

An Efficient Pairing-Based Shuffle Argument^{*}

Prastudy Fauzi¹, Helger Lipmaa², Janno Siim^{2,3}, and Michał Zając²

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 [Cha81] 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 k th 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 \mathbf{t} of randomizers, and sets $\mathfrak{M}'_i = \mathfrak{M}_{\sigma(i)} + \text{Enc}_{\text{pk}}(\mathbf{o}; t_i)$.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

^{*} The current version fixes a subtle flaw in the Asiacrypt 2017 version of this paper which did not satisfy the ZK property.

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

mix-server can do an incorrect shuffle, resulting in a set of decrypted votes that 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 [GMR85] that her shuffle was done correctly. The resulting proof is known as a (*zero-knowledge*) *shuffle argument*. Based on earlier work [GL07,LZ12], in CT-RSA 2016, Fauzi and Lipmaa (FL, [FL16]) proposed the then most efficient shuffle argument in the common reference string (CRS, [BFM88]) model in terms of prover's computation.⁵ Importantly, the FL shuffle argument is based on the standard Elgamal cryptosystem. The culpable soundness [GL07,GOS12] of the FL shuffle argument is proven under a knowledge assumption [Dam91] 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 [GR16] 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 Zajac (FLZ, [FLZ16]) improved on the efficiency of the FL shuffle argument by proving knowledge-soundness (not culpable soundness) in the generic bilinear group model (GBGM, [Sho97,Mau05]). 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.* [EHK⁺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 cryptosys-

⁵ Many random-oracle model shuffle arguments are known, such as [FS01,BG12,Gro10a]. 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., [BCG⁺15]

tem (amongst other things), the prover’s online complexity in the FLZ shuffle argument is more than 4 times worse (using the comparison given in [FLZ16]) 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 [FL16]) seems to be more dubious than culpable soundness achieved by the Groth-Lu [GL07] and FL shuffle arguments. (See [FL16] 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, [FLZ16] 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 [FL16], we use the ElGamal cryptosystem (with plaintexts in \mathbb{G}_2), which means that unlike [FLZ16], 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 [GL07,FL16] (but not in [LZ12,FLZ16]), 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.* [DFGK14]; it is similar to but somewhat simpler than the 1-sparsity argument of [FLZ16]. Basically, to show that a vector \mathbf{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 \mathbf{a} by using a version of the extended Pedersen commitment scheme, $\mathbf{a} = \sum_{i=1}^n a_i [P_i(\chi)]_1 + r[\varrho]_1$ for trapdoor values (χ, ϱ) and randomizer r . (This commitment scheme, though for different polynomials $P_i(X)$, was implicitly used first by Groth [Gro16] in EUROCRYPT 2016, and then used in the FLZ shuffle argument; similar commitment schemes have been used before [GJM02, Gro10b, Lip12].) The new unit vector argument differs from the corresponding (1-sparsity) argument in [FLZ16] 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 [FL16]. 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 [FLZ16].

We then use the same high-level idea as previous NIZK shuffle arguments [GL07, LZ12, FL16, FLZ16] to obtain a shuffle argument from a permutation matrix argument. Namely, we construct a verification equation that holds tautologically under a corresponding KerMDH [MRV16] 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 [GL07, LZ12, FL16, FLZ16], 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 $\hat{\mathbf{a}} = \sum a_i [\hat{P}_i(\chi)]_1 + \hat{r}[\hat{\varrho}]_1$ for trapdoor values $(\chi, \hat{\varrho})$ and randomizer \hat{r} . Here, $\hat{P}_i(X)$ are well-chosen polynomials that satisfy a small number of requirements, including that $\{P_i(X)\hat{P}_j(X)\}_{1 \leq i, j \leq n}$ is linearly independent.

Before going on, we obviously need an efficient argument (that we call, following [FL16], a *same-message argument*) to show that \mathbf{a} and $\hat{\mathbf{a}}$ commit to the same vector \mathbf{a} while using different shrinking commitment schemes and different randomizers r and \hat{r} . We first write down the objective of this argument as the requirement that $(\mathbf{a}, r, \hat{r})^\top$ belongs to a subspace generated by a certain matrix \mathbf{M} . After doing that, we use the quasi-adaptive NIZK (QANIZK, [JR13, JR14]) argument of Kiltz and Wee (EUROCRYPT 2015, [KW15]) 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 [LZ12] 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 \mathbf{E} and two tuples of ciphertexts \mathfrak{M}' and \mathfrak{M} , it holds that $\text{Dec}_{\text{sk}}(\mathfrak{M}') = \mathbf{E} \cdot \text{Dec}_{\text{sk}}(\mathfrak{M})$. Moreover, we prove that the consistency argument is culpably sound [GL07,GOS12] under a suitable KerMDH [MRV16] 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 [BGR98] to speed up verification time. However, we use batching in a more aggressive manner than in the FLZ shuffle argument.

Differences From The Conference Version. The conference version of this paper [FLSZ17] contained a flaw which allows one to break the zero-knowledge property. It assumes incorrectly that commitments $\mathbf{a} = \sum_{i=1}^n a_i [P_i(\chi)]_1 + r[\varrho]_1$ and $\hat{\mathbf{a}} = \sum a_i [\hat{P}_i(\chi)]_1 + \hat{r}[\hat{\varrho}]_1$, as were described above, are perfectly hiding even if $r = \hat{r}$. Luckily, only minimal changes are needed to incorporate independently sampled r and \hat{r} . We change the same-message argument so that instead of proving knowledge of $(\mathbf{a}, r)^\top$ for the statement $(\mathbf{a}, \hat{\mathbf{a}}) = (\mathbf{a}^\top, r) \cdot [\mathbf{M}]_1^\top$ where

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

we prove knowledge of $(\mathbf{a}, r, \hat{r})^\top$ for the statement $(\mathbf{a}, \hat{\mathbf{a}}) = (\mathbf{a}^\top, r, \hat{r}) \cdot [\mathbf{M}]_1^\top$ where

$$[\mathbf{M}]_1^\top := \begin{bmatrix} \mathbf{P} & \hat{\mathbf{P}} \\ \varrho & 0 \\ 0 & \hat{\varrho} \end{bmatrix}_1 \in \mathbb{G}_1^{(n+2) \times 2} .$$

The change causes only slight reduction in efficiency: the CRS contains one additional group element and each same-message argument requires one additional scalar multiplication in group \mathbb{G}_1 (i.e., n additional multiplications for the whole shuffle argument). We also provide a more explicit proof for the zero-knowledge property of the shuffle argument (see theorem 9) to show that the fix is sufficient.

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, [BSCTV14]. 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.

Table 1. A comparison of the new NIZK shuffle argument and prior work by Fauzi and Lipmaa (FL, [FL16]), and Fauzi, Lipmaa and Zając (FLZ, [FLZ16]). 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)
$ \text{CRS} $ in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$	$(6n + 8, 2n + 8, 1)$	$(2n + 6, n + 7, 1)$	$(4n + 8, n + 7, 1)$	$(n + 7, 4n + 8, 1)$
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 $(\mathbb{G}_1, \mathbb{G}_2)$	$(8n - 2, 2n - 2)$	$(4n - 1, 4n - 1)$	$(4n - 1, 3n - 1)$	$(3n - 1, 4n - 1)$
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	2.99	4.53
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

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 [GL07] or Lipmaa and Zhang [LZ12], since their shuffle arguments are slower than [FL16] and [FLZ16] in both prover’s and verifier’s computation. We also do not include the shuffle argument of González and Ràfols [GR16] 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 information about implementation, including the definition of units and data about the test machine.

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

	10,000	100,000	300,000		10,000	100,000	300,000
CRS generation	1.7s	13.4s	37.0s	CRS generation	2.6s	20.5s	55.0s
Prover	6.4s	56.7s	2m38.3s	Prover	9.1s	1m24.0s	4m2.4s
Prover (online)	1.1s	10.2s	32.3s	Prover (online)	0.5s	4.1s	13.5s
Verifier	8.5s	1m27.8s	5m18.8s	Verifier	8.3s	1m22.0s	4m49.2s
Verifier (online)	5.7s	1m0.5s	3m29s	Verifier (online)	5.0s	49.9s	2m55.5s

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 $\mathbf{a} = (a_i)_{i=1}^n$ we denote column vectors and by $\mathbf{a} = (a_1, \dots, a_n)$ we denote row vectors. For a matrix \mathbf{A} , \mathbf{A}_i is its i th row vector and $\mathbf{A}^{(i)}$ is its i th column vector. A Boolean $n \times n$ matrix \mathbf{A} is a permutation matrix representing $\sigma \in S_n$, when $A_{ij} = 1$ iff $\sigma(i) = j$. Clearly, in this case $\mathbf{A}\mathbf{x} = (x_{\sigma(i)})_{i=1}^n$ for any vector \mathbf{x} .

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

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 $i \in [1..n]$.
- $\ell_i(X) = \prod_{j=1, j \neq i}^{n+1} \frac{X - \omega_j}{\omega_i - \omega_j}$: the i th 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 [GPS08]. 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 , $\bullet : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, \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 [BSCTV14] which currently provides (asymmetric) Ate pairings [HSV06] over Barreto-Naehrig curves [BN05,PSNB11] with 256 bit primes. Due to recent advances in computing discrete logarithms [KB16] this curve does not guarantee 128 bits of security, but still roughly achieves 100 bits of security [BD17].

Within this paper, we use additive notation combined with the bracket notation [EHK⁺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, $ba \bullet c = (ba) \bullet c$; 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 $[\mathbf{A}]_1 \bullet [\mathbf{B}]_2 = [\mathbf{AB}]_T$ for matrices \mathbf{A} and \mathbf{B} of compatible size. We sometimes misuse the notation and, say, write $[\mathbf{A}]_2 \mathbf{B}$ instead of the more cumbersome $(\mathbf{B}^{\top} [\mathbf{A}]_2^{\top})^{\top}$. Hence, if \mathbf{A} , \mathbf{B} and \mathbf{C} have compatible dimensions, then $(\mathbf{B}^{\top} [\mathbf{A}]_1^{\top})^{\top} \bullet [\mathbf{C}]_2 = [\mathbf{A}]_1 \mathbf{B} \bullet [\mathbf{C}]_2 = [\mathbf{A}]_1 \bullet \mathbf{B} [\mathbf{C}]_2$.

Recall that a distribution \mathcal{D}_{par} that outputs matrices of group elements $[\mathbf{M}]_1 \in \mathbb{G}_1^{n \times t}$ is *witness-sampleable* [JR13], if there exists a distribution \mathcal{D}'_{par} that outputs matrices of integers $\mathbf{M}' \in \mathbb{Z}_q^{n \times t}$, such that $[\mathbf{M}]_1$ and $[\mathbf{M}']_1$ have the same distribution.

We use the Elgamal cryptosystem [Elg85] (Gen, Enc, Dec) in group \mathbb{G}_2 . In Elgamal, the key generator $\text{Gen}(1^{\kappa})$ chooses a secret key $\text{sk} \leftarrow_r \mathbb{Z}_q$ and a public key $\text{pk} \leftarrow [1, \text{sk}]_2$ for a generator $[1]_2$ of \mathbb{G}_2 fixed by BG. The encryption algorithm sets $\text{Enc}_{\text{pk}}(\mathfrak{m}; r) = ([0]_2, \mathfrak{m}) + r \cdot \text{pk}$ for $\mathfrak{m} \in \mathbb{G}_2$ and $r \leftarrow_r \mathbb{Z}_q$. The decryption algorithm sets $\text{Dec}_{\text{sk}}(\mathfrak{M}_1, \mathfrak{M}_2) = \mathfrak{M}_2 - \text{sk} \cdot \mathfrak{M}_1$. Note that $\text{Enc}_{\text{pk}}(\mathfrak{o}; r) = r \cdot \text{pk}$. The Elgamal cryptosystem is blindable, with $\text{Enc}_{\text{pk}}(\mathfrak{m}; r_1) + \text{Enc}_{\text{pk}}(\mathfrak{o}; r_2) = \text{Enc}_{\text{pk}}(\mathfrak{m}; r_1 + r_2)$. 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 $[\mathfrak{b}]_2 \in \mathbb{G}_2^{1 \times 2}$, let $[a]_1 \circ [\mathfrak{b}]_2 := [a \cdot \mathfrak{b}]_T = [a \cdot b_1, a \cdot b_2]_T \in \mathbb{G}_T^{1 \times 2}$. Analogously, for $[\mathfrak{a}]_1 \in \mathbb{G}_1^n$, and $[\mathbf{B}]_2 \in \mathbb{G}_2^{n \times 2}$, let $[\mathfrak{a}]_1^{\top} \circ [\mathbf{B}]_2 := \sum_{i=1}^n [a_i]_1 \circ [\mathbf{B}_i]_2 \in \mathbb{G}_T^n$. Intuitively, here $[\mathfrak{b}]_2$ is an Elgamal ciphertext and $[\mathbf{B}]_2$ is a vector of Elgamal ciphertexts.

Kernel Matrix Assumptions [MRV16]. Let $k \in \mathbb{N}$. We call $\mathcal{D}_{k+d,k}$ a *matrix distribution* [EHK⁺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 $\mathbf{A} \leftarrow \mathcal{D}_{k+d,k}$ 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 [MRV16] holds in \mathbb{G}_z relative to algorithm BG, if for all probabilistic polynomial-time \mathcal{A} ,

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

By varying the distribution $\mathcal{D}_{k+d,k}$, one can obtain various assumptions (such as the SP assumption of Groth and Lu [GL07] or the PSP assumption of Fauzi and Lipmaa [FL16]) 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 $(\mathbf{M}, \text{aux}; \text{td}) \leftarrow_r \mathcal{D}_{n+d,d}$. 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} ,*

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

Commitment Schemes. A (pairing-based) trapdoor commitment scheme is a pair of efficient algorithms (K, com) , where $\text{K}(\mathbf{gk})$ (for $\mathbf{gk} \leftarrow \text{BG}(1^\kappa)$) outputs a commitment key ck and a trapdoor td , and the commitment algorithm outputs $\mathbf{a} \leftarrow \text{com}(\mathbf{gk}, \text{ck}, \mathbf{a}; r)$. A commitment scheme is *computationally binding* if for any ck output by K , it is computationally infeasible to find $(\mathbf{a}, r) \neq (\mathbf{a}', r')$, such that $\text{com}(\mathbf{gk}, \text{ck}, \mathbf{a}; r) = \text{com}(\mathbf{gk}, \text{ck}, \mathbf{a}'; r')$. A commitment scheme is *perfectly hiding* if for any ck output by K , the distribution of the output of $\text{com}(\mathbf{gk}, \text{ck}, \mathbf{a}; r)$ does not depend on \mathbf{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 [Gro16]. Fix some linearly independent polynomials $P_i(X)$. Let $\mathbf{gk} \leftarrow \text{BG}(1^\kappa)$, and $\text{td} = (\chi, \varrho) \leftarrow_r \mathbb{Z}_q^2$. Denote $\mathbf{P} = (P_i(\chi))_{i=1}^n$. Let $\text{ck} \leftarrow [\frac{\mathbf{P}}{\varrho}]_1$, and $\text{com}(\mathbf{gk}, \text{ck}, \mathbf{a}; r) := (\frac{\mathbf{a}}{r})^\top \cdot \text{ck} = [\sum_{i=1}^n a_i P_i(\chi) + r\varrho]_1$. 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 [GJM02,Gro10b,Lip12]. 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 [MRV16].

Theorem 1. *Let $\mathcal{D}_n^{((P_i(X))_{i=1}^n, X_\varrho)} = \{[\frac{P}{\varrho}]_1 : (\chi, \varrho) \leftarrow_r \mathbb{Z}_q \times \mathbb{Z}_q^*\}$ 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 $\text{td} = (\chi, \varrho)$.*

Proof (Sketch). Given two different openings (\mathbf{a}, r) and (\mathbf{a}', r') to a commitment, $(\mathbf{a} - \mathbf{a}', r - r')$ 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 (\mathbf{a}, r) , one can open $\text{com}(\text{gk}, \text{ck}, \mathbf{a}; r)$ to (\mathbf{a}', r') by taking $r' = r + (\sum_{i=1}^n (a_i - a'_i) P_i(\chi))/\varrho$. \square

Generic Bilinear Group Model. A *generic* algorithm uses only generic group operations to create and manipulate group elements. Shoup [Sho97] formalized it by giving algorithms access to random injective encodings $\langle\langle \mathbf{a} \rangle\rangle$ instead of real group elements \mathbf{a} . Maurer [Mau05] 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 $F_1(\mathbf{X})$ and $F_2(\mathbf{X})$ attached to them. This is motivated by the Schwartz-Zippel lemma [Sch80,Zip79] that states that if $F_1(\mathbf{X}) \neq F_2(\mathbf{X})$ as a polynomial, then there is negligible probability that $F_1(\chi) = F_2(\chi)$ for uniformly random χ .

Zero Knowledge. Let $\mathcal{R} = \{(u, w)\}$ be an efficiently computable binary relation with $|w| = \text{poly}(|u|)$. 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 \mathbf{gk} (output by BG) as an input to \mathcal{L}_n and \mathcal{R}_n . Let $\mathcal{R}_n(\mathbf{gk}) := \{(u, w) : (\mathbf{gk}, u, w) \in \mathcal{R}_n\}$.

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 $\mathbf{gk} \leftarrow (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{g}_1, \mathbf{g}_2, \bullet)$. For $\mathbf{gk} \leftarrow \text{BG}(1^\kappa)$ and $(\text{crs}, \text{td}) \leftarrow \text{K}(\mathbf{gk}, n)$ (where n is input length), $\text{P}(\mathbf{gk}, \text{crs}, u, w)$ produces an argument π , and $\text{V}(\mathbf{gk}, \text{crs}, u, \pi)$ outputs either 1 (accept) or 0 (reject). The verifier may be probabilistic, to speed up verification time by the use of batching techniques [BGR98].

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

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

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

$$\Pr \left[\begin{array}{l} \mathbf{gk} \leftarrow \text{BG}(1^\kappa), (\text{crs}, \text{td}) \leftarrow \text{K}(\mathbf{gk}, n), \\ (u, \pi) \leftarrow \mathcal{A}(\mathbf{gk}, \text{crs}) : (\mathbf{gk}, u) \notin \mathcal{L}_n \wedge \text{V}(\mathbf{gk}, \text{crs}, u, \pi) = 1 \end{array} \right] \approx_\kappa 0 .$$

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* [GL07,GOS12] for \mathcal{L} using a polynomial-time decidable binary relation $\mathcal{R}^{\text{glt}} = \{\mathcal{R}_n^{\text{glt}}\}$ consisting of elements from $\bar{\mathcal{L}}$ and witnesses w^{glt} , if for all $n = \text{poly}(\kappa)$ and all non-uniform probabilistic polynomial-time adversaries \mathcal{A} ,

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

For algorithms \mathcal{A} and $\text{Ext}_{\mathcal{A}}$, we write $(y; y') \leftarrow (\mathcal{A} \parallel \text{Ext}_{\mathcal{A}})(\chi)$ if \mathcal{A} on input χ outputs y , and $\text{Ext}_{\mathcal{A}}$ on the same input (including the random tape of \mathcal{A})

outputs y' . Ψ is *knowledge-sound*, if for all $n = \text{poly}(\kappa)$ and all non-uniform probabilistic polynomial-time adversaries \mathcal{A} , there exists a non-uniform probabilistic polynomial-time extractor $\text{Ext}_{\mathcal{A}}$, such that for every auxiliary input $\text{aux} \in \{0, 1\}^{\text{poly}(\kappa)}$,

$$\Pr \left[\begin{array}{l} \text{gk} \leftarrow \text{BG}(1^\kappa), (\text{crs}, \text{td}) \leftarrow \text{K}(\text{gk}, n), ((u, \pi); w) \leftarrow (\mathcal{A} \parallel \text{Ext}_{\mathcal{A}})(\text{crs}, \text{aux}) : \\ (\text{gk}, u, w) \notin \mathcal{R}_n \wedge \text{V}(\text{gk}, \text{crs}, u, \pi) = 1 \end{array} \right] \approx_{\kappa} 0 .$$

Here, aux can be seen as the common auxiliary input to \mathcal{A} and $\text{Ext}_{\mathcal{A}}$ that is generated by using benign auxiliary input generation [BCPR14].

Ψ is *perfectly (composable) zero-knowledge* [Gro06], if there exists a probabilistic polynomial-time simulator S , such that for all stateful non-uniform probabilistic adversaries \mathcal{A} and $n = \text{poly}(\kappa)$, $\varepsilon_0 = \varepsilon_1$, where

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

Ψ is *perfectly witness-indistinguishable*, if for all stateful non-uniform probabilistic adversaries \mathcal{A} and $n = \text{poly}(\kappa)$, $\varepsilon_0 = \varepsilon_1$, where

$$\varepsilon_b := \Pr \left[\begin{array}{l} \text{gk} \leftarrow \text{BG}(1^\kappa), (\text{crs}, \text{td}) \leftarrow \text{K}(\text{gk}, n), (u, w_0, w_1) \leftarrow \mathcal{A}(\text{gk}, \text{crs}), \\ \pi \leftarrow \text{P}(\text{gk}, \text{crs}, u, w_b) : \\ (\text{gk}, u, w_0) \in \mathcal{R}_n \wedge (\text{gk}, u, w_1) \in \mathcal{R}_n \wedge \mathcal{A}(\text{gk}, \text{crs}, u, \pi) = 1 \end{array} \right] .$$

Shuffle Argument. In a (pairing-based) shuffle argument [GL07], 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:

$$\mathcal{R}_{sh} = \left\{ \left((\text{gk}, \text{pk}, \mathfrak{M}, \mathfrak{M}'), (\sigma, \mathbf{r}) \right) : \sigma \in S_n \wedge \mathbf{r} \in \mathbb{Z}_q^n \wedge (\forall i : \mathfrak{M}'_i = \mathfrak{M}_{\sigma(i)} + \text{Enc}_{\text{pk}}(\mathbf{0}; r_i)) \right\} .$$

A number of pairing-based shuffle arguments have been proposed in the literature, [GL07, LZ12, FL16, FLZ16, GR16].

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 \mathbf{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* [FL16]. In Sect. 4.1 we motivate this name, by showing that if the same-message argument accepts, then the prover knows a message \mathbf{a} and randomizers r and \hat{r} , such that $\mathbf{a}_i = [\sum a_i P_i(\chi) + r \varrho]_1$ and $\hat{\mathbf{a}}_i = [\sum a_i \hat{P}_i(\chi) + \hat{r} \hat{\varrho}]_1$ both commit to \mathbf{a} 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 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 \mathbf{a} and π_{pm} , and (ii) the verifier computes \mathbf{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 $(\mathbf{a}_1, \dots, \mathbf{a}_n)$ 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 [FL16] — where the verifier performs the verification Step 5 for exactly one i .

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

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

$\mathbf{crs}_{con} \leftarrow [\hat{\varrho}]_1$. Set $\mathbf{crs} \leftarrow (\mathbf{pk} = [1, \mathbf{sk}]_2, [\frac{P}{\varrho}]_1, [\frac{P}{\varrho}]_2, \mathbf{crs}_{sm}, \mathbf{crs}_{pm}, \mathbf{crs}_{con})$. Set $\mathbf{td} \leftarrow (\chi, \hat{\varrho})$. Return $(\mathbf{crs}, \mathbf{td})$.

$\mathbf{P}(\mathbf{gk}, \mathbf{crs}, \mathfrak{M} \in \mathbb{G}_2^{n \times 2}; \sigma \in S_n, \mathbf{t} \in \mathbb{Z}_q^n)$:

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

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

$\mathbf{V}(\mathbf{gk}, \mathbf{crs}, \mathfrak{M}; \mathfrak{M}', (\mathbf{a}_j)_{j=1}^{n-1}, \pi_{sm}, \pi_{pm}, \pi_{con})$:

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

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

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 [FLZ16], the verifier adds $[\alpha]_1 + [P_0]_1$ from the CRS to \mathbf{a}_i (and adds $[\alpha]_2 - [P_0]_2$ from the CRS to \mathbf{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 $\hat{\mathbf{a}}(\{\hat{P}_i(X)\}, X_{\hat{\rho}})$ -commits to a permutation, then $\text{Dec}(\mathfrak{M}'_i) = \text{Dec}(\mathfrak{M}_{\sigma(i)})$ 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 [MRV16] 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 $\chi = (\chi, \alpha, \beta, \hat{\beta}, \rho, \hat{\rho}, \text{sk})$ be the tuple of all random values generated by either \mathbf{K} or \mathbf{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 χ will be an indeterminate. Let us denote the tuple of corresponding indeterminates by $\mathbf{X} = (X, X_\alpha, X_\beta, X_{\hat{\beta}}, X_\rho, X_{\hat{\rho}}, X_S)$.

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 \mathbf{X} . E.g., for the CRS element $[\beta P_i(\chi) + \hat{\beta} \hat{P}_i(\chi)]_1$, she knows that the attached polynomial is $X_\beta P_i(X) + X_{\hat{\beta}} \hat{P}_i(X)$. Each element output by the adversary can hence be seen as a polynomial in \mathbf{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 $\text{crs}_z(\mathbf{X}, T, t)$ for symbolic values

T and t :

$$\begin{aligned} \text{crs}_1(\mathbf{X}, T, t) &= t(X) + T_0 P_0(X) + T_\varrho X_\varrho + T^\dagger(X) Z(X) / X_\varrho + T^*(X) + \\ &\quad T_{\hat{\varrho}} X_{\hat{\varrho}} + \sum_{i=1}^n T_{\beta, i} (X_\beta P_i(X) + X_{\hat{\beta}} \hat{P}_i(X)) + T_{\beta_\varrho} X_\beta X_\varrho + T_{\hat{\beta}_{\hat{\varrho}}} X_{\hat{\beta}} X_{\hat{\varrho}}, \\ \text{crs}_2(\mathbf{X}, T, t) &= t(X) + T_0 P_0(X) + T_\varrho X_\varrho + T_S X_S + T_\beta X_\beta + T_{\hat{\beta}} X_{\hat{\beta}}, \end{aligned}$$

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 $\{P_i(X)\}_{i=1}^n$.

4.1 Same-Message Argument

For the sake of this argument, let $\mathbf{P}(X) = (P_i(X))_{i=1}^n$ and $\hat{\mathbf{P}}(X) = (\hat{P}_i(X))_{i=1}^n$ 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 $\mathbf{a}, \hat{\mathbf{a}} \in \mathbb{G}_1$, she knows \mathbf{a} , r , and \hat{r} , such that $\begin{pmatrix} \mathbf{a} \\ \hat{\mathbf{a}} \end{pmatrix} = [\mathbf{M}]_1 \begin{pmatrix} \mathbf{a} \\ r \\ \hat{r} \end{pmatrix}$ for

$$[\mathbf{M}]_1^\top := \begin{bmatrix} \mathbf{P} & \hat{\mathbf{P}} \\ \varrho & 0 \\ 0 & \hat{\varrho} \end{bmatrix}_1 \in \mathbb{G}_1^{(n+2) \times 2} \quad (1)$$

and $\text{ck} := [\mathbf{P}]_1$ and $\hat{\text{ck}} := [\hat{\mathbf{P}}]_1$. That is, \mathbf{a} and $\hat{\mathbf{a}}$ are commitments to the same vector \mathbf{a} , 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 [KW15] for the *linear subspace*

$$\mathcal{L}_{\mathbf{M}} = \left\{ \begin{pmatrix} \mathbf{a} \\ \hat{\mathbf{a}} \end{pmatrix} : \exists \begin{pmatrix} \mathbf{a} \\ r \\ \hat{r} \end{pmatrix} \in \mathbb{Z}_q^{n+2} : \begin{pmatrix} \mathbf{a} \\ \hat{\mathbf{a}} \end{pmatrix} = [\mathbf{M}]_1 \begin{pmatrix} \mathbf{a} \\ r \\ \hat{r} \end{pmatrix} \right\}.$$

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 \mathbf{w} such that $(\mathbf{a}, \hat{\mathbf{a}}) = \mathbf{w}^\top \cdot [\mathbf{M}]_1^\top$. Moreover, we need it to stay knowledge-sound even when the adversary has access to an auxiliary input.

More precisely, denote by $\mathcal{D}_{n+2,2}^{sm}$ the distribution of matrices \mathbf{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, $\mathcal{D}_{n+2,2}^{sm}$ is witness-sampleable.

For a matrix $\mathbf{A} \in \mathbb{Z}_q^{(k+1) \times k}$, let $\bar{\mathbf{A}} \in \mathbb{Z}_q^{k \times k}$ denote the upper square matrix of \mathbf{A} . The same-message argument (i.e., the Kiltz-Wee QANIZK argument that

$(\begin{smallmatrix} \mathbf{a} \\ \hat{\mathbf{a}} \end{smallmatrix}) = [\mathbf{M}]_1 \left(\begin{smallmatrix} \mathbf{a} \\ \hat{\mathbf{a}} \end{smallmatrix} \right)$ for witness-sampleable distributions is depicted as follows:

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

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

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

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

Proof. ZERO KNOWLEDGE: follows from $\pi_{sm} = \mathbf{w}^\top [\mathbf{Q}]_1 = \mathbf{w}^\top [\mathbf{M}^\top \mathbf{K}]_1 = [\mathbf{y}^\top \mathbf{K}]_1$.

KNOWLEDGE-SOUNDNESS: In the generic group model, the adversary knows polynomials $A(\mathbf{X}) = \text{crs}_1(\mathbf{X}, A, a)$, $\hat{A}(\mathbf{X}) = \text{crs}_1(\mathbf{X}, \hat{A}, \hat{a})$, and $\pi(\mathbf{X}) = \text{crs}_1(\mathbf{X}, \Pi, \pi)$, such that $\mathbf{a} = [A(\chi)]_1$, $\hat{\mathbf{a}} = [\hat{A}(\chi)]_1$, $\pi_{sm} = [\pi(\chi)]_1$.

Because the verification accepts, by the Schwartz–Zippel lemma, from this it follows (with all but negligible probability) that $\pi(\mathbf{X}) = X_\beta A(\mathbf{X}) + X_{\hat{\beta}} \hat{A}(\mathbf{X})$ 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) $X_\beta X_\varrho$ and $X_{\hat{\beta}} X_{\hat{\varrho}}$. Thus, we must have that for some \mathbf{a} , r , and \hat{r} ,

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

Hence, $A(\mathbf{X}) = \sum_{i=1}^n a_i P_i(X) + r X_\varrho$ and $\hat{A}(\mathbf{X}) = \sum_{i=1}^n a_i \hat{P}_i(X) + \hat{r} X_{\hat{\varrho}}$. Thus, \mathbf{a} and $\hat{\mathbf{a}}$ commit to the same vector \mathbf{a} . \square

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

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 \mathbf{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 [FLZ16].

Unit Vector Argument. In a unit vector argument [FL16], the prover aims to convince the verifier that he knows how to open a commitment \mathbf{a} to (\mathbf{a}, r) , such that exactly one coefficient a_I , $I \in [1..n]$, is equal to 1, while other coefficients of \mathbf{a} are equal to 0. Recall [FL16,FLZ16] that if we define $\mathbf{V} := \begin{pmatrix} 2 \cdot I_{n \times n} \\ \mathbf{1}_n^\top \end{pmatrix} \in \mathbb{Z}_q^{(n+1) \times n}$ and $\mathbf{b} := \begin{pmatrix} \mathbf{0}_n \\ 1 \end{pmatrix} \in \mathbb{Z}_q^{n+1}$, then \mathbf{a} is a unit vector iff

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

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

Similar to the 1-sparsity argument in [FLZ16], we construct the unit vector argument by using a variant of square span programs (SSP-s, [DFGK14]). To proceed, we need to define the following polynomials. For $i \in [1..n]$, set $P_i(X)$ to be the degree n polynomial that interpolates the i th column of the matrix \mathbf{V} , i.e.,

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

Set

$$P_0(X) := \sum_{i=1}^n -\ell_i(X) = \ell_{n+1}(X) - 1 \quad , \quad (4)$$

i.e., $P_0(X)$ is the polynomial that interpolates $\mathbf{b} - \mathbf{1}_{n+1}$. 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 $i \in [1..n+1]$, which holds iff $Z(X) \mid Q(X)$, [FLZ16].

Lemma 1 ([FL16,FLZ16]). *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

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

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:

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

This argument is similar to the 1-sparsity argument presented in [FLZ16], 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 [FLZ16], 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 [FLZ16] 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*

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

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

Proof. **COMPLETENESS:** For an honest prover, and $I = \sigma^{-1}(j)$, $\mathbf{a}_j = [P_I(\chi) + r_j \varrho]_1$, $\mathbf{b}_j = [P_I(\chi) + r_j \varrho]_2$, and $\mathbf{c}_j = [r_j(2(P_I(\chi) + r_j \varrho + P_0(\chi)) - r_j \varrho) + h(\chi)Z(\chi)/\varrho]_1$ 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 $\mathbf{a} = [A(\chi)]_1$, $\mathbf{b} = [B(\chi)]_2$, $\mathbf{c} = [C(\chi)]_2$, for attached polynomials $A(\mathbf{X}) = crs_1(\mathbf{X}, A, a)$, $B(\mathbf{X}) = crs_2(\mathbf{X}, B, b)$, and $C(\mathbf{X}) = crs_1(\mathbf{X}, C, c)$, such that verification in Step 5 of Prot. 2 accepts. Observing this verification equation, it

is easy to see that for the polynomial $V_{uv}(\mathbf{X}) := (A(\mathbf{X}) + X_\alpha + P_0(X)) \cdot (B(\mathbf{X}) - X_\alpha + P_0(X)) - C(\mathbf{X}) \cdot X_\varrho - (1 - X_\alpha^2)$, the verification equation assesses that $V_{uv}(\mathbf{X}) = 0$. Since we are in the GBGM, the adversary knows all coefficients of $A(\mathbf{X})$, $B(\mathbf{X})$, $C(\mathbf{X})$, and $V_{uv}(\mathbf{X})$.

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

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

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

$\overline{M = X_\varrho}$: $C_M = -Z(X)c^\dagger(X) - 1 + (a(X) + P_0(X))(b(X) + (B_0 + 1)P_0(X)) = 0$: 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 $I_j \in [1..n]$, 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 two witnesses $w_1 = (\mathbf{a}_1, r_1)$ and $w_2 = (\mathbf{a}_2, r_2)$ of the same statement \mathbf{a} . This means that $\sum a_{1,i} P_i + r_1 \delta = \sum a_{2,i} P_i + r_2 \delta$, which implies that $[\sum a_{1,i} P_i + r_1 \delta]_2 = \mathbf{b} = [\sum a_{2,i} P_i + r_2 \delta]_2$. Moreover, \mathbf{c} is uniquely fixed by (\mathbf{a}, \mathbf{b}) and the verification equation. Hence an accepting argument that uses witness w_1 has the same distribution as one that uses w_2 .

Remark 2. While a slight variant of this subargument was proposed in [FLZ16], they did not consider its privacy separately. It is easy to see that neither the new argument nor the 1-sparsity argument [FLZ16] is zero-knowledge in the case of type-III pairings. Knowing the witness, the prover can produce \mathbf{b}_j such that $\mathbf{a}_j \bullet [1]_2 = [1]_1 \bullet \mathbf{b}_j$. On the other hand, given an arbitrary input $\mathbf{a} \in \mathbb{G}_1$ as input, the simulator cannot construct such \mathbf{b} since there is no efficient isomorphism from \mathbb{G}_1 to \mathbb{G}_2 . Witness-indistinguishability suffices in our application. \square

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, [LZ12]. To guarantee that the matrix is left stochastic, it suffices to compute \mathbf{a}_n explicitly, i.e., $\mathbf{a}_n = [\sum_{j=1}^n P_j(\chi)]_1 - \sum_{j=1}^{n-1} \mathbf{a}_j$. After that, we need to perform the unit vector argument on every \mathbf{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 $j \in [1..n]$, and \mathbf{a}_n is explicitly computed as in Prot. 2, then Ext outputs $(\sigma \in S_n, \mathbf{r})$, such that for all $j \in [1..n]$, $\mathbf{a}_j = [P_{\sigma^{-1}(j)}(\chi) + r_j \varrho]_1$.*

Proof. KNOWLEDGE-SOUNDNESS follows from the explicit construction of \mathbf{a}_n , and from the fact that we have a generic adversary that knows all the coefficients, and thus also knows (σ, \mathbf{r}) . More precisely, from the knowledge-soundness of the unit vector argument, for each $j \in [1..n]$ there exists an extractor that outputs (\mathbf{a}_j, r_j) such that \mathbf{a}_j is a unit vector with a 1 at some position $I_j \in [1..n]$ and $\mathbf{a}_j = [P_{I_j}(\chi) + r_j \varrho]_1$. Since $[\sum_{j=1}^n P_{I_j}(\chi)]_1 = \sum_{j=1}^n \mathbf{a}_j = [\sum_{j=1}^n P_j(\chi)]_1$, 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 $[1..n]$, so $(\mathbf{a}_j)_{j=1}^n$ is a permutation matrix.

WITNESS-INDISTINGUISHABILITY follows from the witness-indistinguishability of the unit vector argument. \square

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 $\mathbf{E} \in \mathbb{Z}_q^{n \times n}$ and two tuples $(\mathfrak{M}$ and \mathfrak{M}') of Elgamal ciphertexts, it holds that $\text{Dec}_{\text{sk}}(\mathfrak{M}') = \mathbf{E} \cdot \text{Dec}_{\text{sk}}(\mathfrak{M})$. More precisely,

$$\mathcal{R}_{\text{con},n} = \left\{ (\text{gk}, \text{pk}, (\mathfrak{M}, \mathfrak{M}'), (\mathbf{E}, t)) : \mathfrak{M}' = \mathbf{E}\mathfrak{M} + (t_i \cdot \text{pk})_{i=1}^n \right\}.$$

Since we use a shrinking commitment scheme, each $\hat{\mathbf{a}}$ can commit to any matrix \mathbf{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 proven in Sect. 3) that, after the same-message and the permutation matrix argument, extracts \mathbf{E} and randomizer vector $\hat{\mathbf{r}}$, such that $\hat{\mathbf{a}} = \left(\frac{\mathbf{E}}{\hat{\mathbf{r}}^\top} \right)^\top \left[\frac{\hat{\mathbf{P}}}{\hat{\varrho}} \right]_1$. Hence, the guilt relation is

$$\mathcal{R}_{\text{con},n}^{\text{gt}} = \left\{ (\text{gk}, \text{pk}, (\mathfrak{M}, \mathfrak{M}'), (\mathbf{E}, \hat{\mathbf{r}}, \hat{\mathbf{a}})) : \hat{\mathbf{a}} = \left(\frac{\mathbf{E}}{\hat{\mathbf{r}}^\top} \right)^\top \left[\frac{\hat{\mathbf{P}}}{\hat{\varrho}} \right]_1 \wedge \text{Dec}_{\text{sk}}(\mathfrak{M}') \neq \mathbf{E} \cdot \text{Dec}_{\text{sk}}(\mathfrak{M}) \right\}.$$

We note that in the case of a shuffle argument, \mathbf{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 $\text{td}_{con} = (\chi, \hat{\rho})$, works as follows. Note that the prover should be able to work without knowing the ElGamal secret key, and that $[\mathbf{M}]_1 = \text{ck}$.

$\text{K}_{con}(\text{gk}, n)$: Return $(\text{crs}_{con}; \text{td}) = ([\mathbf{M}]_1 = [\hat{\rho}]_1, \text{aux}; (\chi, \hat{\rho})) \leftarrow \mathcal{D}_{n+1,1}$.
 $\text{P}_{con}(\text{gk}, \text{crs}, (\mathfrak{M}, \mathfrak{M}'), (\mathbf{E}, \mathbf{t}))$: // $\mathfrak{M}' = \mathbf{E}\mathfrak{M} + (t_i \cdot \text{pk})_{i=1}^n$
 $\hat{\mathbf{r}} \leftarrow_r \mathbb{Z}_q^n$; $\hat{\mathbf{a}} \leftarrow (\frac{\mathbf{E}}{\hat{\mathbf{r}}^\top})^\top [\hat{\rho}]_1$; $r_t \leftarrow_r \mathbb{Z}_q$; $\mathbf{t} \leftarrow \mathbf{t}^\top [\hat{\rho}]_1 + r_t [\hat{\rho}]_1$; $\mathfrak{N} \leftarrow \hat{\mathbf{r}}^\top \mathfrak{M} + r_t \cdot \text{pk}$; Return $(\pi_{con} \leftarrow (\hat{\mathbf{a}}, \mathbf{t}, \mathfrak{N}))$.
 $\text{V}_{con}(\text{gk}, \text{crs}, (\mathfrak{M}, \mathfrak{M}'), \pi_{con})$: Check that $[\hat{\rho}]_1^\top \circ \mathfrak{M}' - \hat{\mathbf{a}}^\top \circ \mathfrak{M} \stackrel{?}{=} \mathbf{t} \circ \text{pk} - [\hat{\rho}]_1 \circ \mathfrak{N}$.

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 $[\mathbf{M}]_1 = [\hat{\rho}]_1 \in \mathbb{Z}_q^{n+1}$ for $(\chi, \hat{\rho}) \leftarrow_r \mathbb{Z}_q \times \mathbb{Z}_q^*$. 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 $\mathcal{R}_{con}^{\text{glt}}$ with the CRS $\text{crs} = ([\mathbf{M}]_1, \text{aux})$.*

Proof. **PERFECT COMPLETENESS:** In the case of the honest prover, $[\hat{\rho}]_1^\top \circ \mathfrak{M}' - \hat{\mathbf{a}}^\top \circ \mathfrak{M} = [\hat{\rho}]_1^\top \circ (\mathbf{E}\mathfrak{M} + (t_i \cdot \text{pk})_{i=1}^n) - (\frac{\mathbf{E}}{\hat{\mathbf{r}}^\top})^\top [\hat{\rho}]_1^\top \circ \mathfrak{M} = [\hat{\rho}]_1^\top \circ ((t_i \cdot \text{pk})_{i=1}^n) + r_t [\hat{\rho}]_1 \circ \text{pk} - [\hat{\rho}]_1 \circ r_t \text{pk} - [\hat{\rho}]_1 \circ \hat{\mathbf{r}}^\top \mathfrak{M} = \mathbf{t} \circ \text{pk} - [\hat{\rho}]_1 \circ \mathfrak{N}$.

CULPABLE SOUNDNESS: Assume that \mathcal{A}_{con} is an adversary that, given input $(\text{gk}, \text{crs}_{con})$ for $\text{crs}_{con} = (\mathbf{M}, \text{aux}) \leftarrow_r \mathcal{D}_n^{con}$, returns $u = (\mathfrak{M}, \mathfrak{M}')$, $\pi_{con} = (\hat{\mathbf{a}}, \mathbf{t}, \mathfrak{N})$ and $w^{\text{glt}} = (\mathbf{E}, \hat{\mathbf{r}}, \hat{\mathbf{a}})$. \mathcal{A}_{con} succeeds iff (i) the verifier of the consistency argument accepts, (ii) $\hat{\mathbf{a}} = (\frac{\mathbf{E}}{\hat{\mathbf{r}}^\top})^\top [\hat{\rho}]_1$, and (iii) $\text{Dec}_{\text{sk}}(\mathfrak{M}') \neq \mathbf{E} \cdot \text{Dec}_{\text{sk}}(\mathfrak{M})$. 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 $(\text{gk}, [\mathbf{M}]_1, \text{aux})$ where $\text{gk} \leftarrow \text{BG}(1^\kappa)$ and $\text{crs}_{ker} = ([\mathbf{M}]_1, \text{aux}) \leftarrow_r \mathcal{D}_n^{con}$, and is supposed to output $[c]_2$, such that $\mathbf{M}^\top \mathbf{c} = \mathbf{0}$ but $\mathbf{c} \neq \mathbf{0}$.

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

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

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

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 $[M]_1^\top \bullet \begin{pmatrix} \mathbf{m}' \\ \mathbf{n} \end{pmatrix} - \hat{\mathbf{a}}^\top \bullet \mathbf{m} = [0]_T$.

Since \mathcal{A}_{con} is successful, then $\hat{\mathbf{a}} = \begin{pmatrix} E \\ \hat{\mathbf{r}}^\top \end{pmatrix}^\top [M]_1$, hence we get

$$\begin{aligned} [0]_T &= [M]_1^\top \bullet \begin{pmatrix} \mathbf{m}' \\ \mathbf{n} \end{pmatrix} - [M]_1^\top \begin{pmatrix} E \\ \hat{\mathbf{r}}^\top \end{pmatrix} \bullet \mathbf{m} \\ &= [M]_1^\top \bullet \begin{pmatrix} \mathbf{m}' \\ \mathbf{n} \end{pmatrix} - [M]_1^\top \bullet \begin{pmatrix} E \\ \hat{\mathbf{r}}^\top \end{pmatrix} \mathbf{m} \\ &= [M]_1^\top \bullet \begin{pmatrix} \mathbf{m}' - E\mathbf{m} \\ \mathbf{n} - \hat{\mathbf{r}}^\top \mathbf{m} \end{pmatrix}. \end{aligned}$$

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

PERFECT ZERO-KNOWLEDGE: The simulator $S_{con}(\mathbf{gk}, \text{crs}, (\mathfrak{M}, \mathfrak{M}'), \text{td}_{con} = (\chi, \hat{\rho}))$ creates random commitments $\hat{\mathbf{a}}$ and \mathbf{t} , then uses the trapdoor to pick \mathfrak{N} such that the verification equation would be satisfied. More precisely, the simulator sets $\mathbf{a} \leftarrow_r \mathbb{Z}_q^n, \mathbf{t} \leftarrow_r \mathbb{Z}_q, \hat{\mathbf{a}} \leftarrow [a]_1, \mathbf{t} \leftarrow [t]_1$, and $\mathfrak{N} \leftarrow (t/\hat{\rho}) \cdot \text{pk} - (\hat{P}/\hat{\rho})^\top \mathfrak{M}' + (\mathbf{a}/\hat{\rho})^\top \mathfrak{M}$. The simulator S_{con} then outputs $\pi_{con} \leftarrow (\hat{\mathbf{a}}, \mathbf{t}, \mathfrak{N})$.

Due to the perfect hiding property of the commitment scheme, $(\hat{\mathbf{a}}, \mathbf{t})$ has the same distribution as in the real protocol, and \mathfrak{N} is the uniquely determined by $(\hat{\mathbf{a}}, \mathbf{t})$ and the verification equation. Moreover,

$$\begin{aligned} [\hat{P}]_1^\top \circ \mathfrak{M}' - \hat{\mathbf{a}}^\top \circ \mathfrak{M} &= [\hat{P}]_1^\top \circ \mathfrak{M}' - [a]_1^\top \circ \mathfrak{M} \\ &= [\hat{\rho}]_1 \circ (\hat{P}/\hat{\rho})^\top \mathfrak{M}' - [\hat{\rho}]_1 \circ (\mathbf{a}/\hat{\rho})^\top \mathfrak{M} \\ &= [\hat{\rho}]_1 \circ (t/\hat{\rho} \cdot \text{pk} - t/\hat{\rho} \cdot \text{pk} + (\hat{P}/\hat{\rho})^\top \mathfrak{M}' - (\mathbf{a}/\hat{\rho})^\top \mathfrak{M}) \\ &= \mathbf{t} \circ \text{pk} - [\hat{\rho}]_1 \circ \mathfrak{N} \end{aligned}$$

and thus this choice of \mathfrak{N} makes the verifier accept. Hence we have a perfect simulation. \square

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

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

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

$\{P_i(X)\hat{P}_j(X)\}_{1 \leq i \leq n} \cup \{\hat{P}_j(X)\}$ and $\{P_i(X)\hat{P}_k(X)\}_{1 \leq i \leq n} \cup \{\hat{P}_k(X)\}$ only intersect at 0. This holds since for non-zero vectors (\mathbf{a}, \mathbf{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 $(k+1)(n+1)$. \square

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,
- $\{P_i(X)\hat{P}_j(X)\}_{1 \leq i, j \leq n}$ is linearly independent,

Assume also that the second input \mathbf{aux} output by $\mathcal{D}_n^{\text{con}}$ satisfies the following property. The subset \mathbf{aux}_2 of \mathbb{G}_2 -elements in \mathbf{aux} must satisfy that

- \mathbf{aux}_2 does not depend on $\hat{\rho}$,
- the only element in \mathbf{aux}_2 that depends on ρ is $[\rho]_2$,
- the only elements in \mathbf{aux}_2 that depend on χ are $[P_i(\chi)]_2$ for $i \in [0..n]$.

Then in the GBGM, the $\mathcal{D}_n^{\text{con}}$ -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 \mathbf{aux}) satisfy the required properties.

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

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

$$\hat{V}_{\text{ker}}(\mathbf{X}) := \sum_{i=1}^n \hat{P}_i(X)M_i(\mathbf{X}) + X_{\hat{\rho}} \cdot N(\mathbf{X}) \quad , \quad (7)$$

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

Due to the assumptions on \mathbf{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- $(\leq n)$ polynomials, we can write

$$M_i(\mathbf{X}) = \sum_{j=1}^n M_{ij}P_j(X) + M'_i X_{\hat{\rho}} + m_i(\mathbf{X}) \quad ,$$

where $m_i(\mathbf{X})$ does not depend on X , $X_{\hat{\rho}}$ or $X_{\hat{\rho}}$.

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

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

$$\sum_{i=1}^n \sum_{j=1}^n M_{ij} \hat{P}_i(X)P_j(X) + X_{\hat{\rho}} \sum_{i=1}^n M'_i \hat{P}_i(X) + \sum_{i=1}^n \hat{P}_i(X)m_i(\mathbf{X}) = 0 \quad .$$

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

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

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. \square

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

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

Let $\mathcal{D}_n^{\text{con}}$ be as before with $([M]_1, \text{aux})$, such that aux is equal to the CRS in Prot. 1 minus the elements already in $[M]_1$. If the \mathcal{D}_1 -KerMDH assumption holds in \mathbb{G}_2 , the $\mathcal{D}_n^{((P_i(X))_{i=1}^n, X_e)}$ -KerMDH and $\mathcal{D}_n^{((\hat{P}_i(X))_{i=1}^n, X_e)}$ -KerMDH assumptions hold and the $\mathcal{D}_n^{\text{con}}$ -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 $\{P_i(X) \hat{P}_j(X)\}_{1 \leq i, j \leq n}$, we get straightforwardly that $\{\hat{P}_j(X)\}_{j=1}^n$ is also linearly independent. We construct the following simple sequence of games.

GAME₀: 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 (gk, crs) , \mathcal{A}_{sh} outputs $u = (\mathfrak{M}, \mathfrak{M}')$ and an accepting proof π , such that for all permutations σ , there exists $i \in [1..n]$, such that $\text{Dec}_{\text{sk}}(\mathfrak{M}'_i) \neq \text{Dec}_{\text{sk}}(\mathfrak{M}_{\sigma(i)})$.

GAME₁: Here, we let \mathcal{A}_{con} herself generate a new secret key $\text{sk} \leftarrow_{\mathcal{r}} \mathbb{Z}_q$ and set $\text{pk} \leftarrow [1, \text{sk}]_2$. Since the distribution of pk is witness-sampleable, then for any (even omnipotent) adversary \mathcal{B} , $|\Pr[\text{GAME}_1(\mathcal{B}) = 1] - \Pr[\text{GAME}_0(\mathcal{B}) = 1]| = 0$ (i.e., GAME₁ and GAME₀ are indistinguishable).

Now, consider GAME₁ 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 $\text{Ext}_{sm:i}^A$ and Ext_{pm}^A such that the following holds:

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

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

- If the same-message argument verifier accepts $(\mathbf{a}_i, \hat{\mathbf{a}}_i, \pi_{sm:i}) \leftarrow \mathcal{A}_{sm}(\mathbf{gk}, \text{crs}_{sh}; r')$, then $\text{Ext}_{sm:i}^A(\mathbf{gk}, \text{crs}_{sh}; r')$ outputs the witness $(\mathbf{E}'_i, r_i, \hat{r}_i)$ where (\mathbf{E}'_i, r_i) is the opening of \mathbf{a}_i and $(\mathbf{E}'_i, \hat{r}_i)$ is the opening of $\hat{\mathbf{a}}_i$.
- If the permutation matrix argument verifier accepts $((\mathbf{a}_i)_{i=1}^{n-1}, \pi_{pm}) \leftarrow \mathcal{A}_{pm}(\mathbf{gk}, \text{crs}_{sh}; r')$, then $\text{Ext}_{pm}^A(\mathbf{gk}, \text{crs}_{sh}; r')$ outputs a permutation matrix \mathbf{E} and vector \mathbf{r} such that $\mathbf{a} = [\mathbf{E}^\top \mathbf{P} + \mathbf{r} \varrho]_1$ (here, \mathbf{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 $\mathcal{R}_{con}^{\text{gt}}$ with auxiliary input. \mathcal{A}_{con} gets an input $(\mathbf{gk}, \text{crs}_{con})$ where $\mathbf{gk} \leftarrow \text{BG}(1^\kappa)$, and $\text{crs}_{con} \leftarrow ([\mathbf{M}]_1, \text{aux}) \leftarrow_r \mathcal{D}_n^{\text{con}}$. \mathcal{A}_{con} is supposed to output $(u, \pi_{con}, w^{\text{gt}})$ where $u = (\mathfrak{M}, \mathfrak{M}')$, $\pi_{con} = (\hat{\mathbf{a}}, \mathbf{t}, \mathfrak{N})$, and $w^{\text{gt}} = (\mathbf{E}, \hat{\mathbf{r}}, \hat{\mathbf{a}})$ such that $\hat{\mathbf{a}} = [\mathbf{E}^\top \hat{\mathbf{P}} + \hat{\mathbf{r}} \hat{\varrho}]_1$, π_{con} is accepting, and $\text{Dec}_{sk}(\mathfrak{M}') \neq \mathbf{E} \cdot \text{Dec}_{sk}(\mathfrak{M})$.

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, $\text{Ext}_{sm:i}^A$ 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 \mathbf{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. \square

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

Proof. We present an explicit proof to be sure that composing subprotocols reveals no additional information.

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

Protocol 4: The simulator of the shuffle argument

We will construct a simulator S_{sh} , that given a CRS crs and trapdoor $\text{td} = (\chi, \hat{\varrho})$ simulates the prover in Prot. 1. S_{sh} takes any pair $(\mathfrak{M}, \mathfrak{M}')$ as input and performs the steps in Prot. 4. We argue that each element in the simulated proof has the same distribution as in the honest proof:

1. $(\mathbf{a}_i)_{i=1}^{n-1}$, $(\hat{\mathbf{a}}_i)_{i=1}^{n-1}$, and \mathbf{t} are perfectly hiding commitments both in the honest and in the simulated proof. More precisely, they are uniformly random and independent group elements.
2. For $i \in [1 .. n - 1]$, \mathbf{b}_i has the same discrete logarithm as \mathbf{a}_i both in the honest proof and in the simulated proof.
3. The simulator computes \mathbf{a}_n , \mathbf{b}_n , and $\hat{\mathbf{a}}_n$ in the same way as the honest prover. I.e., they are uniquely determined respectively by $(\mathbf{a}_i)_{i=1}^{n-1}$, $(\mathbf{b}_i)_{i=1}^{n-1}$, and $(\hat{\mathbf{a}}_i)_{i=1}^{n-1}$.
4. Fixing the above elements, there is exactly one value for \mathbf{c}_i , \mathbf{d}_i , and \mathfrak{M} such that the respective verification equations would pass. The simulator computes \mathbf{c}_i and \mathbf{d}_i just as the honest prover does, hence the verification equations for the unit vector argument and the same-message argument will hold by the completeness property. Finally, the simulator computes \mathfrak{M} in the same way as in Thm. 5 where we already showed that it makes the consistency verification equation pass.

We conclude that elements output by S_{sh} have the same distribution as in the honest proof. \square

$V(\text{gk}, \text{crs}, \mathfrak{M}; (\mathfrak{M}'_i)_{i=1}^n, (\mathbf{a}_j)_{j=1}^{n-1}, \pi_{uv}, \pi_{sm}, \pi_{con})$:

1. Parse $(\pi_{uv}, \pi_{sm}, \pi_{con})$ as in the prover's Steps 11–12,
2. Compute \mathbf{a}_n , \mathbf{b}_n , and $\hat{\mathbf{a}}_n$ 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

$$\sum_{j=1}^n ((y_{1j}(\mathbf{a}_j + \alpha[1]_1 + [P_0(\chi)]_1)) \bullet (\mathbf{b}_j - \alpha[1]_2 + [P_0(\chi)]_2)) = (\mathbf{y}_1^\top \mathbf{c}) \bullet [\varrho]_2 + (\sum_{j=1}^n y_{1j})(1 - \alpha^2)[1]_T$$
;
7. Check that $(\mathbf{y}_1^\top \mathfrak{D}) \bullet [1]_2 = (\mathbf{y}_1^\top (\mathbf{a}, \hat{\mathbf{a}})) \bullet \begin{bmatrix} \beta \\ \hat{\beta} \end{bmatrix}_2$ // Same-message argument
8. Set $\mathfrak{q} \leftarrow \mathbf{t} \bullet (\text{pk} \cdot \mathbf{y}_2)$.
9. Check that // Consistency argument, online

$$[\hat{\mathbf{P}}]_1^\top \bullet (\mathfrak{M}' \mathbf{y}_2) - \hat{\mathbf{a}}^\top \bullet (\mathfrak{M} \mathbf{y}_2) = \mathfrak{q} - [\hat{\varrho}]_1 \bullet (\mathfrak{N} \mathbf{y}_2)$$
.

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

6 Batching

The following lemma, slightly modified from [Lip16,FLZ16] (and relying on the batching concept introduced in [BGR98]), shows that one can often use a batched version of the verification equations.

Lemma 3. *Assume $1 < t < q$. Assume \mathbf{y} is a vector chosen uniformly random from $[1..t]^{k-1} \times \{1\}$, χ is a vector of integers in \mathbb{Z}_q , and $(f_i)_{i \in [1..k]}$ are some polynomials of arbitrary degree. If there exists $i \in [1..k]$ such that $f_i(\chi)([1]_1 \bullet [1]_2) \neq [0]_T$ then $\sum_{i=1}^k f_i(\chi)y_i \cdot ([1]_1 \bullet [1]_2) = [0]_T$ with probability $\leq \frac{1}{t}$.*

Proof. Let us define a multivariate polynomial $V(Y_1, \dots, Y_{k-1}) = \sum_{i=1}^{k-1} f_i(\chi)Y_i + f_k(\chi) \neq 0$. 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 $\sum_{i=1}^k f_i(\chi)y_i \cdot ([1]_1 \bullet [1]_2) = [0]_T$ with probability $\leq \frac{1}{t}$. \square

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 $\chi = (\chi, \alpha, \beta, \hat{\beta}, \varrho, \hat{\varrho}, \text{sk}) \in \mathbb{Z}_q^4 \times (\mathbb{Z}_q^*)^2 \times \mathbb{Z}_q$. 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 i th 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 i th verification equation on Step 4 of Prot. 2 does not hold, then the verification equation on Step 7 of Prot. 5 holds with probability $\leq \frac{1}{i}$.
- 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 $\leq \frac{1}{i}$.

7 Implementation

We implement the new shuffle argument in C++ using the `libsark` library [BSCTV14]. 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, `libsark` provides algorithms for multi-exponentiation and fixed-base exponentiation. Here, an n -wide multi-exponentiation means evaluating an expression of the form $\sum_{i=1}^n L_i \mathbf{a}_i$, and n fixed-base exponentiations means evaluating an expression of the form $(L_1 \mathbf{b}, \dots, L_n \mathbf{b})$, where $L_i \in \mathbb{Z}_q$ and $\mathbf{b}, \mathbf{a}_i \in \mathbb{G}_k$. 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 $(\ell_i(X))_{i \in [1..n+1]}$ 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 $F_i = \prod_{j=1, j \neq i}^{n+1} (i - j)$. Then for $i \in [1..n]$, 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.

Table 3. Efficiency comparison of various operations based on `libsark`. 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

References

- BCG⁺15. Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In *IEEE SP 2015*, pages 287–304, San Jose, CA, USA, May 17–21, 2015. IEEE Computer Society.
- BCPR14. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the Existence of Extractable One-Way Functions. In David Shmoys, editor, *STOC 2014*, pages 505–514, New York, NY, USA, May 31 – Jun 3, 2014. ACM Press.
- BD17. Razvan Barbulescu and Sylvain Duquesne. Updating Key Size Estimations for Pairings. Technical Report 2017/334, IACR, April 14, 2017. Available from <http://eprint.iacr.org/2017/334>, revision from April 26, 2017.
- BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-Interactive Zero-Knowledge and Its Applications. In *STOC 1988*, pages 103–112, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.
- BG12. Stephanie Bayer and Jens Groth. Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg.
- BGR98. Mihir Bellare, Juan A. Garay, and Tal Rabin. Batch Verification with Applications to Cryptography and Checking. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN 1998*, volume 1380 of *LNCS*, pages 170–191, Campinas, Brazil, April 20–24, 1998. Springer, Heidelberg.
- BN05. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. In Bart Preneel and Stafford E. Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331, Kingston, ON, Canada, August 11–12, 2005. Springer, Heidelberg.
- BSCTV14. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. In Kevin Fu and Jaeyeon Jung, editors, *USENIX 2014*, pages 781–796, San Jose, CA, USA, August 20–22, 2014. USENIX Association.
- Cha81. David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

- Dam91. Ivan Damgård. Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In Joan Feigenbaum, editor, *CRYPTO 1991*, volume 576 of *LNCS*, pages 445–456, Santa Barbara, California, USA, August 11–15, 1991. Springer, Heidelberg, 1992.
- DFGK14. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square Span Programs with Applications to Succinct NIZK Arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014 (1)*, volume 8873 of *LNCS*, pages 532–550, Kaohsiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg.
- EHK⁺13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge L. Villar. An Algebraic Framework for Diffie-Hellman Assumptions. In Ran Canetti and Juan Garay, editors, *CRYPTO (2) 2013*, volume 8043 of *LNCS*, pages 129–147, Santa Barbara, California, USA, August 18–22, 2013. Springer, Heidelberg.
- Elg85. Taher Elgamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Trans. Inf. Theory*, 31(4):469–472, 1985.
- FL16. Prastudy Fauzi and Helger Lipmaa. Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 200–216, San Francisco, CA, USA, February 29–March 4, 2016. Springer, Heidelberg.
- FLSZ17. Prastudy Fauzi, Helger Lipmaa, Janno Siim, and Michał Zając. An Efficient Pairing-Based Shuffle Argument. In Thomas Peyrin and Tsuyoshi Takagi, editors, *ASIACRYPT 2017*, volume ? of *LNCS*, pages ?–?, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg.
- FLZ16. Prastudy Fauzi, Helger Lipmaa, and Michał Zając. A Shuffle Argument Secure in the Generic Model. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016 (2)*, volume 10032 of *LNCS*, pages 841–872, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg.
- FS01. Jun Furukawa and Kazue Sako. An Efficient Scheme for Proving a Shuffle. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387, Santa Barbara, USA, August 19–23, 2001. Springer, Heidelberg.
- GJM02. Philippe Golle, Stanislaw Jarecki, and Ilya Mironov. Cryptographic Primitives Enforcing Communication and Storage Complexity. In Matt Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 120–135, Southampton Beach, Bermuda, March 11–14, 2002. Springer, Heidelberg.
- GL07. Jens Groth and Steve Lu. A Non-interactive Shuffle with Pairing Based Verifiability. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 51–67, Kuching, Malaysia, December 2–6, 2007. Springer, Heidelberg.
- GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof-Systems. In Robert Sedgwick, editor, *STOC 1985*, pages 291–304, Providence, Rhode Island, USA, May 6–8, 1985. ACM Press.
- GOS12. Jens Groth, Rafail Ostrovsky, and Amit Sahai. New Techniques for Non-interactive Zero-Knowledge. *Journal of the ACM*, 59(3), 2012.
- GPS08. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for Cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
- GR16. Alonso González and Carla Ràfols. New Techniques for Non-interactive Shuffle and Range Arguments. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 2016*, volume 9696 of *LNCS*, pages 427–444, Guildford, UK, June 19–22, 2016. Springer, Heidelberg.

- Gro06. Jens Groth. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459, Shanghai, China, December 3–7, 2006. Springer, Heidelberg.
- Gro10a. Jens Groth. A Verifiable Secret Shuffle of Homomorphic Encryptions. *J. Cryptology*, 23(4):546–579, 2010.
- Gro10b. Jens Groth. Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340, Singapore, December 5–9, 2010. Springer, Heidelberg.
- Gro16. Jens Groth. On the Size of Pairing-based Non-interactive Arguments. In Marc Fischlin and Jean-Sebastien Coron, editors, *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 305–326, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg.
- HSV06. Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The Eta Pairing Revisited. *IEEE Trans. Inf. Theory*, 52(10):4595–4602, 2006.
- JR13. Charanjit S. Jutla and Arnab Roy. Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013 (1)*, volume 8269 of *LNCS*, pages 1–20, Bangalore, India, December 1–5, 2013. Springer, Heidelberg.
- JR14. Charanjit S. Jutla and Arnab Roy. Switching Lemma for Bilinear Tests and Constant-Size NIZK Proofs for Linear Subspaces. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO (2) 2014*, volume 8617 of *LNCS*, pages 295–312, Santa Barbara, California, USA, August 17–21, 2014. Springer, Heidelberg.
- KB16. Taechan Kim and Razvan Barbulescu. Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016*, volume 9814 of *LNCS*, pages 543–571, Santa Barbara, California, USA, August 14–18, 2016. Springer, Heidelberg.
- KW15. Eike Kiltz and Hoeteck Wee. Quasi-Adaptive NIZK for Linear Subspaces Revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015*, volume 9057 of *LNCS*, pages 101–128, Sofia, Bulgaria, May 26–30, 2015. Springer, Heidelberg.
- Lip12. Helger Lipmaa. Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189, Taormina, Italy, March 18–21, 2012. Springer, Heidelberg.
- Lip16. Helger Lipmaa. Prover-Efficient Commit-And-Prove Zero-Knowledge SNARKs. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 2016*, volume 9646 of *LNCS*, pages 185–206, Fes, Morocco, April 13–15, 2016. Springer, Heidelberg.
- LZ12. Helger Lipmaa and Bingsheng Zhang. A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. In Ivan Visconti and Roberto De Prisco, editors, *SCN 2012*, volume 7485 of *LNCS*, pages 477–502, Amalfi, Italy, September 5–7, 2012. Springer, Heidelberg.
- Mau05. Ueli M. Maurer. Abstract Models of Computation in Cryptography. In Nigel P. Smart, editor, *Cryptography and Coding 2005*, pages 1–12, Cirencester, UK, December 19–21, 2005.
- MRV16. Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The Kernel Matrix Diffie-Hellman Assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors,

- ASIACRYPT 2016 (1)*, volume 10032 of *LNCS*, pages 729–758, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg.
- PSNB11. C. C. F. Pereira Geovandro, Marcos A. Simplício Jr., Michael Naehrig, and Paulo S. L. M. Barreto. A Family of Implementation-Friendly BN Elliptic Curves. *Journal of Systems and Software*, 84(8):1319–1326, 2011.
- Sch80. Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, 27(4):701–717, 1980.
- Sho97. Victor Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In Walter Fumy, editor, *EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 256–266, Konstanz, Germany, 11–15 May 1997. Springer, Heidelberg.
- Zip79. Richard Zippel. Probabilistic Algorithms for Sparse Polynomials. In Edward W. Ng, editor, *EUROSM 1979*, volume 72 of *LNCS*, pages 216–226, Marseille, France, June 1979. Springer, Heidelberg.