	1.1s	10.2s	32.3s
Verifier	8.5s	1m27.8s	5m18.8s
Verifier (online)	5.7s	1m0.5s	3m29s

	10,000	100,000	300,000
CRS generation	2.6s	20.5s	55.0s
Prover	9.1s	1m24.0s	4m2.4s
Prover (online)	0.5s	4.1s	13.5s
Verifier	8.3s	1m22.0s	4m49.2s
Verifier (online)	5.0s	49.9s	2m55.5s

- $\ell_i(X) = \prod_{j=1, j\neq i}^{n+1} \frac{X - \omega_j}{\omega_i - \omega_j}$: the ith Lagrange basis polynomial, i.e., the unique degree n polynomial such that $\ell_i(\omega_i) = 1$ and $\ell_i(\omega_j) = 0$ for $j \neq i$.

Cryptography. Let κ be the security parameter; intuitively it should be difficult to break a hardness assumption or a protocol in time $O(2^\kappa)$. If $f(\kappa)$ is a negligible function then we write $f(\kappa) \approx_\kappa 0$. We use type-III, asymmetric, pairings [17]. Assume we use a secure bilinear group generator BG that on input (1^κ) returns $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathfrak{g}_1, \mathfrak{g}_2, \bullet)$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three groups of prime order q, $\bullet : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, $\mathfrak{g}_1$ generates $\mathbb{G}_1$, $\mathfrak{g}_2$ generates $\mathbb{G}_2$, and $\mathfrak{g}_T = \mathfrak{g}_1 \bullet \mathfrak{g}_2$ generates $\mathbb{G}_T$. It is required that $\bullet$ is efficiently computable, bilinear, and non-degenerate.

To implement pairing-based cryptography, we use the libsnark library [6] which currently provides (asymmetric) Ate pairings [27] over Barreto-Naehrig curves [2,37] with 256 bit primes. Due to recent advances in computing discrete logarithms [31] this curve does not guarantee 128 bits of security, but still roughly achieves 100 bits of security [1].

Within this paper, we use additive notation combined with the bracket notation [13] and denote the elements of $\mathbb{G}_z$, $z \in \{1, 2, T\}$, as in $[a]_z$ (even if a is unknown). Alternatively, we denote group elements by using the Fraktur font as in $\mathfrak{b}$. We assume that $\cdot$ has higher precedence than $\bullet$; for example, $b\mathfrak{a} \bullet \mathfrak{c} = (b\mathfrak{a}) \bullet \mathfrak{c}$; while this is not important mathematically, it makes a difference in implementation since exponentiation in $\mathbb{G}_1$ is cheaper than in $\mathbb{G}_T$. In this notation, we write the generator of $\mathbb{G}_z$ as $\mathfrak{g}_z = [1]_z$ for $z \in \{1, 2, T\}$. Hence, $[a]_z = a\mathfrak{g}_z$, so the bilinear property can be written as $[a]_1 \bullet [b]_2 = (ab)[1]_T = [ab]_T$.

We freely combine additive notation with vector notation, by defining say $[a_1, \dots, a_s]_z = ([a_1]_z, \dots, [a_s]_z)$, and $[\boldsymbol{A}]_1 \bullet [\boldsymbol{B}]_2 = [\boldsymbol{AB}]_T$ for matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of compatible size. We sometimes misuse the notation and, say, write $[\boldsymbol{A}]_2\boldsymbol{B}$ instead of the more cumbersome $(\boldsymbol{B}^\top [\boldsymbol{A}]_2^\top)^\top$. Hence, if $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ have compatible dimensions, then $(\boldsymbol{B}^\top [\boldsymbol{A}]_1^\top)^\top \bullet [\boldsymbol{C}]_2 = [\boldsymbol{A}]_1 \boldsymbol{B} \bullet [\boldsymbol{C}]_2 = [\boldsymbol{A}]_1 \bullet \boldsymbol{B}[\boldsymbol{C}]_2$.

Recall that a distribution $\mathcal{D}_{par}$ that outputs matrices of group elements $[\boldsymbol{M}]_1 \in \mathbb{G}_1^{n \times t}$ is *witness-sampleable* [28], if there exists a distribution $\mathcal{D}'_{par}$ that outputs matrices of integers $\boldsymbol{M}' \in \mathbb{Z}_q^{n \times t}$, such that $[\boldsymbol{M}]_1$ and $[\boldsymbol{M}']_1$ have the same distribution.

We use the Elgamal cryptosystem [12] $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ in group $\mathbb{G}_2$. In Elgamal, the key generator $\mathsf{Gen}(1^\kappa)$ chooses a secret key $\mathsf{sk} \leftarrow_r \mathbb{Z}_q$ and a public key $\mathsf{pk} \leftarrow [1, \mathsf{sk}]_2$ for a generator $[1]_2$ of $\mathbb{G}_2$ fixed by BG. The encryption algorithm sets $\mathsf{Enc}_{\mathsf{pk}}(\mathfrak{m}; r) = ([0]_2, \mathfrak{m}) + r \cdot \mathsf{pk}$ for $\mathfrak{m} \in \mathbb{G}_2$ and $r \leftarrow_r \mathbb{Z}_q$. The decryption algorithm sets $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{M}_1, \mathfrak{M}_2) = \mathfrak{M}_2 - \mathsf{sk} \cdot \mathfrak{M}_1$. Note that $\mathsf{Enc}_{\mathsf{pk}}(\mathfrak{o}; r) = r \cdot \mathsf{pk}$. The Elgamal cryptosystem is blindable, with $\mathsf{Enc}_{\mathsf{pk}}(\mathfrak{m}; r_1) + \mathsf{Enc}_{\mathsf{pk}}(\mathfrak{o}; r_2) = \mathsf{Enc}_{\mathsf{pk}}(\mathfrak{m}; r_1 + r_2)$. Clearly, if r_2 is uniformly random, then $r_1 + r_2$ is also uniformly random.

Finally, for $[a]_1 \in \mathbb{G}_1$, and $[\boldsymbol{b}]_2 \in \mathbb{G}_2^{1 \times 2}$, let $[a]_1 \circ [\boldsymbol{b}]_2 := [a \cdot \boldsymbol{b}]_T = [a \cdot b_1, a \cdot b_2]_T \in \mathbb{G}_T^{1 \times 2}$. Analogously, for $[\boldsymbol{a}]_1 \in \mathbb{G}_1^n$, and $[\boldsymbol{B}]_2 \in \mathbb{G}_2^{n \times 2}$, let $[\boldsymbol{a}]_1^\top \circ [\boldsymbol{B}]_2 := \sum_{i=1}^n [a_i]_1 \circ [\boldsymbol{B}_i]_2 \in \mathbb{G}_T^2$. Intuitively, here $[\boldsymbol{b}]_2$ is an Elgamal ciphertext and $[\boldsymbol{B}]_2$ is a vector of Elgamal ciphertexts.

Kernel Matrix Assumptions [36]. Let $k \in \mathbb{N}$. We call $\mathcal{D}_{k+d,k}$ a *matrix distribution* [13] if it outputs matrices in $\mathbb{Z}_q^{(k+d) \times k}$ of full rank k in polynomial time. W.l.o.g., we assume that the first k rows of $\boldsymbol{A} \leftarrow \mathcal{D}_{k+d,k}$ form an invertible matrix. We denote $\mathcal{D}_{k+1,k}$ as $\mathcal{D}_k$.

Let $\mathcal{D}_{k+d,k}$ be a matrix distribution and $z \in \{1, 2\}$. The *$\mathcal{D}_{k+d,k}$-KerMDH assumption* [36] holds in $\mathbb{G}_z$ relative to algorithm BG, if for all probabilistic polynomial-time $\mathcal{A}$,

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa), \boldsymbol{M} \leftarrow_r \mathcal{D}_{k+d,k}, [\boldsymbol{c}]_{3-z} \leftarrow \mathcal{A}(\mathsf{gk}, [\boldsymbol{M}]_z) : \\ \boldsymbol{M}^\top \boldsymbol{c} = \boldsymbol{0} \wedge \boldsymbol{c} \neq \boldsymbol{0}\end{array}\right] \approx_\kappa 0 \enspace .$$

By varying the distribution $\mathcal{D}_{k+d,k}$, one can obtain various assumptions (such as the SP assumption of Groth and Lu [25] or the PSP assumption of Fauzi and Lipmaa [14]) needed by some previous shuffle arguments.

Since we use KerMDH to prove soundness of subarguments of the shuffle argument, we define a version of this assumption with an auxiliary input that corresponds to the CRS of the shuffle argument. We formalize it by defining a *valid CRS distribution* $\mathcal{D}_{n+d,d}$ to be a joint distribution of an $(n + d) \times d$ matrix distribution and a distribution of auxiliary inputs aux, such that

1. aux contains only elements of the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$,
2. $\mathcal{D}_{n+d,d}$ outputs as a trapdoor td all the used random coins from $\mathbb{Z}_q$.

We denote this as $(\boldsymbol{M}, \mathsf{aux}; \mathsf{td}) \leftarrow_r \mathcal{D}_{n+d,d}$. We prove the culpable soundness of the consistency argument under a variant of the KerMDH assumption that allows for an auxiliary input.

Definition 1 (KerMDH with an auxiliary input). *Let $\mathcal{D}_{n+d,d}$ be a valid CRS distribution. The $\mathcal{D}_{n+d,d}$-KerMDH assumption with an auxiliary input holds in $\mathbb{G}_z$, $z \in \{1, 2\}$, relative to algorithm* BG, *if for all probabilistic polynomial-time $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa), ([\boldsymbol{M}]_z, \mathsf{aux}) \leftarrow_r \mathcal{D}_{n+d,d}, [\boldsymbol{c}]_{3-z} \leftarrow \mathcal{A}(\mathsf{gk}, [\boldsymbol{M}]_z, \mathsf{aux}) : \\ \boldsymbol{M}^\top \boldsymbol{c} = \boldsymbol{0} \wedge \boldsymbol{c} \neq \boldsymbol{0}\end{array}\right] \approx_\kappa 0 \enspace .$$

Commitment Schemes. A (pairing-based) trapdoor commitment scheme is a pair of efficient algorithms $(\mathsf{K}, \mathsf{com})$, where $\mathsf{K}(\mathsf{gk})$ (for $\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa)$) outputs a commitment key ck and a trapdoor td, and the commitment algorithm outputs $\mathfrak{a} \leftarrow \mathsf{com}(\mathsf{gk}, \mathsf{ck}, \boldsymbol{a}; r)$. A commitment scheme is *computationally binding* if for any ck output by K, it is computationally infeasible to find $(\boldsymbol{a}, r) \neq (\boldsymbol{a}', r)$, such that $\mathsf{com}(\mathsf{gk}, \mathsf{ck}, \boldsymbol{a}; r) = \mathsf{com}(\mathsf{gk}, \mathsf{ck}, \boldsymbol{a}'; r')$. A commitment scheme is *perfectly hiding* if for any ck output by K, the distribution of the output of $\mathsf{com}(\mathsf{gk}, \mathsf{ck}, \boldsymbol{a}; r)$ does not depend on $\boldsymbol{a}$, assuming that r is uniformly random. A commitment scheme is *trapdoor*, if given access to td it is trivial to open the commitment to any value.

Throughout this paper, we use the following trapdoor commitment scheme, first implicitly used by Groth [24]. Fix some linearly independent polynomials $P_i(X)$. Let $\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa)$, and $\mathsf{td} = (\chi, \varrho) \leftarrow_r \mathbb{Z}_q^2$. Denote $\boldsymbol{P} = (P_i(\chi))_{i=1}^n$. Let $\mathsf{ck} \leftarrow [\begin{smallmatrix}\boldsymbol{P} \\ \varrho\end{smallmatrix}]_1$, and $\mathsf{com}(\mathsf{gk}, \mathsf{ck}, \boldsymbol{a}; r) := (\begin{smallmatrix}\boldsymbol{a} \\ r\end{smallmatrix})^\top \cdot \mathsf{ck} = [\sum_{i=1}^n a_i P_i(\chi) + r\varrho]_1$. We will call it the $((P_i(X))_{i=1}^n, X_\varrho)$*-commitment scheme.*

Several variants of this commitment scheme are known to be perfectly hiding and computational binding under a suitable computational assumption [19,23,32]. Since this commitment scheme is a variant of the extended Pedersen commitment scheme, it can be proven to be computationally binding under a suitable KerMDH assumption [36].

Theorem 1. *Let* $\mathcal{D}_n^{((P_i(X))_{i=1}^n, X_\varrho)} = \{[\begin{smallmatrix}\boldsymbol{P} \\ \varrho\end{smallmatrix}]_1 : (\chi, \varrho) \leftarrow_r \mathbb{Z}_q \times \mathbb{Z}_q^*\}$ *be the distribution of* ck *in this commitment scheme. The* $((P_i(X))_{i=1}^n, X_\varrho)$*-commitment scheme is perfectly hiding, and computationally binding under the* $\mathcal{D}_n^{((P_i(X))_{i=1}^n, X_\varrho)}$*-KerMDH assumption. It is trapdoor with* $\mathsf{td} = (\chi, \varrho)$.

Proof (Sketch). Given two different openings $(\boldsymbol{a}, r)$ and $(\boldsymbol{a}', r')$ to a commitment, $(\boldsymbol{a} - \boldsymbol{a}', r - r')$ is a solution to the KerMDH problem. Perfect hiding is obvious, since ϱ is not 0, and hence $r[\varrho]_1$ is uniformly random. Given $(\boldsymbol{a}, r)$, one can open $\mathsf{com}(\mathsf{gk}, \mathsf{ck}, \boldsymbol{a}; r)$ to $(\boldsymbol{a}', r')$ by taking $r' = r + (\sum_{i=1}^n (a_i - a_i') P_i(\chi))/\varrho$. □

Generic Bilinear Group Model. A *generic* algorithm uses only generic group operations to create and manipulate group elements. Shoup [39] formalized it by giving algorithms access to random injective encodings $\langle\langle \mathfrak{a} \rangle\rangle$ instead of real group elements $\mathfrak{a}$. Maurer [35] considered a different formalization, where the group elements are given in memory cells, and the adversary is given access to the address (but not the contents) of the cells. For simplicity, let's consider Maurer's formalization. The memory cells initially have some input (in our case, the random values generated by the CRS generator together with other group elements in the CRS). The generic group operations are handled through an oracle that on input $(op, i_1, \dots, i_n)$, where op is a generic group operation and i_j are addresses, first checks that memory cells in i_j have compatible type $z \in \{1, 2, T\}$ (e.g., if $op = +$, then i_1 and i_2 must belong to the same group $\mathbb{G}_z$), performs op on inputs $(i_1, \dots, i_n)$, stores the output in a new memory cell, and then returns the address of this cell. In the generic bilinear group model

(GBGM), the oracle can also compute the pairing $\bullet$. In addition, the oracle can answer equality queries.

We will use the following, slightly less formal way of thinking about the GBGM. Each memory cell has an implicit polynomial attached to it. A random value generated by the CRS generator is assigned a new indeterminate. If an $(+, i_1, i_2)$ (resp., $(\bullet, i_1, i_2)$) input is given to the oracle that successfully returns address i_3, then the implicit polynomial attached to i_3 will be the sum (resp., product) of implicit polynomials attached to i_1 and i_2. Importantly, since one can only add together elements from the same group (or, from the memory cells with the same type), this means that any attached polynomials with type 1 or 2 can only depend on the elements of the CRS that belong to the same group.

A generic algorithm does not know the values of the indeterminates (for her they really are indeterminates), but she will know all attached polynomials. At any moment, one can ask the oracle for an equality test between two memory cells i_1 and i_2. A generic algorithm is considered to be successful if she makes — in polynomial time — an equality test query $(=, i_1, i_2)$ that returns success (the entries are equal) but i_1 and i_2 have different polynomials $F_1(\boldsymbol{X})$ and $F_2(\boldsymbol{X})$ attached to them. This is motivated by the Schwartz-Zippel lemma [38,40] that states that if $F_1(\boldsymbol{X}) \neq F_2(\boldsymbol{X})$ as a polynomial, then there is negligible probability that $F_1(\boldsymbol{\chi}) = F_2(\boldsymbol{\chi})$ for uniformly random $\boldsymbol{\chi}$.

Zero Knowledge. Let $\mathcal{R} = \{(u, w)\}$ be an efficiently computable binary relation with $|w| = \mathsf{poly}(|u|)$. Here, u is a statement, and w is a witness. Let $\mathcal{L} = \{u : \exists w, (u, w) \in \mathcal{R}\}$ be an **NP**-language. Let $n = |u|$ be the input length. For fixed n, we have a relation $\mathcal{R}_n$ and a language $\mathcal{L}_n$. Since we argue about group elements, both $\mathcal{L}_n$ and $\mathcal{R}_n$ are group-dependent and thus we add gk (output by BG) as an input to $\mathcal{L}_n$ and $\mathcal{R}_n$. Let $\mathcal{R}_n(\mathsf{gk}) := \{(u, w) : (\mathsf{gk}, u, w) \in \mathcal{R}_n\}$.

A *non-interactive argument* for a group-dependent relation family $\mathcal{R}$ consists of four probabilistic polynomial-time algorithms: a setup algorithm BG, a common reference string (CRS) generator K, a prover P, and a verifier V. Within this paper, BG is always the bilinear group generator that outputs $\mathsf{gk} \leftarrow (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathfrak{g}_1, \mathfrak{g}_2, \bullet)$. For $\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa)$ and $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}(\mathsf{gk}, n)$ (where n is input length), $\mathsf{P}(\mathsf{gk}, \mathsf{crs}, u, w)$ produces an argument π, and $\mathsf{V}(\mathsf{gk}, \mathsf{crs}, u, \pi)$ outputs either 1 (accept) or 0 (reject). The verifier may be probabilistic, to speed up verification time by the use of batching techniques [4].

A non-interactive argument Ψ is *perfectly complete*, if for all $n = \mathsf{poly}(\kappa)$,

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}(\mathsf{gk}, n), (u, w) \leftarrow \mathcal{R}_n(\mathsf{gk}) : \\ \mathsf{V}(\mathsf{gk}, \mathsf{crs}, u, \mathsf{P}(\mathsf{gk}, \mathsf{crs}, u, w)) = 1\end{array}\right] = 1 \enspace .$$

Ψ is adaptively *computationally sound* for $\mathcal{L}$, if for all $n = \mathsf{poly}(\kappa)$ and all non-uniform probabilistic polynomial-time adversaries $\mathcal{A}$,

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}(\mathsf{gk}, n), \\ (u, \pi) \leftarrow \mathcal{A}(\mathsf{gk}, \mathsf{crs}) : (\mathsf{gk}, u) \not\in \mathcal{L}_n \wedge \mathsf{V}(\mathsf{gk}, \mathsf{crs}, u, \pi) = 1\end{array}\right] \approx_\kappa 0 \enspace .$$

We recall that in situations where the inputs have been committed to using a computationally binding trapdoor commitment scheme, the notion of computational soundness does not make sense (since the commitments could be to any input messages). Instead, one should either prove culpable soundness or knowledge-soundness.

Ψ is adaptively *computationally culpably sound* [25,26] for $\mathcal{L}$ using a polynomial-time decidable binary relation $\mathcal{R}^{\mathsf{glt}} = \{\mathcal{R}^{\mathsf{glt}}_n\}$ consisting of elements from $\bar{\mathcal{L}}$ and witnesses w^{glt}, if for all $n = \mathsf{poly}(\kappa)$ and all non-uniform probabilistic polynomial-time adversaries $\mathcal{A}$,

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}(\mathsf{gk}, n), (u, \pi, w^{\mathsf{glt}}) \leftarrow \mathcal{A}(\mathsf{gk}, \mathsf{crs}) : \\ (\mathsf{gk}, u, w^{\mathsf{glt}}) \in \mathcal{R}^{\mathsf{glt}}_n \wedge \mathsf{V}(\mathsf{gk}, \mathsf{crs}, u, \pi) = 1\end{array}\right] \approx_\kappa 0 \enspace .$$

For algorithms $\mathcal{A}$ and $\mathsf{Ext}_\mathcal{A}$, we write $(y; y') \leftarrow (\mathcal{A}||\mathsf{Ext}_\mathcal{A})(\chi)$ if $\mathcal{A}$ on input χ outputs y, and $\mathsf{Ext}_\mathcal{A}$ on the same input (including the random tape of $\mathcal{A}$) outputs y'. Ψ is *knowledge-sound*, if for all $n = \mathsf{poly}(\kappa)$ and all non-uniform probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a non-uniform probabilistic polynomial-time extractor $\mathsf{Ext}_\mathcal{A}$, such that for every auxiliary input $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\kappa)}$,

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}(\mathsf{gk}, n), ((u, \pi); w) \leftarrow (\mathcal{A}||\mathsf{Ext}_\mathcal{A})(\mathsf{crs}, \mathsf{aux}) : \\ (\mathsf{gk}, u, w) \not\in \mathcal{R}_n \wedge \mathsf{V}(\mathsf{gk}, \mathsf{crs}, u, \pi) = 1\end{array}\right] \approx_\kappa 0 \enspace .$$

Here, aux can be seen as the common auxiliary input to $\mathcal{A}$ and $\mathsf{Ext}_\mathcal{A}$ that is generated by using benign auxiliary input generation [7].

Ψ is *perfectly (composable) zero-knowledge [21]*, if there exists a probabilistic polynomial-time simulator S, such that for all stateful non-uniform probabilistic adversaries $\mathcal{A}$ and $n = \mathsf{poly}(\kappa)$, $\varepsilon_0 = \varepsilon_1$, where

$$\varepsilon_b := \Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}(\mathsf{gk}, n), (u, w) \leftarrow \mathcal{A}(\mathsf{gk}, \mathsf{crs}), \\ \mathbf{if}\ b = 0\ \mathbf{then}\ \pi \leftarrow \mathsf{P}(\mathsf{gk}, \mathsf{crs}, u, w)\ \mathbf{else}\ \pi \leftarrow \mathsf{S}(\mathsf{gk}, \mathsf{crs}, u, \mathsf{td})\ \mathbf{endif} : \\ (\mathsf{gk}, u, w) \in \mathcal{R}_n \wedge \mathcal{A}(\mathsf{gk}, \mathsf{crs}, u, \pi) = 1\end{array}\right] \enspace .$$

Shuffle Argument. In a (pairing-based) shuffle argument [25], the prover aims to convince the verifier that, given system parameters gk output by BG, a public key pk, and two tuples of ciphertexts $\mathfrak{M}$ and $\mathfrak{M}'$, the second tuple is a permutation of rerandomized versions of the first. More precisely, we will construct a shuffle argument that is sound with respect to the following relation:

$$\mathcal{R}_{sh} = \left\{\begin{array}{l}((\mathsf{gk}, \mathsf{pk}, \mathfrak{M}, \mathfrak{M}'), (\sigma, \boldsymbol{r})) : \sigma \in S_n \wedge \\ \boldsymbol{r} \in \mathbb{Z}_q^n \wedge \left(\forall i : \mathfrak{M}'_i = \mathfrak{M}_{\sigma(i)} + \mathsf{Enc}_{\mathsf{pk}}(\mathsf{o}; r_i)\right)\end{array}\right\} \enspace .$$

A number of pairing-based shuffle arguments have been proposed in the literature, [25,34,14,15,20].

3 New Shuffle Argument

Intuitively, in the new shuffle argument the prover first commits to the permutation σ (or more precisely, to the corresponding permutation matrix), then executes three subarguments (the same-message, the permutation matrix, and the consistency arguments). Each of the subarguments corresponds to one check performed by the verifier (see Prot. 2). However, since all subarguments use the same CRS, they are not independent. For example, the permutation matrix argument uses the $((P_i(X))_{i=1}^n, X_\varrho)$-commitment scheme and the consistency argument uses the $((\hat{P}_i(X))_{i=1}^n, X_{\hat{\varrho}})$-commitment scheme for different polynomials $(\hat{P}_i(X))_{i=1}^n$. (See Eq. (3) and Eq. (5) for the actual definition of $P_i(X)$ and $\hat{P}_i(X)$.) Both commitment schemes share a part of their trapdoor (χ), while the second part of the trapdoor is different (either ϱ or $\hat{\varrho}$). Moreover, the knowledge-soundness of the same-message argument is a prerequisite for the knowledge-soundness of the permutation matrix argument. The verifier recovers explicitly the commitment to the last row of the permutation matrix (this guarantees that the committed matrix is left stochastic), then verifies the three subarguments.

The full description of the new shuffle argument is given in Prot. 1 (the CRS generation and the prover) and in Prot. 2 (the verifier). The CRS has entries that allow to efficiently evaluate all subarguments, and hence also both commitment schemes. The CRS in Prot. 1 includes three substrings, crs_{sm}, crs_{pm}, and crs_{con}, that are used in the three subarguments. To prove and verify (say) the first subargument (the same-message argument), one needs access to crs_{sm}. However, the adversary of the same-message argument will get access to the full CRS. For the sake of exposition, the verifier's description in Prot. 2 does not include batching. In Sect. 6, we will explain how to speed up the verifier considerably by using batching techniques.

We will next briefly describe the subarguments. In Sect. 4, we will give more detailed descriptions of each subargument, and in Sect. 5, we will prove the security of the shuffle argument.

Same-Message Argument. Consider the subargument of the new shuffle argument where the verifier only computes $\mathfrak{a}_n$ and then performs the check on Step 4 of Prot. 2 for one concrete i. We will call it the *same-message argument* [14]. In Sect. 4.1 we motivate this name, by showing that if the same-message argument accepts, then the prover knows a message $\boldsymbol{a}$ and a randomizer r, such that $\mathfrak{a}_i = [\sum a_i P_i(\chi) + r\varrho]_1$ and $\hat{\mathfrak{a}}_i = [\sum a_i \hat{P}_i(\chi) + r\hat{\varrho}]_1$ both commit to $\boldsymbol{a}$ with randomizer r, by using respectively the $((P_i(X))_{i=1}^n, X_\varrho)$-commitment scheme and the $((\hat{P}_i(X))_{i=1}^n, X_{\hat{\varrho}})$-commitment scheme.

For the same-message argument to be knowledge-sound, we will require that $\{P_i(X)\}_{i=1}^n$ and $\{\hat{P}_i(X)\}_{i=1}^n$ are both linearly independent sets.

Permutation Matrix Argument. Consider the subargument of Prot. 1 and Prot. 2, where (i) the prover computes $\mathfrak{a}$ and π_{pm}, and (ii) the verifier computes

$\mathsf{K}(\mathsf{gk}, n)$**:** Generate random $(\chi, \beta, \hat{\beta}, \varrho, \hat{\varrho}, \mathsf{sk}) \leftarrow_r \mathbb{Z}_q^3 \times (\mathbb{Z}_q^*)^2 \times \mathbb{Z}_q$. Denote $\boldsymbol{P} = (P_i(\chi))_{i=1}^n$, $P_0 = P_0(\chi)$, and $\hat{\boldsymbol{P}} = (\hat{P}_i(\chi))_{i=1}^n$. Let $\mathsf{crs}_{sm} \leftarrow \left([(\beta P_i + \hat{\beta}\hat{P}_i)_{i=1}^n, \beta\varrho + \hat{\beta}\hat{\varrho}]_1, [\beta, \hat{\beta}]_2^\top\right)$,

$$\mathsf{crs}_{pm} \leftarrow \begin{pmatrix} [1, P_0, (((P_i + P_0)^2 - 1)/\varrho)_{i=1}^n, \sum_{i=1}^n P_i, \sum_{i=1}^n \hat{P}_i]_1, \\ [P_0, \sum_{i=1}^n P_i]_2, [1]_T \end{pmatrix} ,$$

$\mathsf{crs}_{con} \leftarrow [\begin{smallmatrix}\hat{\boldsymbol{P}} \\ \hat{\varrho}\end{smallmatrix}]_1$. Set $\mathsf{crs} \leftarrow \left(\mathsf{pk} = [1, \mathsf{sk}]_2, [\begin{smallmatrix}\boldsymbol{P} \\ \varrho\end{smallmatrix}]_1, [\begin{smallmatrix}\boldsymbol{P} \\ \varrho\end{smallmatrix}]_2, \mathsf{crs}_{sm}, \mathsf{crs}_{pm}, \mathsf{crs}_{con}\right)$. Set $\mathsf{td} \leftarrow (\chi, \hat{\varrho})$. Return $(\mathsf{crs}, \mathsf{td})$.

```
$\mathsf{P}(\mathsf{gk}, \mathsf{crs}, \mathfrak{M} \in \mathbb{G}_2^{n \times 2}; \sigma \in S_n, \boldsymbol{t} \in \mathbb{Z}_q^n)$:
    1. For $i = 1$ to $n - 1$: // commits to the permutation $\sigma$
       (a) $r_i \leftarrow_r \mathbb{Z}_q$; $\mathfrak{r}_i \leftarrow r_i[\varrho]_1$;
       (b) $\mathfrak{a}_i \leftarrow [P_{\sigma^{-1}(i)}]_1 + \mathfrak{r}_i$; $\mathfrak{b}_i \leftarrow [P_{\sigma^{-1}(i)}]_2 + r_i[\varrho]_2$; $\hat{\mathfrak{a}}_i \leftarrow [\hat{P}_{\sigma^{-1}(i)}]_1 + r_i[\hat{\varrho}]_1$;
    2. $\mathfrak{a}_n \leftarrow [\sum_{i=1}^n P_i]_1 - \sum_{j=1}^{n-1} \mathfrak{a}_j$; $\mathfrak{b}_n \leftarrow [\sum_{i=1}^n P_i]_2 - \sum_{j=1}^{n-1} \mathfrak{b}_j$;
    3. $\hat{\mathfrak{a}}_n \leftarrow [\sum_{i=1}^n \hat{P}_i]_1 - \sum_{j=1}^{n-1} \hat{\mathfrak{a}}_j$;
    4. $r_n \leftarrow -\sum_{i=1}^{n-1} r_i$; $\mathfrak{r}_n \leftarrow r_n[\varrho]_1$;
    5. For $i = 1$ to $n$:
       (a) $\mathfrak{d}_i \leftarrow [\beta P_{\sigma^{-1}(i)} + \hat{\beta}\hat{P}_{\sigma^{-1}(i)}]_1 + r_i[\beta\varrho + \hat{\beta}\hat{\varrho}]_1$;
       (b) $\mathfrak{c}_i \leftarrow r_i \cdot (2(\mathfrak{a}_i + [P_0]_1) - \mathfrak{r}_i) + [((P_{\sigma^{-1}(i)} + P_0)^2 - 1)/\varrho]_1$;
    6. $r_t \leftarrow_r \mathbb{Z}_q$; $\mathfrak{t} \leftarrow \boldsymbol{t}^\top[\hat{\boldsymbol{P}}]_1 + r_t[\hat{\varrho}]_1$;
    7. For $i = 1$ to $n$: $\mathfrak{t}_i' \leftarrow t_i \cdot \mathsf{pk}$;
    8. $\mathfrak{M}' \leftarrow (\mathfrak{M}_{\sigma(i)} + \mathfrak{t}_i')_{i=1}^n$; // Shuffling, online
    9. $\mathfrak{N} \leftarrow \boldsymbol{r}^\top \mathfrak{M} + r_t \cdot \mathsf{pk}$; // Online
    10. $\pi_{sm} \leftarrow \mathfrak{d}$; // Same-message argument
    11. $\pi_{pm} \leftarrow ((\mathfrak{b}_i)_{i=1}^{n-1}, \mathfrak{c})$; // Permutation matrix argument
    12. $\pi_{con} \leftarrow ((\hat{\mathfrak{a}}_i)_{i=1}^{n-1}, \mathfrak{t}, \mathfrak{N})$; // Consistency argument
    13. Return $\pi_{sh} \leftarrow (\mathfrak{M}', (\mathfrak{a}_j)_{j=1}^{n-1}, \pi_{sm}, \pi_{pm}, \pi_{con})$.
```

Protocol 1: The CRS generation and the prover of the new shuffle argument.

```
$\mathsf{V}(\mathsf{gk}, \mathsf{crs}, \mathfrak{M}; \mathfrak{M}', (\mathfrak{a}_j)_{j=1}^{n-1}, \pi_{sm}, \pi_{pm}, \pi_{con})$:
    1. Parse $(\pi_{sm}, \pi_{pm}, \pi_{con})$ as in the prover's Steps 11–12, abort if unsuccessful;
    2. Compute $\mathfrak{a}_n$, $\mathfrak{b}_n$, and $\hat{\mathfrak{a}}_n$ as in the prover's Steps 2 and 3;
    3. $\alpha \leftarrow_r \mathbb{Z}_q$;
    4. For $i = 1$ to $n$: check that $\mathfrak{d}_i \bullet [1]_2 \overset{?}{=} (\mathfrak{a}_i, \hat{\mathfrak{a}}_i) \bullet \begin{bmatrix} \beta \\ \hat{\beta} \end{bmatrix}_2$; // Same-message argument
    5. For $i = 1$ to $n$: check that // Permutation matrix argument
       $(\mathfrak{a}_i + \alpha[1]_1 + [P_0]_1) \bullet (\mathfrak{b}_i - \alpha[1]_2 + [P_0]_2) \overset{?}{=} \mathfrak{c}_i \bullet [\varrho]_2 + (1 - \alpha^2)[1]_T$;
    6. Check that // Consistency argument
       $[\hat{\boldsymbol{P}}]_1^\top \circ \mathfrak{M}' - \hat{\mathfrak{a}}^\top \circ \mathfrak{M} \overset{?}{=} \mathfrak{t} \circ \mathsf{pk} - [\hat{\varrho}]_1 \circ \mathfrak{N}$;
```

Protocol 2: The non-batched verifier of the new shuffle argument.

$\mathfrak{a}_n$ and then checks the verification equation on Step 5 of Prot. 2. We will call it the *permutation matrix argument*. In Sect. 4.2 we motivate this name, by proving in the GBGM that if the verifier accepts the permutation matrix argument, then either the prover knows how to open $(\mathfrak{a}_1, \ldots, \mathfrak{a}_n)$ as a $((P_i(X))_{i=1}^n, X_\varrho)$-commitment to a permutation matrix or we can break the same-message argument. For this, we first prove the security of a subargument of the permutation matrix argument — the unit vector argument [14] — where the verifier performs the verification Step 5 for exactly one i.

For the unit vector argument to be efficient, we need to make a specific choice of the polynomials $P_i(X)$ (see Eq. (3)). For the knowledge-soundness of the unit vector argument, we additionally need that $\{P_i(X)\}_{i=0}^n$ and $\{P_i(X)\}_{i=1}^n \cup \{1\}$ are linearly independent.

In [15], the verifer adds $[\alpha]_1 + [P_0]_1$ from the CRS to $\mathfrak{a}_i$ (and adds $[\alpha]_2 - [P_0]_2$ from the CRS to $\mathfrak{b}_i$), while in our case, the verifier samples α itself during verification. Due to this small change, we can make the CRS independent of α. (In fact, it suffices if the verifier chooses α once and then uses it at each verification.) This makes the CRS shorter, and also simplifies the latter soundness proof. For this optimization to be possible, one has to rely on the same-message argument (see Sect. 5).

Consistency Argument. Consider the subargument of the new shuffle argument where the prover only computes π_{con} and the verifier performs the check on Step 6 of Prot. 2. We will call it the *consistency argument*. In Sect. 4.3 we motivate this name, by showing that if $\hat{\mathfrak{a}}$ $(\{\hat{P}_i(X)\}, X_{\hat{\varrho}})$-commits to a permutation, then $\mathsf{Dec}(\mathfrak{M}'_i) = \mathsf{Dec}(\mathfrak{M}_{\sigma(i)})$ for the same permutation σ that the prover committed to earlier. We show that the new consistency argument is culpably sound under a (novel) variant of the KerMDH computational assumption [36] that we describe in Sect. 4.3. In particular, the KerMDH assumption has to hold even when the adversary is given access to the full CRS of the shuffle argument.

For the consistency argument to be sound (and in particular, for the KerMDH variant to be secure in the GBGM), we will require that $\{\hat{P}_i(X)\}_{i=1}^n$ and $\{P_i(X)\hat{P}_j(X)\}_{1 \leq i,j \leq n}$ are both linearly independent sets.

4 Subarguments

Several of the following knowledge-soundness proofs use the GBGM and therefore we will first give common background for those proofs. In the GBGM, the generic adversary in each proof has only access to generic group operations, pairings, and equality tests. However, she will have access to the full CRS of the new shuffle argument.

Let $\boldsymbol{\chi} = (\chi, \alpha, \beta, \hat{\beta}, \varrho, \hat{\varrho}, \mathsf{sk})$ be the tuple of all random values generated by either K or V. Note that since α is sampled by the verifier each time, for an adversary it is essentially an indeterminate. Thus, for the generic adversary each element of $\boldsymbol{\chi}$ will be an indeterminate. Let us denote the tuple of corresponding indeterminates by $\boldsymbol{X} = (X, X_\alpha, X_\beta, X_{\hat{\beta}}, X_\varrho, X_{\hat{\varrho}}, X_S)$.

The adversary is given oracle access to the full CRS. This means that for each element of the CRS, she knows the attached (Laurent) polynomial in $\boldsymbol{X}$.

E.g., for the CRS element $[\beta\varrho + \hat{\beta}\hat{\varrho}]_1$, she knows that the attached polynomial is $X_\beta X_\varrho + X_{\hat{\beta}} X_{\hat{\varrho}}$. Each element output by the adversary can hence be seen as a polynomial in $\boldsymbol{X}$. Moreover, if this element belongs to $\mathbb{G}_z$ for $z \in \{1, 2\}$, then this polynomial has to belong to the span of attached polynomials corresponding to the elements of CRS from the same group. Observing the definition of crs in Prot. 1, we see that this means that each $\mathbb{G}_z$ element output by the adversary must have an attached polynomial of the form $\mathsf{crs}_z(\boldsymbol{X}, T, t)$ for symbolic values T and t:

$$\begin{aligned} crs_1(\boldsymbol{X}, T, t) =& t(X) + T_0 P_0(X) + T_\varrho X_\varrho + T^\dagger(X) Z(X)/X_\varrho + T^*(X) + \\ & T_{\hat{\varrho}} X_{\hat{\varrho}} + \sum_{i=1}^{n} T_{\beta,i}(X_\beta P_i(X) + X_{\hat{\beta}} \hat{P}_i(X)) + T_{\beta\varrho}(X_\beta X_\varrho + X_{\hat{\beta}} X_{\hat{\varrho}}) \enspace , \\ crs_2(\boldsymbol{X}, T, t) =& t(X) + T_0 P_0(X) + T_\varrho X_\varrho + T_S X_S + T_\beta X_\beta + T_{\hat{\beta}} X_{\hat{\beta}} \enspace , \end{aligned}$$

where $T^\dagger(X)$ is in the span of $\{((P_i(X) + P_0(X))^2 - 1)/Z(X)\}_{i=1}^n$ (it will be a polynomial due to the definition of $P_i(X)$ and $P_0(X)$), $T^*(X)$ is in the span of $\{\hat{P}_i(X)\}_{i=1}^n$, and $t(X)$ is in the span of $\{P_i(X)\}_{i=1}^n$. (Here we use Lem. 1, given below, that states that $\{P_i(X)\}_{i=1}^n \cup \{1\}$ and $\{P_i(X)\}_{i=0}^n$ are two bases of degree-$\leq n$ polynomials.) We will follow the same notation in the rest of the paper. E.g., polynomials with a star (like $b^*(X)$) are in the span of $\{\hat{P}_i(X)\}_{i=1}^n$.

4.1 Same-Message Argument

For the sake of this argument, let $\boldsymbol{P}(X) = (P_i(X))_{i=1}^n$ and $\hat{\boldsymbol{P}}(X) = (\hat{P}_i(X))_{i=1}^n$ be two families of linearly independent polynomials. We do not specify the parameters X, X_ϱ and $X_{\hat{\varrho}}$ in the case when they take their canonical values χ, ϱ, and $\hat{\varrho}$.

In the *same-message argument*, the prover aims to prove that given $\mathfrak{a}, \hat{\mathfrak{a}} \in \mathbb{G}_1$, she knows $\boldsymbol{a}$ and r, such that $\left(\begin{smallmatrix}\mathfrak{a}\\ \hat{\mathfrak{a}}\end{smallmatrix}\right) = [\boldsymbol{M}]_1 \left(\begin{smallmatrix}\boldsymbol{a}\\ r\end{smallmatrix}\right)$ for

$$[\boldsymbol{M}]_1^\top := (\mathsf{ck}, \widehat{\mathsf{ck}}) = \begin{bmatrix} \boldsymbol{P} & \hat{\boldsymbol{P}} \\ \varrho & \hat{\varrho} \end{bmatrix}_1 \in \mathbb{G}_1^{(n+1)\times 2} \tag{1}$$

and $\mathsf{ck} = \left[\begin{smallmatrix}\boldsymbol{P}\\ \varrho\end{smallmatrix}\right]_1$ and $\widehat{\mathsf{ck}} = \left[\begin{smallmatrix}\hat{\boldsymbol{P}}\\ \hat{\varrho}\end{smallmatrix}\right]_1$. That is, $\mathfrak{a}$ and $\hat{\mathfrak{a}}$ are commitments to the same vector $\boldsymbol{a}$ with the same randomness r, but using the $((P_i(X))_{i=1}^n, X_\varrho)$-commitment scheme and the $((\hat{P}_i(X))_{i=1}^n, X_{\hat{\varrho}})$-commitment scheme correspondingly.

We construct the same-message argument by essentially using the (second) QANIZK argument of Kiltz and Wee [30] for the *linear subspace*

$$\mathcal{L}_{\boldsymbol{M}} = \{\left(\begin{smallmatrix}\mathfrak{a}\\ \hat{\mathfrak{a}}\end{smallmatrix}\right) : \exists \left(\begin{smallmatrix}\boldsymbol{a}\\ r\end{smallmatrix}\right) \in \mathbb{Z}_q^{n+1} : \left(\begin{smallmatrix}\mathfrak{a}\\ \hat{\mathfrak{a}}\end{smallmatrix}\right) = [\boldsymbol{M}]_1 \left(\begin{smallmatrix}\boldsymbol{a}\\ r\end{smallmatrix}\right)\} \enspace .$$

However, as we will see in the proof of the permutation matrix argument, we need knowledge-soundness of the same-message argument. Therefore, while we use exactly the QANIZK argument of Kiltz and Wee, we prove its knowledge-soundness in the GBGM: we show that if the verifier accepts then the prover knows a witness $\boldsymbol{w}$ such that $(\mathfrak{a}, \hat{\mathfrak{a}}) = \boldsymbol{w}^\top \cdot [\boldsymbol{M}]_1$. Moreover, we need it to stay knowledge-sound even when the adversary has access to an auxiliary input.

More precisely, denote by $\mathcal{D}^{sm}_{n,2}$ the distribution of matrices $\boldsymbol{M}$ in Eq. (1) given that $(\chi, \varrho, \hat{\varrho}) \leftarrow_r \mathbb{Z}_q \times (\mathbb{Z}_q^*)^2$. For $k \geq 1$, let $\mathcal{D}_k$ be a distribution such that the $\mathcal{D}_k$-KerMDH assumption holds. Clearly, $\mathcal{D}^{sm}_{n,2}$ is witness-sampleable.

For a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{(k+1)\times k}$, let $\bar{\boldsymbol{A}} \in \mathbb{Z}_q^{k \times k}$ denote the upper square matrix of $\boldsymbol{A}$. The same-message argument (i.e., the Kiltz-Wee QANIZK argument that $\left(\begin{smallmatrix}\mathfrak{a}\\ \hat{\mathfrak{a}}\end{smallmatrix}\right) = [\boldsymbol{M}]_1 \left(\begin{smallmatrix}\boldsymbol{a}\\ r\end{smallmatrix}\right)$) for witness-sampleable distributions is depicted as follows:

$\mathsf{K}_{sm}(\mathsf{gk}, [\boldsymbol{M}]_1 \in \mathbb{G}_1^{2\times(n+1)})$: $\boldsymbol{A} \leftarrow_r \mathcal{D}_k$; $\boldsymbol{K} \leftarrow_r \mathbb{Z}_q^{2\times k}$; $[\boldsymbol{Q}]_1 \leftarrow [\boldsymbol{M}]_1^\top \boldsymbol{K} \in \mathbb{G}_1^{(n+1)\times k}$; $\boldsymbol{C} \leftarrow \boldsymbol{K}\bar{\boldsymbol{A}} \in \mathbb{Z}_q^{2\times k}$; $\mathsf{crs}_{sm} \leftarrow ([\boldsymbol{Q}]_1, [\boldsymbol{C}]_2, [\bar{\boldsymbol{A}}]_2)$; $\mathsf{td}_{sm} \leftarrow \boldsymbol{K}$;
    Return $(\mathsf{crs}_{sm}, \mathsf{td}_{sm})$;
$\mathsf{P}_{sm}(\mathsf{gk}, \mathsf{crs}_{sm}, \left(\begin{smallmatrix}\mathfrak{a}\\ \hat{\mathfrak{a}}\end{smallmatrix}\right), \left(\begin{smallmatrix}\boldsymbol{a}\\ r\end{smallmatrix}\right))$: Return $\pi_{sm} \leftarrow \left(\begin{smallmatrix}\boldsymbol{a}\\ r\end{smallmatrix}\right)^\top [\boldsymbol{Q}]_1 \in \mathbb{G}_1^{1\times k}$;
$\mathsf{S}_{sm}(\mathsf{gk}, \mathsf{crs}_{sm}, \mathsf{td}_{sm}, \left(\begin{smallmatrix}\mathfrak{a}\\ \hat{\mathfrak{a}}\end{smallmatrix}\right))$: Return $\pi_{sm} \leftarrow \left(\begin{smallmatrix}\mathfrak{a}\\ \hat{\mathfrak{a}}\end{smallmatrix}\right)^\top \boldsymbol{K} \in \mathbb{G}_1^{1\times k}$;
$\mathsf{V}_{sm}(\mathsf{gk}, \mathsf{crs}_{sm}, \left(\begin{smallmatrix}\mathfrak{a}\\ \hat{\mathfrak{a}}\end{smallmatrix}\right), \pi_{sm})$: Check that $\pi_{sm} \bullet [\bar{\boldsymbol{A}}]_2 \stackrel{?}{=} \left(\begin{smallmatrix}\mathfrak{a}\\ \hat{\mathfrak{a}}\end{smallmatrix}\right)^\top \bullet [\boldsymbol{C}]_2$;

Clearly, the verification accepts since $\pi_{sm} \bullet [\bar{\boldsymbol{A}}]_2 = \left(\begin{smallmatrix}\boldsymbol{a}\\ r\end{smallmatrix}\right)^\top [\boldsymbol{Q}]_1 \bullet [\bar{\boldsymbol{A}}]_2 = \left(\begin{smallmatrix}\boldsymbol{a}\\ r\end{smallmatrix}\right)^\top [\boldsymbol{M}]_1^\top \boldsymbol{K} \bullet [\bar{\boldsymbol{A}}]_2 = \left(\begin{smallmatrix}\mathfrak{a}\\ \hat{\mathfrak{a}}\end{smallmatrix}\right)^\top \bullet [\boldsymbol{C}]_2$.

For the sake of efficiency, we will assume $k = 1$ and $\mathcal{D}_1 = \mathcal{L}_1 = \{\left(\begin{smallmatrix}1\\ a\end{smallmatrix}\right) : a \leftarrow_r \mathbb{Z}_q\}$. Then, $\bar{\boldsymbol{A}} = 1$, $\boldsymbol{K} = (\beta, \hat{\beta})^\top$, $\boldsymbol{Q} = (\beta P_1 + \hat{\beta}\hat{P}_1, \ldots, \beta P_n + \hat{\beta}\hat{P}_n, \beta\varrho + \hat{\beta}\hat{\varrho})^\top$, and $\boldsymbol{C} = \boldsymbol{K}$. Thus, crs_{sm} and td_{sm} are as in Prot. 1. In the case of a shuffle, $\left(\begin{smallmatrix}\boldsymbol{a}\\ r\end{smallmatrix}\right) = \left(\begin{smallmatrix}\boldsymbol{e}_{\sigma^{-1}(i)}\\ r\end{smallmatrix}\right)$, and thus $\pi_{sm} \leftarrow [\beta P_{\sigma^{-1}(i)} + \hat{\beta}\hat{P}_{\sigma^{-1}(i)}]_1 + r[\beta\varrho + \hat{\beta}\hat{\varrho}]_1$ as in Prot. 1. The verifier has to check that $\pi_{sm} \bullet [1]_2 = \mathfrak{a} \bullet [\beta]_2 + \hat{\mathfrak{a}} \bullet [\hat{\beta}]_2$, as in Prot. 2. The simulator, given the trapdoor $\mathsf{td}_{sm} = (\beta, \hat{\beta})^\top$, sets $\pi_{sm} \leftarrow \beta\mathfrak{a} + \hat{\beta}\hat{\mathfrak{a}}$.

Theorem 2. *Assume* $\mathsf{crs} = (\mathsf{crs}_{sm}, \mathsf{aux})$ *where* aux *does not depend on* β *or* $\hat{\beta}$. *The same-message argument has perfect zero knowledge for* $\mathcal{L}_{\boldsymbol{M}}$. *It has adaptive knowledge-soundness in the GBGM.*

Proof. Zero Knowledge: follows from $\pi_{sm} = [\boldsymbol{Q}^\top \boldsymbol{\chi}]_1 = [\boldsymbol{K}^\top \boldsymbol{M}^\top \boldsymbol{\chi}]_1 = [\boldsymbol{K}^\top \boldsymbol{y}]_1$.

Knowledge-soundness: In the generic group model, the adversary knows polynomials $A(\boldsymbol{X}) = crs_1(\boldsymbol{X}, A, a)$, $\hat{A}(\boldsymbol{X}) = crs_1(\boldsymbol{X}, \hat{A}, \hat{a})$, and $\pi(\boldsymbol{X}) = crs_1(\boldsymbol{X}, \Pi, \pi)$, such that $\mathfrak{a} = [A(\boldsymbol{\chi})]_1$, $\hat{\mathfrak{a}} = [\hat{A}(\boldsymbol{\chi})]_1$, $\pi_{sm} = [\pi(\boldsymbol{\chi})]_1$.

Because the verification accepts, by the Schwartz–Zippel lemma, from this it follows (with all but negligible probability) that $\pi(\boldsymbol{X}) = X_\beta A(\boldsymbol{X}) + X_{\hat{\beta}}\hat{A}(\boldsymbol{X})$ as a polynomial. Now, the only elements in crs in group $\mathbb{G}_1$ that depend on X_β and $X_{\hat{\beta}}$ are the elements from crs_{sm}: (a) $X_\beta P_i(X) + X_{\hat{\beta}}\hat{P}_i(X)$ for each i, and (b) $X_\beta X_\varrho + X_{\hat{\beta}} X_{\hat{\varrho}}$. Thus, we must have that for some $\boldsymbol{a}$ and r,

$$\begin{aligned}\pi(\boldsymbol{X}) &= \sum_{i=1}^n a_i (X_\beta P_i(X) + X_{\hat{\beta}}\hat{P}_i(X)) + r(X_\beta X_\varrho + X_{\hat{\beta}} X_{\hat{\varrho}}) \\ &= X_\beta (\sum_{i=1}^n a_i P_i(X) + r X_\varrho) + X_{\hat{\beta}} (\sum_{i=1}^n a_i P_i(X) + r X_{\hat{\varrho}})\end{aligned}$$

Hence, $A(\boldsymbol{X}) = \sum_{i=1}^{n} a_i P_i(X) + rX_{\varrho}$ and $\hat{A}(\boldsymbol{X}) = \sum_{i=1}^{n} a_i \hat{P}_i(X) + rX_{\hat{\varrho}}$. Thus, $\mathfrak{a}$ and $\hat{\mathfrak{a}}$ commit to the same vector $\boldsymbol{a}$ using the same randomness r. □

Remark 1. Here, the only thing we require from the distribution $\mathcal{D}_{n,2}^{sm}$ is witness-sampleability and therefore exactly the same zero-knowledge argument can be used with any two shrinking commitment schemes. □

4.2 Permutation Matrix Argument

In this section, we show that a subargument of the new shuffle argument (see Prot. 1 and Prot. 2), where the verifier only computes $\mathfrak{a}_n$ as in prover Step 2 and then checks verification Step 5, gives us a permutation matrix argument. However, we first define the unit vector argument, prove its knowledge-soundness, and then use this to prove knowledge-soundness of the permutation matrix argument. The resulting permutation matrix argument is significantly simpler than in the FLZ shuffle argument [15].

Unit Vector Argument. In a unit vector argument [14], the prover aims to convince the verifier that he knows how to open a commitment $\mathfrak{a}$ to $(\boldsymbol{a}, r)$, such that exactly one coefficient a_I, $I \in [1 \mathinner{..} n]$, is equal to 1, while other coefficients of $\boldsymbol{a}$ are equal to 0. Recall [14,15] that if we define $\boldsymbol{V} := \left(\begin{smallmatrix} 2 \cdot I_{n \times n} \\ \boldsymbol{1}_n^\top \end{smallmatrix}\right) \in \mathbb{Z}_q^{(n+1) \times n}$ and $\boldsymbol{b} := \left(\begin{smallmatrix} \boldsymbol{0}_n \\ 1 \end{smallmatrix}\right) \in \mathbb{Z}_q^{n+1}$, then $\boldsymbol{a}$ is a unit vector iff

$$(\boldsymbol{V}\boldsymbol{a} + \boldsymbol{b} - \boldsymbol{1}_{n+1}) \circ (\boldsymbol{V}\boldsymbol{a} + \boldsymbol{b} - \boldsymbol{1}_{n+1}) = \boldsymbol{1}_{n+1} \ , \tag{2}$$

where $\circ$ denotes the Hadamard (entry-wise) product of two vectors. Really, this equation states that $a_i \in \{0, 1\}$ for each $i \in [1 \mathinner{..} n]$, and that $\sum a_i = 1$.

Similar to the 1-sparsity argument in [15], we construct the unit vector argument by using a variant of square span programs (SSP-s, [11]). To proceed, we need to define the following polynomials. For $i \in [1 \mathinner{..} n]$, set $P_i(X)$ to be the degree n polynomial that interpolates the ith column of the matrix $\boldsymbol{V}$, i.e.,

$$P_i(X) := 2\ell_i(X) + \ell_{n+1}(X) \ . \tag{3}$$

Set

$$P_0(X) := \ell_{n+1}(X) - 1 \ , \tag{4}$$

i.e., $P_0(X)$ is the polynomial that interpolates $\boldsymbol{b} - \boldsymbol{1}_{n+1}$. Define $Q(X) = (\sum_{i=1}^{n} a_i P_i(X) + P_0(X))^2 - 1$. Due to the choice of the polynomials, Eq. (2) holds iff $Q(\omega_i) = 0$ for all $i \in [1 \mathinner{..} n+1]$, which holds iff $Z(X) \mid Q(X)$, [15].

Lemma 1 ([14,15]). *The sets* $\{P_i(X)\}_{i=1}^{n} \cup \{P_0(X)\}$ *and* $\{P_i(X)\}_{i=1}^{n} \cup \{1\}$ *are both linearly independent.*

Clearly, both $\{P_i(X)\}_{i=1}^{n} \cup \{P_0(X)\}$ and $\{P_i(X)\}_{i=1}^{n} \cup \{1\}$ are a basis of all polynomials of degree $\leq n$.

Let

$$\hat{P}_i(X) := X^{(i+1)(n+1)} \ . \tag{5}$$

As explained before, $\{\hat{P}_i(X)\}_{i=1}^n$ are needed in the same-message argument and in the consistency argument. Due to that, the CRS has entries allowing to efficiently compute $[\hat{P}_i(\chi)]_1$, which means that a generic adversary of the unit vector argument of the current section has access to those polynomials.

Let $\mathcal{U}_n$ be the set of all unit vectors of length n. The unit vector argument is the following subargument of the new shuffle argument:

$\mathsf{K}_{uv}(\mathsf{gk}, n)$**:** the same as in Prot. 1.
$\mathsf{P}_{uv}(\mathsf{gk}, \mathsf{crs}, \mathfrak{a}_j, (\boldsymbol{a} \in \mathcal{U}_n, r))$**:** Compute $(\mathfrak{b}_j, \mathfrak{c}_j)$ as in Prot. 1.
$\mathsf{V}_{uv}(\mathsf{gk}, \mathsf{crs}, \mathfrak{a}_j, (\mathfrak{b}_j, \mathfrak{c}_j))$**:** $\alpha \leftarrow_r \mathbb{Z}_q$; Check that $(\mathfrak{a}_j + [\alpha]_1 + [P_0]_1) \bullet (\mathfrak{b}_j + [-\alpha]_2 + [P_0]_2) \stackrel{?}{=} \mathfrak{c}_i \bullet [\varrho]_2 + [1 - \alpha^2]_T$;

This argument is similar to the 1-sparsity argument presented in [15], but with a different CRS, meaning we cannot directly use their knowledge-soundness proof. Moreover, here the verifier generates α randomly, while in the argument of [15], α is generated by K. Fortunately, the CRS of the new shuffle argument is in a form that facilitates writing down a human readable and verifiable knowledge-soundness proof while [15] used a computer algebra system to solve a complicated system of polynomial equations.

The only problem of this argument is that it guarantees that the committed vector is a unit vector only under the condition that $A_0 = 0$ (see the statement of the following theorem). However, this will be fine since in the soundness proof of the shuffle argument, we can use the same-message argument to guarantee that $A_0 = 0$.

Theorem 3. *The described unit vector argument is perfectly complete and perfectly witness-indistinguishable. Assume that* $\{P_i(X)\}_{i=1}^n \cup \{1\}$*, and* $\{P_i(X)\}_{i=1}^n \cup \{P_0(X)\}$ *are two linearly independent sets. Assume that the same-message argument accepts. The unit vector argument is knowledge-sound in the GBGM in the following sense: there exists an extractor* Ext *such that if the verifier accepts Eq. 6 for* $j = i$*, then* Ext *returns* $(r_j, I_j \in [1\,..\,n])$*, such that*

$$\mathfrak{a}_j = [P_{I_j}(\chi) + r_j X_\varrho]_1 \ . \tag{6}$$

We note that the two requirements for linear independence follow from Lem. 1.

Proof. Completeness: For an honest prover, and $I = \sigma^{-1}(j)$, $\mathfrak{a}_j = [P_I(\chi) + r_j\varrho]_1$, $\mathfrak{b}_j = [P_I(\chi) + r_j\varrho]_2$, and $\mathfrak{c}_j = [r_j(2(P_I(\chi) + r_j\varrho + P_0(\chi)) - r_j\varrho) + h(\chi)Z(\chi)/\varrho]_1$ for $r_j \in \mathbb{Z}_q$. Hence, the verification equation assesses that $(P_I(\chi) + r_j\varrho + \alpha + P_0(\chi)) \cdot (P_I(\chi) + r_j\varrho - \alpha + P_0(\chi)) - (r_j(2(P_I(\chi) + r_j\varrho + P_0(\chi)) - r_j\varrho) + h(\chi)Z(\chi)/\varrho) \cdot \varrho - (1 - \alpha^2) = 0$. This simplifies to the claim that $(P_I(\chi) + P_0(\chi))^2 - 1 - h(\chi)Z(\chi) = 0$, or $h(\chi) = ((P_I(\chi) + P_0(\chi))^2 - 1)/Z(\chi)$ which holds since the prover is honest.

Knowledge-Soundness: Assume a generic adversary has returned $\mathfrak{a} = [A(\boldsymbol{\chi})]_1, \mathfrak{b} = [B(\boldsymbol{\chi})]_2, \mathfrak{c} = [C(\boldsymbol{\chi})]_2$, for attached polynomials $A(\boldsymbol{X}) =$

$crs_1(\boldsymbol{X}, A, a)$, $B(\boldsymbol{X}) = crs_2(\boldsymbol{X}, B, b)$, and $C(\boldsymbol{X}) = crs_1(\boldsymbol{X}, C, c)$, such that verification in Step 5 of Prot. 2 accepts. Observing this verification equation, it is easy to see that for the polynomial $V_{uv}(\boldsymbol{X}) := (A(\boldsymbol{X}) + X_\alpha + P_0(X)) \cdot (B(\boldsymbol{X}) - X_\alpha + P_0(X)) - C(\boldsymbol{X}) \cdot X_\varrho - (1 - X_\alpha^2)$, the verification equation assesses that $V_{uv}(\boldsymbol{\chi}) = 0$. Since we are in the GBGM, the adversary knows all coefficients of $A(\boldsymbol{X})$, $B(\boldsymbol{X})$, $C(\boldsymbol{X})$, and $V_{uv}(\boldsymbol{X})$.

Moreover, due to the knowledge-soundness of the same-message argument, we know that $A(\boldsymbol{X}) = a(X) + A_\varrho X_\varrho$ for $a(X) \in \operatorname{span}\{P_i(X)\}_{i=1}^n$. This simplifies correspondingly the polynomial $V_{uv}(\boldsymbol{X})$.

Now, let $V_{uv}(\boldsymbol{X} \setminus X)$ be equal to $V_{uv}(\boldsymbol{X})$ but without X being considered as an indeterminate; in particular, this means that the coefficients of $V_{uv}(\boldsymbol{X} \setminus X)$ can depend on X. Since the verifier accepts, $V_{uv}(\boldsymbol{\chi}) = 0$, so by the Schwartz–Zippel lemma, with all but negligible probability $V_{uv}(\boldsymbol{X}) \cdot X_\varrho = 0$ and hence also $V_{uv}(\boldsymbol{X} \setminus X) \cdot X_\varrho = 0$ as a polynomial. The latter holds iff all coefficients of $V_{uv}(\boldsymbol{X} \setminus X) \cdot X_\varrho$ are equal to 0. We will now consider the corollaries from the fact that coefficients C_M of the following monomials M of $V_{uv}(\boldsymbol{X} \setminus X) \cdot X_\varrho$ are equal to 0, and use them to prove the theorem:

$\underline{M = X_\alpha X_\varrho}$: $C_M = b(X) - a(X) + B_0 P_0(X) = 0$. Since $\{P_i(X)\}_{i=0}^n$ is linearly independent, we get that $B_0 = 0$, $b(X) = a(X)$.

$\underline{M = X_\varrho}$: $C_M = -Z(X)c^\dagger(X) - 1 + (a(X) + P_0(X))(b(X) + (B_0 + 1)P_0(X)) = 0$: since $B_0 = 0$ and $b(X) = a(X)$, we get that $C_M = -Z(X)c^\dagger(X) - 1 + (a(X) + P_0(X))^2 = 0$, or alternatively

$$C^\dagger(X) = \frac{(a(X) + P_0(X))^2 - 1}{Z(X)}$$

and hence, $Z(X) \mid ((a(X) + P_0(X))^2 - 1)$.

Therefore, due to the definition of $P_i(X)$ and the properties of square span programs, $a(X) = P_{I_j}(X)$ for some $I_j \in [1 \mathinner{..} n]$, and Eq. (6) holds. Denote $r_j := A_\varrho$. The theorem follows from the fact that we have a generic adversary who knows all the coefficients, and thus we can build an extractor that just outputs two of them, (r_j, I_j).

Witness-indistinguishability: Consider a witness $(\boldsymbol{a}, r_j)$. If $\boldsymbol{a}$ is fixed and the prover picks $r_j \leftarrow_r \mathbb{Z}_q$ (as in the honest case), then an accepting argument $(\mathfrak{a}_j, \mathfrak{b}_j)$ has distribution $\{([a']_1, [a']_2) : a' \leftarrow_r \mathbb{Z}_q\}$ and $\mathfrak{c}_j$ is uniquely fixed by $(\mathfrak{a}_j, \mathfrak{b}_j)$ and the verification equation. Hence an accepting argument $(\mathfrak{a}_j, \mathfrak{b}_j, \mathfrak{c}_j)$ has equal probability of being constructed from any valid $\boldsymbol{a} \in \mathcal{U}_n$. □

Remark 2. While a slight variant of this subargument was proposed in [15], they did not consider its privacy separately. It is easy to see that neither the new argument nor the 1-sparsity argument [15] is zero-knowledge in the case of type-III pairings. Knowing the witness, the prover can produce $\mathfrak{b}_j$ such that $\mathfrak{a}_j \bullet [1]_2 = [1]_1 \bullet \mathfrak{b}_j$. On the other hand, given an arbitrary input $\mathfrak{a} \in \mathbb{G}_1$ as input, the simulator cannot construct such $\mathfrak{b}$ since there is no efficient isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$. Witness-indistinguishability suffices in our application. □

Permutation Matrix Argument. A left stochastic matrix (i.e., its row vectors add up to $\mathbf{1}^\top$) where every row is 1-sparse is a permutation matrix, [34]. To guarantee that the matrix is left stochastic, it suffices to compute $\mathfrak{a}_n$ explicitly, i.e., $\mathfrak{a}_n = [\sum_{j=1}^{n} P_j(\chi)]_1 - \sum_{j=1}^{n-1} \mathfrak{a}_j$. After that, we need to perform the unit vector argument on every $\mathfrak{a}_j$; this guarantees that Eq. (6) holds for each row. Since a unit vector is also a 1-sparse vector, we get the following result.

Theorem 4. *The permutation matrix argument of this section is perfectly complete and perfectly witness-indistinguishable. Assume that $\{P_i(X)\}_{i=1}^{n} \cup \{1\}$ is a linearly independent set. The permutation matrix argument is knowledge-sound in the GBGM in the following sense: there exists an extractor* Ext *such that if the verifier accepts the verification equation on Step 5 of Prot. 2 for all $j \in [1 \mathinner{..} n]$, and $\mathfrak{a}_n$ is explicitly computed as in Prot. 2, then* Ext *outputs $(\sigma \in S_n, \boldsymbol{r})$, such that for all $j \in [1 \mathinner{..} n]$, $\mathfrak{a}_j = [P_{\sigma^{-1}(j)}(\chi) + r_j \varrho]_1$.*

Proof. Knowledge-soundness follows from the explicit construction of $\mathfrak{a}_n$, and from the fact that we have a generic adversary that knows all the coefficients, and thus also knows $(\sigma, \boldsymbol{r})$. More precisely, from the knowledge-soundness of the unit vector argument, for each $j \in [1 \mathinner{..} n]$ there exists an extractor that outputs $(\boldsymbol{a}_j, r_j)$ such that $\boldsymbol{a}_j$ is a unit vector with a 1 at some position $I_j \in [1 \mathinner{..} n]$ and $\mathfrak{a}_j = [P_{I_j}(\chi) + r_j \varrho]_1$. Since $[\sum_{j=1}^{n} P_{I_j}(\chi)]_1 = \sum_{j=1}^{n} \mathfrak{a}_j = [\sum_{j=1}^{n} P_j(\chi)]_1$, by the Schwartz-Zippel lemma we have that with overwhelming probability $\sum_{j=1}^{n} P_{I_j}(X) = \sum_{j=1}^{n} P_j(X)$ as a polynomial. Since due to Lem. 1 $\{P_i(X)\}_{i=1}^{n}$ is linearly independent, this means that $(I_1, \dots, I_n)$ is a permutation of $[1 \mathinner{..} n]$, so $(\boldsymbol{a}_j)_{j=1}^{n}$ is a permutation matrix.

Witness-indistinguishability follows from the witness-indistinguishability of the unit vector argument. □

4.3 Consistency Argument

The last subargument of the shuffle argument is the consistency argument. In this argument the prover aims to show that, given $\hat{\mathfrak{a}}$ that commits to a matrix $\boldsymbol{E} \in \mathbb{Z}_q^{n \times n}$ and two tuples ($\mathfrak{M}$ and $\mathfrak{M}'$) of Elgamal ciphertexts, it holds that $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{M}') = \boldsymbol{E} \cdot \mathsf{Dec}_{\mathsf{sk}}(\mathfrak{M})$. Since we use a shrinking commitment scheme, each $\hat{\mathfrak{a}}$ can commit to any matrix $\boldsymbol{E}$. Because of that, we use the fact that the permutation matrix argument is knowledge-sound in the GBGM, and thus there exists an extractor (this is formally provern in Sect. 3) that, after the same-message and the permutation matrix argument, extracts $\boldsymbol{E}$ and randomizer vectors $\boldsymbol{r}$ and $\boldsymbol{t}$, such that $\hat{\mathfrak{a}} = \left(\begin{smallmatrix}\boldsymbol{E}\\ \boldsymbol{r}^\top\end{smallmatrix}\right)^\top [\begin{smallmatrix}\hat{\boldsymbol{P}}\\ \hat{\varrho}\end{smallmatrix}]_1$. Hence, the guilt relation is

$$\mathcal{R}_{con,n}^{\mathsf{glt}} = \left\{ \begin{array}{l} (\mathsf{gk}, \mathsf{pk}, (\mathfrak{M}, \mathfrak{M}', \hat{\mathfrak{a}}), (\boldsymbol{E}, \boldsymbol{r})) : \\ \hat{\mathfrak{a}} = \left(\begin{smallmatrix}\boldsymbol{E}\\ \boldsymbol{r}^\top\end{smallmatrix}\right)^\top [\begin{smallmatrix}\hat{\boldsymbol{P}}\\ \hat{\varrho}\end{smallmatrix}]_1 \wedge \mathsf{Dec}_{\mathsf{sk}}(\mathfrak{M}') \neq \boldsymbol{E} \cdot \mathsf{Dec}_{\mathsf{sk}}(\mathfrak{M}) \end{array} \right\} .$$

We note that in the case of a shuffle argument, $\boldsymbol{E}$ is a permutation matrix. However, we will prove the soundness of the consistency argument for the general case of arbitrary matrices.

The new (general) consistency argument for a valid CRS distribution $\mathcal{D}_{n+1,1}$, such that $\mathsf{td}_{con} = (\chi, \hat{\varrho})$, works as follows. Note that the prover should be able to work without knowing the Elgamal secret key, and that $[\boldsymbol{M}]_1 = \mathsf{ck}$.

```
$\mathsf{K}_{con}(\mathsf{gk}, n)$: Return $(\mathsf{crs}_{con}; \mathsf{td}) = ([\boldsymbol{M}]_1 = [\begin{smallmatrix}\hat{\boldsymbol{P}}\\ \hat{\varrho}\end{smallmatrix}]_1, \mathsf{aux}; (\chi, \hat{\varrho})) \leftarrow \mathcal{D}_{n+1,1}$.
$\mathsf{P}_{con}(\mathsf{gk}, \mathsf{crs}, (\boldsymbol{\mathfrak{M}}, \boldsymbol{\mathfrak{M}}'), (\boldsymbol{E}, \boldsymbol{r}))$: // $\boldsymbol{\mathfrak{M}}' = \boldsymbol{E}\boldsymbol{\mathfrak{M}} + (t_i \cdot \mathsf{pk})_{i=1}^{n}$
    $\hat{\boldsymbol{\mathfrak{a}}} \leftarrow \left(\begin{smallmatrix}\boldsymbol{E}\\ \boldsymbol{r}^\top\end{smallmatrix}\right)^\top [\begin{smallmatrix}\hat{\boldsymbol{P}}\\ \hat{\varrho}\end{smallmatrix}]_1$; $r_t \leftarrow_r \mathbb{Z}_q$; $\mathsf{t} \leftarrow \boldsymbol{t}^\top[\hat{\boldsymbol{P}}]_1 + r_t[\hat{\varrho}]_1$; $\boldsymbol{\mathfrak{N}} \leftarrow \boldsymbol{r}^\top\boldsymbol{\mathfrak{M}} + r_t \cdot \mathsf{pk}$;
    Return $(\pi_{con} \leftarrow (\hat{\boldsymbol{\mathfrak{a}}}, \mathsf{t}, \boldsymbol{\mathfrak{N}}))$.
$\mathsf{V}_{con}(\mathsf{gk}, \mathsf{crs}, (\boldsymbol{\mathfrak{M}}, \boldsymbol{\mathfrak{M}}'), \pi_{con})$: Check that $[\hat{\boldsymbol{P}}]_1^\top \circ \boldsymbol{\mathfrak{M}}' - \hat{\boldsymbol{\mathfrak{a}}}^\top \circ \boldsymbol{\mathfrak{M}} \overset{?}{=} \mathsf{t} \circ \mathsf{pk} - [\hat{\varrho}]_1 \circ \boldsymbol{\mathfrak{N}}$.
```

Next, we prove that the consistency argument is culpably sound under a suitable KerMDH assumption (with an auxiliary input). After that, we prove that this variant of the KerMDH assumption holds in the GBGM, given that the auxiliary input satisfies some easily verifiable conditions.

Theorem 5. *Assume that $\mathcal{D}_n^{con}$ is a valid CRS distribution, where the matrix distribution outputs $[\boldsymbol{M}]_1 = [\begin{smallmatrix}\hat{\boldsymbol{P}}(\chi)\\ \hat{\varrho}\end{smallmatrix}]_1 \in \mathbb{Z}_q^{n+1}$ for $(\chi, \hat{\varrho}) \leftarrow_r \mathbb{Z}_q \times \mathbb{Z}_q^*$. The consistency argument is perfectly complete and perfectly zero knowledge. Assume that the $\mathcal{D}_n^{con}$-KerMDH assumption with an auxiliary input holds in $\mathbb{G}_1$. Then the consistency argument is culpably sound using $\mathcal{R}_{con}^{\mathsf{glt}}$ with the CRS $\mathsf{crs} = ([\boldsymbol{M}]_1, \mathsf{aux})$.*

Proof. Perfect completeness: In the case of the honest prover, $[\hat{\boldsymbol{P}}]_1^\top \circ \boldsymbol{\mathfrak{M}}' - \hat{\boldsymbol{\mathfrak{a}}}^\top \circ \boldsymbol{\mathfrak{M}} = [\hat{\boldsymbol{P}}]_1^\top \circ (\boldsymbol{E}\boldsymbol{\mathfrak{M}} + (t_i \cdot \mathsf{pk})_{i=1}^{n}) - \left(\begin{smallmatrix}\boldsymbol{E}\\ \boldsymbol{r}^\top\end{smallmatrix}\right)^\top [\begin{smallmatrix}\hat{\boldsymbol{P}}\\ \hat{\varrho}\end{smallmatrix}]_1^\top \circ \boldsymbol{\mathfrak{M}} = [\hat{\boldsymbol{P}}]_1^\top \circ ((t_i \cdot \mathsf{pk})_{i=1}^{n}) + r_t[\hat{\varrho}]_1 \circ \mathsf{pk} - [\hat{\varrho}]_1 \circ r_t\mathsf{pk} - [\hat{\varrho}]_1 \circ \boldsymbol{r}^\top\boldsymbol{\mathfrak{M}} = \mathsf{t} \circ \mathsf{pk} - [\hat{\varrho}]_1 \circ \boldsymbol{\mathfrak{N}}$.

Culpable soundness: Assume that $\mathcal{A}_{con}$ is an adversary that, given input $(\mathsf{gk}, \mathsf{crs}_{con})$ for $\mathsf{crs}_{con} = (\boldsymbol{M}, \mathsf{aux}) \leftarrow_r \mathcal{D}_n^{con}$, returns $u = (\boldsymbol{\mathfrak{M}}, \boldsymbol{\mathfrak{M}}')$, $\pi_{con} = (\hat{\boldsymbol{\mathfrak{a}}}, \mathsf{t}, \boldsymbol{\mathfrak{N}})$ and $w^{\mathsf{glt}} = (\boldsymbol{E}, \boldsymbol{r})$. $\mathcal{A}_{con}$ succeeds iff (i) the verifier of the consistency argument accepts, (ii) $\hat{\boldsymbol{\mathfrak{a}}} = \left(\begin{smallmatrix}\boldsymbol{E}\\ \boldsymbol{r}^\top\end{smallmatrix}\right)^\top [\begin{smallmatrix}\hat{\boldsymbol{P}}\\ \hat{\varrho}\end{smallmatrix}]_1$, and (iii) $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{\mathfrak{M}}') = \boldsymbol{E} \cdot \mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{\mathfrak{M}})$. Assume $\mathcal{A}_{con}$ succeeds with probability ε.

We construct the following adversary $\mathcal{A}_{ker}$ that breaks the $\mathcal{D}_n^{con}$-KerMDH assumption with auxiliary input. $\mathcal{A}_{ker}$ gets an input $(\mathsf{gk}, [\boldsymbol{M}]_1, \mathsf{aux})$ where $\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa)$ and $\mathsf{crs}_{ker} = ([\boldsymbol{M}]_1, \mathsf{aux}) \leftarrow_r \mathcal{D}_n^{con}$, and is supposed to output $[\boldsymbol{c}]_2$, such that $\boldsymbol{M}^\top\boldsymbol{c} = \boldsymbol{0}$ but $\boldsymbol{c} \neq \boldsymbol{0}$.

On such an input, $\mathcal{A}_{ker}$ first parses the auxiliary input as $\mathsf{aux} = (\mathsf{aux}', \mathsf{pk})$, then picks $\mathsf{sk}' \leftarrow_r \mathbb{Z}_q$ and creates $\mathsf{crs}_{con} = ([\boldsymbol{M}]_1, (\mathsf{aux}', \mathsf{pk}'))$, where $\mathsf{pk}' = ([1]_2, [\mathsf{sk}']_2)$. Note that crs_{ker} and crs_{con} have the same distribution. $\mathcal{A}_{ker}$ makes a query to $\mathcal{A}_{con}$ with input $(\mathsf{gk}, \mathsf{crs}_{con})$.

After obtaining the answer $u = (\boldsymbol{\mathfrak{M}}, \boldsymbol{\mathfrak{M}}')$, $\pi_{con} = (\hat{\boldsymbol{\mathfrak{a}}}, \mathsf{t}, \boldsymbol{\mathfrak{N}})$ and $w^{\mathsf{glt}} = (\boldsymbol{E}, \boldsymbol{r})$ from $\mathcal{A}_{con}$, he does the following:

1. If $[\hat{\boldsymbol{P}}]_1^\top \circ \boldsymbol{\mathfrak{M}}' - \hat{\boldsymbol{\mathfrak{a}}}^\top \circ \boldsymbol{\mathfrak{M}} \neq \mathsf{t} \circ \mathsf{pk}' - [\hat{\varrho}]_1 \circ \boldsymbol{\mathfrak{N}}$ then abort.
2. Use sk' to decrypt: $\boldsymbol{\mathfrak{m}} \leftarrow \mathsf{Dec}_{\mathsf{sk}'}(\boldsymbol{\mathfrak{M}})$, $\boldsymbol{\mathfrak{m}}' \leftarrow \mathsf{Dec}_{\mathsf{sk}'}(\boldsymbol{\mathfrak{M}}')$, $\mathfrak{n} \leftarrow \mathsf{Dec}_{\mathsf{sk}'}(\boldsymbol{\mathfrak{N}})$.
3. Return $[\boldsymbol{c}]_2 \leftarrow \left(\begin{smallmatrix}\boldsymbol{\mathfrak{m}}' - \boldsymbol{E}\boldsymbol{\mathfrak{m}}\\ \mathfrak{n} - \boldsymbol{r}^\top\boldsymbol{\mathfrak{m}}\end{smallmatrix}\right)$.

Let us now analyze $\mathcal{A}_{ker}$'s success probability. With probability $1 - \varepsilon$, $\mathcal{A}_{ker}$ fails, in which case $\mathcal{A}_{ker}$ will abort. Otherwise, the verification equation holds.

Decrypting the right-hand sides of each $\circ$ in the verification equation in Step 6, we get that $[\boldsymbol{M}]_1^\top \bullet \left(\begin{smallmatrix} \boldsymbol{\mathfrak{m}}' \\ \mathfrak{n} \end{smallmatrix} \right) - \hat{\boldsymbol{\mathfrak{a}}}^\top \bullet \boldsymbol{\mathfrak{m}} = [0]_T$.

Since $\mathcal{A}_{con}$ is successful, then $\boldsymbol{\mathfrak{a}} = \left(\begin{smallmatrix} \boldsymbol{E} \\ \boldsymbol{r}^\top \end{smallmatrix} \right)^\top [\boldsymbol{M}]_1$, hence we get

$$\begin{aligned}
[0]_T =& [\boldsymbol{M}]_1^\top \bullet \left(\begin{smallmatrix} \boldsymbol{\mathfrak{m}}' \\ \mathfrak{n} \end{smallmatrix} \right) - [\boldsymbol{M}]_1^\top \left(\begin{smallmatrix} \boldsymbol{E} \\ \boldsymbol{r}^\top \end{smallmatrix} \right) \bullet \boldsymbol{\mathfrak{m}} \\
=& [\boldsymbol{M}]_1^\top \bullet \left(\begin{smallmatrix} \boldsymbol{\mathfrak{m}}' \\ \mathfrak{n} \end{smallmatrix} \right) - [\boldsymbol{M}]_1^\top \left(\begin{smallmatrix} \boldsymbol{E} \\ \boldsymbol{r}^\top \end{smallmatrix} \right) \bullet \boldsymbol{\mathfrak{m}} \\
=& [\boldsymbol{M}]_1^\top \bullet \left(\begin{smallmatrix} \boldsymbol{\mathfrak{m}}' \\ \mathfrak{n} \end{smallmatrix} \right) - [\boldsymbol{M}]_1^\top \bullet \left(\begin{smallmatrix} \boldsymbol{E} \\ \boldsymbol{r}^\top \end{smallmatrix} \right) \boldsymbol{\mathfrak{m}} \\
=& [\boldsymbol{M}]_1^\top \bullet \begin{pmatrix} \boldsymbol{\mathfrak{m}}' - \boldsymbol{E}\boldsymbol{\mathfrak{m}} \\ \mathfrak{n} - \boldsymbol{r}^\top \boldsymbol{\mathfrak{m}} \end{pmatrix} \quad .
\end{aligned}$$

Since $\mathcal{A}_{con}$ is successful, then $\boldsymbol{\mathfrak{m}}' \neq \boldsymbol{E}\boldsymbol{\mathfrak{m}}$, which means that $\boldsymbol{c} \neq \boldsymbol{0}$ but $\boldsymbol{M}^\top \boldsymbol{c} = \boldsymbol{0}$. Thus, $\mathcal{A}_{ker}$ solves the $\mathcal{D}_n^{con}$-KerMDH problem with probability ε.

Perfect Zero-Knowledge: The simulator $\mathsf{S}_{con}(\mathsf{gk}, \mathsf{crs}, (\boldsymbol{\mathfrak{M}}, \boldsymbol{\mathfrak{M}}'), \mathsf{td}_{con} = (\chi, \hat{\varrho}))$ proceeds like the prover in the case $\boldsymbol{E} = \boldsymbol{I}$ (the identity matrix) and $\boldsymbol{t} = \boldsymbol{0}$, and then computes an $\boldsymbol{\mathfrak{N}}$ that makes the verifier accept. More precisely, the simulator sets $\boldsymbol{r} \leftarrow_r \mathbb{Z}_q$, $\hat{\boldsymbol{\mathfrak{a}}} \leftarrow [\hat{\boldsymbol{P}}]_1 + \boldsymbol{r}[\hat{\varrho}]_1$, $r_t \leftarrow_r \mathbb{Z}_q$, $\mathfrak{t} \leftarrow r_t[\hat{\varrho}]_1$, and $\boldsymbol{\mathfrak{N}} \leftarrow (\hat{\boldsymbol{P}}/\hat{\varrho} + \boldsymbol{r})^\top \boldsymbol{\mathfrak{M}} - (\hat{\boldsymbol{P}}/\hat{\varrho})^\top \boldsymbol{\mathfrak{M}}' + r_t \cdot \mathsf{pk}$. S_{con} then outputs $\pi_{con} \leftarrow (\hat{\boldsymbol{\mathfrak{a}}}, \mathfrak{t}, \boldsymbol{\mathfrak{N}})$.

Due to the perfect hiding property of the commitment scheme, $(\hat{\boldsymbol{\mathfrak{a}}}, \mathfrak{t})$ has the same distribution as in the real protocol. Moreover,

$$\begin{aligned}
[\hat{\boldsymbol{P}}]_1^\top \circ \boldsymbol{\mathfrak{M}}' - \hat{\boldsymbol{\mathfrak{a}}}^\top \circ \boldsymbol{\mathfrak{M}} =& [\hat{\boldsymbol{P}}]_1^\top \circ \boldsymbol{\mathfrak{M}}' - ([\hat{\boldsymbol{P}}]_1 + \boldsymbol{r}[\hat{\varrho}]_1)^\top \circ \boldsymbol{\mathfrak{M}} \\
=& [\hat{\varrho}]_1 \circ (\hat{\boldsymbol{P}}/\hat{\varrho})^\top \boldsymbol{\mathfrak{M}}' - [\hat{\varrho}]_1 \circ (\hat{\boldsymbol{P}}/\hat{\varrho} + \boldsymbol{r})^\top \boldsymbol{\mathfrak{M}} \\
=& [\hat{\varrho}]_1 \circ (r_t \cdot \mathsf{pk} - r_t \cdot \mathsf{pk} + (\hat{\boldsymbol{P}}/\hat{\varrho})^\top \boldsymbol{\mathfrak{M}}' - (\hat{\boldsymbol{P}}/\hat{\varrho} + \boldsymbol{r})^\top \boldsymbol{\mathfrak{M}}) \\
=& \mathfrak{t} \circ \mathsf{pk} - [\hat{\varrho}]_1 \circ \boldsymbol{\mathfrak{N}}
\end{aligned}$$

and thus this choice of $\boldsymbol{\mathfrak{N}}$ makes the verifier accept. Since $\mathfrak{t}$ is uniformly random and $\boldsymbol{\mathfrak{N}}$ is uniquely defined by $(\hat{\boldsymbol{\mathfrak{a}}}, \mathfrak{t})$ and the verification equation, we have a perfect simulation. □

Example 1. In the case of the shuffle argument, $\boldsymbol{E}$ is a permutation matrix with $E_{ij} = 1$ iff $j = \sigma(i)$. Thus, $[\hat{\boldsymbol{\mathfrak{a}}}]_1 = \left(\begin{smallmatrix} \boldsymbol{E} \\ \boldsymbol{r}^\top \end{smallmatrix} \right)^\top \left[\begin{smallmatrix} \hat{\boldsymbol{P}} \\ \hat{\varrho} \end{smallmatrix} \right]_1 = [(\hat{P}_{\sigma^{-1}(i)})_{i=1}^n + \boldsymbol{r}\hat{\varrho}]_1$, and $\boldsymbol{E}\boldsymbol{\mathfrak{m}} = (\mathfrak{m}_{\sigma(i)})_{i=1}^n$. Hence in the shuffle argument, assuming that it has been already established that $\boldsymbol{E}$ is a permutation matrix, then after the verification in Step 6 one is assured that $\boldsymbol{\mathfrak{m}}' = (\mathfrak{m}_{\sigma(i)})_{i=1}^n$. □

We will now prove that the used variant of KerMDH with auxiliary input is secure in the GBGM. Before going on, we will establish the following linear independence result.

Lemma 2. *The set $\Psi_\times := \{P_i(X)\hat{P}_j(X)\}_{1 \le i,j \le n} \cup \{\hat{P}_i(X)\}_{i=1}^n$ is linearly independent.*

Proof. By Lemma 1, for each $j \in [1 \mathinner{..} n]$ we have that $\{P_i(X)\hat{P}_j(X)\}_{1 \le i \le n} \cup \{\hat{P}_j(X)\}$ is linearly independent. Hence to show that $\Psi_\times$ is linearly independent, it suffices to show that for all $1 \le j < k \le n$, the span of sets

$\{P_i(X)\hat{P}_j(X)\}_{1 \le i \le n} \cup \{\hat{P}_j(X)\}$ and $\{P_i(X)\hat{P}_k(X)\}_{1 \le i \le n} \cup \{\hat{P}_k(X)\}$ only intersect at 0. This holds since for non-zero vectors $(\boldsymbol{a}, \boldsymbol{b})$ and integers (A, B), $\sum_{i=1}^n a_i P_i(X)\hat{P}_j(X) + A\hat{P}_j(X) = (\sum_{i=1}^n a_i P_i(X) + A)X^{(j+1)(n+1)}$ has degree at most $n + (j+1)(n+1) < (k+1)(n+1)$, while $\sum_{i=1}^n b_i P_i(X)\hat{P}_k(X) + B\hat{P}_k(X) = (\sum_{i=1}^n b_i P_i(X) + B)X^{(k+1)(n+1)}$ has degree at least $(k+1)(n+1)$. □

Theorem 6. *Assume that $(\hat{P}_i(X))_{i=1}^n$ satisfy the following properties:*
- *$\{\hat{P}_i(X)\}_{i=1}^n$ is linearly independent,*
- *$\{P_i(X)\hat{P}_j(X)\}_{1 \le i,j \le n}$ is linearly independent,*

Assume also that the second input aux *output by $\mathcal{D}_n^{con}$ satisfies the following property. The subset* aux_2 *of $\mathbb{G}_2$-elements in* aux *must satisfy that*
- aux_2 *does not depend on $\hat{\varrho}$,*
- *the only element in* aux_2 *that depends on ϱ is $[\varrho]_2$,*
- *the only elements in* aux_2 *that depend on χ are $[P_i(\chi)]_2$ for $i \in [0\,..\,n]$.*

Then in the GBGM, the $\mathcal{D}_n^{con}$-KerMDH assumption with an auxiliary input holds in $\mathbb{G}_1$.

Clearly, $(P_i(X))_{i=1}^n$, $(\hat{P}_i(X))_{i=1}^n$, and crs in Prot. 1 (if used as aux) satisfy the required properties.

Proof. Consider a generic group adversary $\mathcal{A}_{ker}$ who, given $([\begin{smallmatrix}\hat{\boldsymbol{P}}(\boldsymbol{\chi})\\ \hat{\varrho}\end{smallmatrix}]_1, \mathsf{aux})$ as an input, outputs a non-zero solution $\left(\begin{smallmatrix}\mathfrak{m}\\ \mathfrak{n}\end{smallmatrix}\right) \in \mathbb{G}_2^{n+1}$ to the KerMDH problem, i.e., $[\boldsymbol{M}]_1^\top \bullet \left(\begin{smallmatrix}\mathfrak{m}\\ \mathfrak{n}\end{smallmatrix}\right) = [0]_T$. In the generic model, each element in aux_2 has some polynomial attached to it. Since $\mathcal{A}_{ker}$ is a generic adversary, he knows polynomials $M_i(\boldsymbol{X})$ (for each i) and $N(\boldsymbol{X})$, such that $\mathfrak{m}_i = [M_i(\boldsymbol{\chi})]_2$ and $\mathfrak{n} = [N(\boldsymbol{\chi})]_2$. Those polynomials are linear combinations of the polynomials involved in aux_2.

Hence, if $\left(\begin{smallmatrix}\mathfrak{m}\\ \mathfrak{n}\end{smallmatrix}\right) \in \mathbb{G}_2^{n+1}$ is a solution to the KerMDH problem, then $[\boldsymbol{M}]_1^\top \bullet \left(\begin{smallmatrix}\mathfrak{m}\\ \mathfrak{n}\end{smallmatrix}\right) = [0]_T$ or equivalently, $\hat{V}_{ker}(\boldsymbol{\chi}) = 0$, where

$$\hat{V}_{ker}(\boldsymbol{X}) := \sum_{i=1}^n \hat{P}_i(X)M_i(\boldsymbol{X}) + X_{\hat{\varrho}} \cdot N(\boldsymbol{X}) \ , \qquad (7)$$

for coefficients known to the generic adversary. By the Schwartz–Zippel lemma, from this it follows (with all but negligible probability) that $\hat{V}_{ker}(\boldsymbol{X}) = 0$ as a polynomial.

Due to the assumptions on aux_2, and since $\{P_i(X)\}_{i=0}^n \cup \{1\}$ and $\{P_i(X)\}_{i=0}^n \cup \{P_0(X)\}$ are interchangeable bases of degree-($\le n$) polynomials, we can write

$$M_i(\boldsymbol{X}) = \textstyle\sum_{j=1}^n M_{ij}P_j(X) + M_i' X_\varrho + m_i(\boldsymbol{X}) \ ,$$

where $m_i(\boldsymbol{X})$ does not depend on X, X_ϱ or $X_{\hat{\varrho}}$.

Since $\sum \hat{P}_i(X)M_i(\boldsymbol{X})$ does not depend on $X_{\hat{\varrho}}$, we get from $\hat{V}_{ker}(\boldsymbol{X}) = 0$ that $\sum_{i=1}^n \hat{P}_i(X)M_i(\boldsymbol{X}) = N(\boldsymbol{X}) = 0$.

Next, we can rewrite $\sum_{i=1}^n \hat{P}_i(X)M_i(\boldsymbol{X}) = 0$ as

$$\sum_{i=1}^n \sum_{j=1}^n M_{ij}\hat{P}_i(X)P_j(X) + X_\varrho \sum_{i=1}^n M_i'\hat{P}_i(X) + \sum_{i=1}^n \hat{P}_i(X)m_i(\boldsymbol{X}) = 0 \ .$$

Due to assumptions on aux_2, this means that

- $\sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} \hat{P}_i(X) P_j(X) = 0$. Since $\{\hat{P}_i(X) P_j(X)\}_{i,j \in [1 \mathinner{..} n]}$ is linearly independent, $M_{ij} = 0$ for all $i, j \in [1 \mathinner{..} n]$,
- $\sum_{i=1}^{n} M'_i \hat{P}_i(X) = 0$. Since $\{\hat{P}_i(X)\}_{i \in [1 \mathinner{..} n]}$ is linearly independent, $M'_i = 0$ for all $i \in [1 \mathinner{..} n]$.
- $\sum_{i=1}^{n} m_i(\boldsymbol{X}) \hat{P}_i(X) = 0$. Since $m_i(\boldsymbol{X})$ does not depend on X and $\{\hat{P}_i(X)\}_{i \in [1 \mathinner{..} n]}$ is linearly independent, we get $m_i(\boldsymbol{X}) = 0$ for all $i \in [1 \mathinner{..} n]$.

Hence, $M_i(\boldsymbol{X}) = 0$ for each $i \in [1 \mathinner{..} n]$. Thus, with all but negligible probability, we have that $N(\boldsymbol{X}) = M_i(\boldsymbol{X}) = 0$, which means $\binom{\mathfrak{m}}{\mathfrak{n}} = [\boldsymbol{0}]_2$. □

5 Security Proof of Shuffle

Theorem 7. *The new shuffle argument is perfectly complete.*

Proof. We showed the completeness of the same-message argument (i.e., that the verification equation on Step 4 holds) in Sect. 4.1, the completeness of the unit vector argument (i.e., that the verification equation on Step 5 holds) in Thm. 3, and the completeness of the consistency argument (i.e., that the verification equation on Step 6 holds) in Sect. 4.3. □

Theorem 8 (Soundness of Shuffle Argument). *Assume that the following sets are linearly independent:*

- $\{P_i(X)\}_{i=0}^{n}$,
- $\{P_i(X)\}_{i=1}^{n} \cup \{1\}$,
- $\{P_i(X) \hat{P}_j(X)\}_{1 \le i,j \le n}$.

Let $\mathcal{D}_n^{con}$ *be as before with* $([\boldsymbol{M}]_1, \mathsf{aux})$, *such that* aux *is equal to the CRS in Prot. 1 minus the elements already in* $[\boldsymbol{M}]_1$. *If the* $\mathcal{D}_1$*-KerMDH assumption holds in* $\mathbb{G}_2$, *the* $\mathcal{D}_n^{((P_i(X))_{i=1}^{n}, X_\varrho)}$*-KerMDH and* $\mathcal{D}_n^{((\hat{P}_i(X))_{i=1}^{n}, X_{\hat{\varrho}})}$*-KerMDH assumptions hold and the* $\mathcal{D}_n^{con}$*-KerMDH assumption with auxiliary input holds in* $\mathbb{G}_1$, *then the new shuffle argument is sound in the GBGM.*

Proof. First, from the linear independence of $\{P_i(X) \hat{P}_j(X)\}_{1 \le i,j \le n}$, we get straightforwardly that $\{\hat{P}_j(X)\}_{j=1}^{n}$ is also linearly independent. We construct the following simple sequence of games.

$\underline{\mathsf{GAME}_0}$: This is the original soundness game. Let $\mathcal{A}_{sh}$ be an adversary that breaks the soundness of the new shuffle argument. That is, given $(\mathsf{gk}, \mathsf{crs})$, $\mathcal{A}_{sh}$ outputs $u = (\boldsymbol{\mathfrak{M}}, \boldsymbol{\mathfrak{M}}')$ and an accepting proof π, such that for all permutations σ, there exists $i \in [1 \mathinner{..} n]$, such that $\mathsf{Dec}_{\mathsf{sk}}(\mathfrak{M}'_i) \neq \mathsf{Dec}_{\mathsf{sk}}(\mathfrak{M}_{\sigma(i)})$.

$\underline{\mathsf{GAME}_1}$: Here, we let $\mathcal{A}_{con}$ herself generate a new secret key $\mathsf{sk} \leftarrow_r \mathbb{Z}_q$ and set $\mathsf{pk} \leftarrow [1, \mathsf{sk}]_2$. Since the distribution of pk is witness-sampleable, then for any (even omnipotent) adversary $\mathcal{B}$, $|\Pr[\mathsf{GAME}_1(\mathcal{B}) = 1] - \Pr[\mathsf{GAME}_0(\mathcal{B}) = 1]| = 0$ (i.e., GAME_1 and GAME_0 are indistinguishable).

Now, consider GAME_1 in more detail. Clearly, under given assumptions and due to the construction of crs in Prot. 1 the same-message and permutation matrix arguments are knowledge-sound. Hence there exist extractors $\mathsf{Ext}_{sm:i}^{\mathcal{A}}$ and $\mathsf{Ext}_{pm}^{\mathcal{A}}$ such that the following holds:

```
1. Generate a new random tape $r'$ for $\mathcal{A}_{sh}$
2. Run $\mathcal{A}_{sh}(\mathsf{gk}, \mathsf{crs}_{sh}; r')$ to obtain a shuffle argument $\pi_{sh} = (\boldsymbol{\mathfrak{M}}', (\mathfrak{a}_i)_{i=1}^{n-1}, \pi_{sp}, \pi_{sm}, (\hat{\mathfrak{a}}_i)_{i=1}^{n-1}, \mathfrak{t}, \boldsymbol{\mathfrak{N}})$. Compute $\mathfrak{a}_n$ and $\hat{\mathfrak{a}}_n$ as in Prot. 1.
3. Abort when $\pi_{sh}$ is not accepting.
4. For $i = 1$ to $n$:
   (a) Extract $(\boldsymbol{E}'_i, r'_i) \leftarrow \mathsf{Ext}^{\mathcal{A}}_{sm:i}(\mathsf{gk}, \mathsf{crs}_{sh}; r')$.
   (b) Abort when $(\boldsymbol{E}'_i, r'_i)$ is not a valid opening of $\mathfrak{a}_i$ or $\hat{\mathfrak{a}}_i$.
5. Extract $(\boldsymbol{E}, \boldsymbol{r}) \leftarrow \mathsf{Ext}^{\mathcal{A}}_{pm}(\mathsf{gk}, \mathsf{crs}_{sh}; r')$.
6. Abort when $\boldsymbol{E}' \neq \boldsymbol{E}$ or $\boldsymbol{r}' \neq \boldsymbol{r}$.
7. // At this point $\mathcal{A}_{con}$ knows a permutation matrix $\boldsymbol{E}$ and vector $\boldsymbol{r}$, such that $\hat{\boldsymbol{\mathfrak{a}}} = [\boldsymbol{E}^\top \hat{\boldsymbol{P}} + \boldsymbol{r}\hat{\varrho}]_1$ and $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{\mathfrak{M}}') \neq \boldsymbol{E} \cdot \mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{\mathfrak{M}})$.
8. Return $u = (\boldsymbol{\mathfrak{M}}, \boldsymbol{\mathfrak{M}}')$, $\pi_{con} = (\hat{\boldsymbol{\mathfrak{a}}}, \mathfrak{t}, \boldsymbol{\mathfrak{N}})$, and $w^{\mathsf{glt}} = (\boldsymbol{E}, \boldsymbol{r})$.
```

Protocol 3: Adversary $\mathcal{A}_{con}$ on input $(\mathsf{gk}, \mathsf{crs}_{con})$

- If the same-message argument verifier accepts $(\mathfrak{a}_i, \hat{\mathfrak{a}}_i, \pi_{sm:i}) \leftarrow \mathcal{A}_{sm}(\mathsf{gk}, \mathsf{crs}_{sh}; r')$, then $\mathsf{Ext}^{\mathcal{A}}_{sm:i}(\mathsf{gk}, \mathsf{crs}_{sh}; r')$ outputs the common opening $(\boldsymbol{E}'_i, r_i)$ of $\mathfrak{a}_i$ and $\hat{\mathfrak{a}}_i$.
- If the permutation matrix argument verifier accepts $((\mathfrak{a}_i)_{i=1}^{n-1}, \pi_{pm}) \leftarrow \mathcal{A}_{pm}(\mathsf{gk}, \mathsf{crs}_{sh}; r')$, then $\mathsf{Ext}^{\mathcal{A}}_{pm}(\mathsf{gk}, \mathsf{crs}_{sh}; r')$ outputs a permutation matrix $\boldsymbol{E}$ and vector $\boldsymbol{r}$ such that $\boldsymbol{\mathfrak{a}} = [\boldsymbol{E}^\top \boldsymbol{P} + \boldsymbol{r}\varrho]_1$ (here, $\mathfrak{a}_n$ is computed as in Prot. 1).

We construct the following adversary $\mathcal{A}_{con}$, see Prot. 3, that breaks the culpable soundness of the consistency argument using $\mathcal{R}^{\mathsf{glt}}_{con}$ with auxiliary input. $\mathcal{A}_{con}$ gets an input $(\mathsf{gk}, \mathsf{crs}_{con})$ where $\mathsf{gk} \leftarrow \mathsf{BG}(1^\kappa)$, and $\mathsf{crs}_{con} \leftarrow ([\boldsymbol{M}]_1, \mathsf{aux}) \leftarrow_r \mathcal{D}^{con}_n$. $\mathcal{A}_{con}$ is supposed to output $(u, \pi_{con}, w^{\mathsf{glt}})$ where $u = (\boldsymbol{\mathfrak{M}}, \boldsymbol{\mathfrak{M}}')$, $\pi_{con} = (\hat{\boldsymbol{\mathfrak{a}}}, \mathfrak{t}, \boldsymbol{\mathfrak{N}})$, and $w^{\mathsf{glt}} = (\boldsymbol{E}, \boldsymbol{r})$ such that $\hat{\boldsymbol{\mathfrak{a}}} = [\boldsymbol{E}^\top \hat{\boldsymbol{P}} + \boldsymbol{r}\hat{\varrho}]_1$, π_{con} is accepting, and $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{\mathfrak{M}}') \neq \boldsymbol{E} \cdot \mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{\mathfrak{M}})$.

It is clear that if $\mathcal{A}_{con}$ does not abort, then he breaks the culpable soundness of the consistency argument. Let us now analyze the probability that $\mathcal{A}_{con}$ does abort in any step.

1. $\mathcal{A}_{con}$ aborts in Step 3 if $\mathcal{A}_{sh}$ fails, i.e., with probability $1 - \varepsilon_{sh}$.
2. $\mathcal{A}_{con}$ will never abort in Step 4b since by our definition of knowledge-soundness, $\mathsf{Ext}^{\mathcal{A}}_{sm:i}$ always outputs a valid opening as part of the witness.
3. $\mathcal{A}_{con}$ aborts in Step 6 only when he has found two different openings of $\mathfrak{a}_i$ for some $i \in [1 \,..\, n]$. Hence, the probability of abort is upper bounded by $n\varepsilon_{binding}$, where $\varepsilon_{binding}$ is the probability of breaking the binding property of the $((P_i(X))_{i=1}^n, \varrho)$-commitment scheme.

Thus, $\mathcal{A}_{con}$ aborts with probability $\leq 1 - \varepsilon_{sh} + n\varepsilon_{binding}$, or succeeds with probability $\varepsilon_{con} \geq \varepsilon_{sh} - n\varepsilon_{binding}$. Hence $\varepsilon_{sh} \leq \varepsilon_{con} + n\varepsilon_{binding}$. Since under the assumptions of this theorem, both ε_{con} and $\varepsilon_{binding}$ are negligible, we have that ε_{sh} is also negligible. □

Theorem 9. *The new shuffle argument is perfectly zero-knowledge.*

```
1. For $i = 1$ to $n - 1$: // commits to the permutation $\sigma = \mathrm{Id}$
   (a) $r_i \leftarrow_r \mathbb{Z}_q$; $\mathfrak{r}_i \leftarrow r_i[\varrho]_1$;
   (b) $\mathfrak{a}_i \leftarrow [P_i]_1 + \mathfrak{r}_i$; $\mathfrak{b}_i \leftarrow [P_i]_2 + r_i[\varrho]_2$; $\hat{\mathfrak{a}}_i \leftarrow [\hat{P}_i]_1 + r_i[\hat{\varrho}]_1$;
2. $\mathfrak{a}_n \leftarrow [\sum_{i=1}^{n} P_i]_1 - \sum_{j=1}^{n-1} \mathfrak{a}_j$; $\mathfrak{b}_n \leftarrow [\sum_{i=1}^{n} P_i]_2 - \sum_{j=1}^{n-1} \mathfrak{b}_j$;
3. $\hat{\mathfrak{a}}_n \leftarrow [\sum_{i=1}^{n} \hat{P}_i]_1 - \sum_{j=1}^{n-1} \hat{\mathfrak{a}}_j$;
4. $r_n \leftarrow -\sum_{i=1}^{n-1} r_i$; $\mathfrak{r}_n \leftarrow r_n[\varrho]_1$;
5. For $i = 1$ to $n$:
   (a) $\mathfrak{c}_i \leftarrow r_i \cdot (2(\mathfrak{a}_i + [P_0]_1) - \mathfrak{r}_i) + [((P_i + P_0)^2 - 1)/\varrho]_1$;
   (b) $\mathfrak{d}_i \leftarrow [\beta P_i + \hat{\beta}\hat{P}_i]_1 + r_i[\beta\varrho + \hat{\beta}\hat{\varrho}]_1$;
6. Set $r_t \leftarrow_r \mathbb{Z}_q$;
7. $\mathfrak{t} \leftarrow r_t[\hat{\varrho}]_1$; // Commits to $\mathbf{0}$
8. $\boldsymbol{\mathfrak{N}} \leftarrow (\hat{\boldsymbol{P}}/\hat{\varrho} + \boldsymbol{r})^\top \boldsymbol{\mathfrak{M}} - (\hat{\boldsymbol{P}}/\hat{\varrho})^\top \boldsymbol{\mathfrak{M}}' + r_t \cdot \mathsf{pk}$;
9. $\pi_{sm} \leftarrow \boldsymbol{\mathfrak{d}}$; // Same-message argument
10. $\pi_{uv} \leftarrow ((\mathfrak{b}_i)_{i=1}^{n-1}, \boldsymbol{\mathfrak{c}})$; // Unit vector argument
11. $\pi_{con} \leftarrow ((\hat{\mathfrak{a}}_i)_{i=1}^{n-1}, \mathfrak{t}, \boldsymbol{\mathfrak{N}})$; // Consistency argument
12. Return $\pi_{sh} \leftarrow ((\mathfrak{M}'_i)_{i=1}^{n}, (\mathfrak{a}_i)_{i=1}^{n-1}, \pi_{uv}, \pi_{sm}, \pi_{con})$.
```

Protocol 4: The simulator of the shuffle argument

Proof. We proved that the permutation matrix argument is witness-indistinguishable (Thm. 4), and that the same-message argument is zero-knowledge (Thm. 2) and hence also witness-indistinguishable.

We will construct a simulator S_{sh}, that given a CRS crs and trapdoor $\mathsf{td} = (\chi, \hat{\varrho})$ simulates the prover in Prot. 1. S_{sh} takes any pair $(\boldsymbol{\mathfrak{M}}, \boldsymbol{\mathfrak{M}}')$ as input and performs the steps in Prot. 4. Clearly, the simulator computes $((\mathfrak{a}_i)_{i=1}^{n-1}, (\hat{\mathfrak{a}}_i)_{i=1}^{n-1}, \pi_{uv}, \pi_{sm})$ exactly as an honest prover with permutation $\sigma = \mathrm{Id}$ would, which ensures correct distribution of these values. Moreover, since the commitment scheme is perfectly hiding and the permutation matrix and same-message arguments are witness-indistinguishable, the distribution of these values is identical to that of an honest prover that uses a (possibly different) permutation σ' and randomness values to compute $\boldsymbol{\mathfrak{M}}'$ from $\boldsymbol{\mathfrak{M}}$.

On the other hand, on the last step, S_{sh} uses the simulator S_{con} of the consistency argument from Thm. 5. Since π_{con} output by S_{sh} has been computed exactly as by S_{con} in Thm. 5, the verification equation in Step 6 of Prot. 2 accepts. Moreover, as in Thm. 5, $(\hat{\boldsymbol{\mathfrak{a}}}, \mathfrak{t})$ has the same distribution as in the real protocol due to the perfect hiding of the commitment scheme, and $\boldsymbol{\mathfrak{N}}$ is uniquely fixed by $(\hat{\boldsymbol{\mathfrak{a}}}, \mathfrak{t})$ and the verification equation. Hence the simulation is perfect. □

6 Batching

The following lemma, slightly modified from [33,15] (and relying on the batching concept introduced in [4]), shows that one can often use a batched version of the verification equations.

Lemma 3. *Assume* $1 < t < q$. *Assume* $\boldsymbol{y}$ *is a vector chosen uniformly random from* $[1 \mathinner{..} t]^{k-1} \times \{1\}$, $\boldsymbol{\chi}$ *is a vector of integers in* $\mathbb{Z}_q$, *and* $(f_i)_{i \in [1 \mathinner{..} k]}$ *are some*

```
$\mathsf{V}(\mathsf{gk}, \mathsf{crs}, \mathfrak{M}; (\mathfrak{M}'_i)_{i=1}^n, (\mathfrak{a}_j)_{j=1}^{n-1}, \pi_{uv}, \pi_{sm}, \pi_{con})$:
    1. Parse $(\pi_{uv}, \pi_{sm}, \pi_{con})$ as in the prover's Steps 11–12,
    2. Compute $\mathfrak{a}_n$, $\mathfrak{b}_n$, and $\hat{\mathfrak{a}}_n$ as in the prover's Step 2,
    3. Set $(y_{1j})_{j \in [1 \,..\, n-1]} \leftarrow_r [1 \,..\, t]^{n-1}$, Set $y_{1n} \leftarrow 1$,
    4. Set $y_{21} \leftarrow_r [1 \,..\, t]$, $y_{22} \leftarrow 1$,
    5. $\alpha \leftarrow_r \mathbb{Z}_q$;
    6. Check that // Permutation matrix argument
       $\sum_{j=1}^n ((y_{1j}(\mathfrak{a}_j + \alpha[1]_1 + [P_0(\chi)]_1)) \bullet (\mathfrak{b}_j - \alpha[1]_2 + [P_0(\chi)]_2)) = (\boldsymbol{y}_1^\top \mathfrak{c}) \bullet [\varrho]_2 + (\sum_{j=1}^n y_{1j})(1 - \alpha^2)[1]_T$;
    7. Check that $(\boldsymbol{y}_1^\top \mathfrak{d}) \bullet [1]_2 = (\boldsymbol{y}_1^\top (\mathfrak{a}, \hat{\mathfrak{a}})) \bullet \left[\begin{smallmatrix} \beta \\ \hat{\beta} \end{smallmatrix}\right]_2$ // Same-message argument
    8. Set $\mathfrak{q} \leftarrow \mathfrak{t} \bullet (\mathsf{pk} \cdot \boldsymbol{y}_2)$.
    9. Check that // Consistency argument, online
       $[\hat{\boldsymbol{P}}]_1^\top \bullet (\mathfrak{M}' \boldsymbol{y}_2) - \hat{\mathfrak{a}}^\top \bullet (\mathfrak{M} \boldsymbol{y}_2) = \mathfrak{q} - [\hat{\varrho}]_1 \bullet (\mathfrak{N} \boldsymbol{y}_2)$.
```

Protocol 5: The batched verifier of the new shuffle argument.

polynomials of arbitrary degree. If there exists $i \in [1 \,..\, k]$ such that $f_i(\boldsymbol{\chi})([1]_1 \bullet [1]_2) \neq [0]_T$ then $\sum_{i=1}^k f_i(\boldsymbol{\chi}) y_i \cdot ([1]_1 \bullet [1]_2) = [0]_T$ with probability $\leq \frac{1}{t}$.

Proof. Let us define a multivariate polynomial $V(Y_1, \dots, Y_{k-1}) = \sum_{i=1}^{k-1} f_i(\boldsymbol{\chi}) Y_i + f_k(\boldsymbol{\chi}) \neq 0$. By the Schwartz–Zippel lemma $V(y_1, \dots, y_{k-1}) = 0$ with probability $\leq \frac{1}{t}$. Since $[1]_1 \bullet [1]_2 \neq [0]_T$, we get that $\sum_{i=1}^k f_i(\boldsymbol{\chi}) y_i \cdot ([1]_1 \bullet [1]_2) = [0]_T$ with probability $\leq \frac{1}{t}$. □

For the sake of concreteness, the full batched version of the verifier of the new shuffle argument is described in Prot. 5. The following corollary follows immediately from Lem. 3. It implies that if the non-batched verifier in Prot. 2 does not accept then the batched verifier in Prot. 5 only accepts with probability $\leq \frac{3}{t}$. This means that with all but negligible probability, the non-batched verifier accepts iff the batched verifier accepts. In practice a verifier could even set (say) $t = 3 \cdot 2^{40}$.

Corollary 1. *Let $1 < t < q$. Assume $\boldsymbol{\chi} = (\chi, \alpha, \beta, \hat{\beta}, \varrho, \hat{\varrho}, \mathsf{sk}) \in \mathbb{Z}_q^4 \times (\mathbb{Z}_q^*)^2 \times \mathbb{Z}_q$. Assume $(y_{1j})_{j \in [1 \,..\, n-1]}$, and y_{21} are values chosen uniformly random from $[1 \,..\, t]$, and set $y_{1n} = y_{22} = 1$.*

- *If there exists $i \in [1 \,..\, n]$ such that the ith verification equation on Step 5 of Prot. 2 does not hold, then the verification equation on Step 6 of Prot. 5 holds with probability $\leq \frac{1}{t}$.*
- *If there exists $i \in [1 \,..\, n]$ such that the ith verification equation on Step 4 of Prot. 2 does not hold, then the verification equation on Step 7 of Prot. 5 holds with probability $\leq \frac{1}{t}$.*
- *If the verification equation on Step 6 of Prot. 2 does not hold, then the verification equation on Step 9 of Prot. 5 holds with probability $\leq \frac{1}{t}$.*

Table 3. Efficiency comparison of various operations based on libsnark. Units in the last column show efficiency relative to n exponentiations in $\mathbb{G}_1$ for $n = 100,000$.

	10,000	100,000	1000,000	units
Multi-exp. $\mathbb{G}_1$	0.26s	2.54s	24.2s	0.13
Fixed-base exp. $\mathbb{G}_1$	0.28s	2.40s	18.44s	0.12
Multi-exp. $\mathbb{G}_2$	0.65s	5.54s	48.04s	0.27
Fixed-base exp. $\mathbb{G}_2$	0.75s	5.62s	44.34s	0.28
Exp. $\mathbb{G}_1$	2.15s	20.29s	207.11s	1
Exp. $\mathbb{G}_2$	5.54s	51.10s	506.26s	2.52
Pairing product	4.38s	43.37s	471.72s	2.14
Pairings	10.24s	97.07s	915.21s	4.78
Exp. $\mathbb{G}_T$	10.65s	100.20s	1110.53s	4.94

7 Implementation

We implement the new shuffle argument in C++ using the libsnark library [6]. This library provides an efficient implementation of pairings over several different elliptic curves. We run our measurements on a machine with the following specifications: (i) Intel Core i5-4200U CPU with 1.60GHz and 4 threads; (ii) 8 GB RAM; (iii) 64bit Linux Ubuntu 16.10; (iv) Compiler GCC version 6.2.0.

In addition, libsnark provides algorithms for multi-exponentiation and fixed-base exponentiation. Here, an n-wide multi-exponentiation means evaluating an expression of the form $\sum_{i=1}^{n} L_i \mathfrak{a}_i$, and n fixed-base exponentiations means evaluating an expression of the form $(L_1 \mathfrak{b}, \dots, L_n \mathfrak{b})$, where $L_i \in \mathbb{Z}_q$ and $\mathfrak{b}, \mathfrak{a}_i \in \mathbb{G}_k$. Both can be computed significantly faster than n individual scalar multiplications. (See Tbl. 3.) Importantly, most of the scalar multiplications in the new shuffle argument can be computed by using either a multi-exponentiation or a fixed-base exponentiation.

We also optimize a sum of n pairings by noting that the final exponentiation of a pairing can be done only once instead of n times. All but a constant number of pairings in our protocol can use this optimization. We use parallelization to further optimize computation of pairings and exponentiations.

To generate the CRS, we have to evaluate Lagrange basis polynomials $(\ell_i(X))_{i \in [1 \mathinner{..} n+1]}$ at $X = \chi$. In our implementation, we pick $\omega_j = j$ and make two optimizations. First, we precompute $Z(\chi) = \prod_{j=1}^{n+1}(\chi - j)$. This allows us to write $\ell_i(\chi) = Z(\chi)/((\chi - i)\prod_{j=1, j\neq i}^{n+1}(i-j))$. Second, denote $F_i = \prod_{\substack{j=1 \\ j \neq i}}^{n+1}(i-j)$. Then for $i \in [1 \mathinner{..} n]$, we have $F_{i+1} = (iF_i)/(i-n-1)$. This allows us to compute all F_i with $2n$ multiplications and n divisions. Computing all $\ell_i(\chi) = \frac{Z(\chi)}{(\chi - i)F_i}$ takes $4n+2$ multiplications and $2n+1$ divisions.

Acknowledgment. The majority of this work was done while the first author was working at the University of Tartu, Estonia. This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 653497 (project PANORAMIX) and grant agreement No 731583 (project SODA), by institutional research funding IUT2-1 of the Estonian Ministry of Education and Research, and by the Danish Independent Research Council, Grant-ID DFF-6108-00169.

References

1. Barbulescu, R., Duquesne, S.: Updating Key Size Estimations for Pairings. Technical Report 2017/334, IACR (2017) Available from `http://eprint.iacr.org/2017/334`, revision from April 26, 2017.
2. Barreto, P.S.L.M., Naehrig, M.: Pairing-Friendly Elliptic Curves of Prime Order. In: SAC 2005. LNCS, vol. 3897, pp. 319–331
3. Bayer, S., Groth, J.: Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280
4. Bellare, M., Garay, J.A., Rabin, T.: Batch Verification with Applications to Cryptography and Checking. In: LATIN 1998. LNCS, vol. 1380, pp. 170–191
5. Ben-Sasson, E., Chiesa, A., Green, M., Tromer, E., Virza, M.: Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In: IEEE SP 2015, pp. 287–304
6. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. In: USENIX 2014, pp. 781–796
7. Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the Existence of Extractable One-Way Functions. In: STOC 2014, pp. 505–514
8. Blum, M., Feldman, P., Micali, S.: Non-Interactive Zero-Knowledge and Its Applications. In: STOC 1988, pp. 103–112
9. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM **24**(2) (1981) pp. 84–88
10. Damgård, I.: Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In: CRYPTO 1991. LNCS, vol. 576, pp. 445–456
11. Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square Span Programs with Applications to Succinct NIZK Arguments. In: ASIACRYPT 2014 (1). LNCS, vol. 8873, pp. 532–550
12. Elgamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Trans. Inf. Theory **31**(4) (1985) pp. 469–472
13. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An Algebraic Framework for Diffie-Hellman Assumptions. In: CRYPTO (2) 2013. LNCS, vol. 8043, pp. 129–147
14. Fauzi, P., Lipmaa, H.: Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles. In: CT-RSA 2016. LNCS, vol. 9610, pp. 200–216
15. Fauzi, P., Lipmaa, H., Zając, M.: A Shuffle Argument Secure in the Generic Model. In: ASIACRYPT 2016 (2). LNCS, vol. 10032, pp. 841–872
16. Furukawa, J., Sako, K.: An Efficient Scheme for Proving a Shuffle. In: CRYPTO 2001. LNCS, vol. 2139, pp. 368–387
17. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for Cryptographers. Discrete Applied Mathematics **156**(16) (2008) pp. 3113–3121
18. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. In: STOC 1985, pp. 291–304
19. Golle, P., Jarecki, S., Mironov, I.: Cryptographic Primitives Enforcing Communication and Storage Complexity. In: FC 2002. LNCS, vol. 2357, pp. 120–135

20. González, A., Ràfols, C.: New Techniques for Non-interactive Shuffle and Range Arguments. In: ACNS 2016. LNCS, vol. 9696, pp. 427–444
21. Groth, J.: Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In: ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459
22. Groth, J.: A Verifiable Secret Shuffle of Homomorphic Encryptions. J. Cryptology **23**(4) (2010) pp. 546–579
23. Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340
24. Groth, J.: On the Size of Pairing-based Non-interactive Arguments. In: EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326
25. Groth, J., Lu, S.: A Non-interactive Shuffle with Pairing Based Verifiability. In: ASIACRYPT 2007. LNCS, vol. 4833, pp. 51–67
26. Groth, J., Ostrovsky, R., Sahai, A.: New Techniques for Noninteractive Zero-Knowledge. Journal of the ACM **59**(3) (2012)
27. Hess, F., Smart, N.P., Vercauteren, F.: The Eta Pairing Revisited. IEEE Trans. Inf. Theory **52**(10) (2006) pp. 4595–4602
28. Jutla, C.S., Roy, A.: Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces. In: ASIACRYPT 2013 (1). LNCS, vol. 8269, pp. 1–20
29. Jutla, C.S., Roy, A.: Switching Lemma for Bilinear Tests and Constant-Size NIZK Proofs for Linear Subspaces. In: CRYPTO (2) 2014. LNCS, vol. 8617, pp. 295–312
30. Kiltz, E., Wee, H.: Quasi-Adaptive NIZK for Linear Subspaces Revisited. In: EUROCRYPT 2015. LNCS, vol. 9057, pp. 101–128
31. Kim, T., Barbulescu, R.: Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case. In: CRYPTO 2016. LNCS, vol. 9814, pp. 543–571
32. Lipmaa, H.: Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In: TCC 2012. LNCS, vol. 7194, pp. 169–189
33. Lipmaa, H.: Prover-Efficient Commit-And-Prove Zero-Knowledge SNARKs. In: AFRICACRYPT 2016. LNCS, vol. 9646, pp. 185–206
34. Lipmaa, H., Zhang, B.: A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. In: SCN 2012. LNCS, vol. 7485, pp. 477–502
35. Maurer, U.M.: Abstract Models of Computation in Cryptography. In: Cryptography and Coding 2005, pp. 1–12
36. Morillo, P., Ràfols, C., Villar, J.L.: The Kernel Matrix Diffie-Hellman Assumption. In: ASIACRYPT 2016 (1). LNCS, vol. 10032, pp. 729–758
37. Pereira Geovandro, C.C.F., Simplício Jr., M.A., Naehrig, M., Barreto, P.S.L.M.: A Family of Implementation-Friendly BN Elliptic Curves. Journal of Systems and Software **84**(8) (2011) pp. 1319–1326
38. Schwartz, J.T.: Fast Probabilistic Algorithms for Verification of Polynomial Identities. Journal of the ACM **27**(4) (1980) pp. 701–717
39. Shoup, V.: Lower Bounds for Discrete Logarithms and Related Problems. In: EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266
40. Zippel, R.: Probabilistic Algorithms for Sparse Polynomials. In: EUROSM 1979. LNCS, vol. 72, pp. 216–226