A Shuffle Argument Secure in the Generic Model

Prastudy Fauzi, Helger Lipmaa, and Michał Zając

University of Tartu, Estonia

Abstract. We propose a new random oracle-less NIZK shuffle argument. It has a simple structure, where the first verification equation ascertains that the prover has committed to a permutation matrix, the second verification equation ascertains that the same permutation was used to permute the ciphertexts, and the third verification equation ascertains that input ciphertexts were "correctly" formed. The new argument has 3.5 times more efficient verification than the up-to-now most efficient shuffle argument by Fauzi and Lipmaa (CT-RSA 2016). Compared to the Fauzi-Lipmaa shuffle argument, we (i) remove the use of knowledge assumptions and prove our scheme is sound in the generic bilinear group model, and (ii) prove standard soundness, instead of culpable soundness.

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

1 Introduction

A typical application of mix-nets is in e-voting, where each voter (assume that there are n of them) encrypts his ballot by using an additively homomorphic public-key cryptosystem, and sends it to the bulletin board. After the vote casting period has ended, the bulletin board (considered to be the 0th, non-mixing, mix-server) forwards all encrypted ballots to the first mix-server. A small number (say, M) of mix-servers are ordered sequentially. The kth mix-server obtains a tuple $\mathfrak{v}$ of input ciphertexts from the $(k-1)$th mix-server, shuffles them, and sends a tuple $\mathfrak{v}'$ of output ciphertexts to the $(k+1)$th mix-server. Shuffling means that the kth mix-server generates a random permutation $\sigma \leftarrow_r S_n$ and a vector $\boldsymbol{s}$ of randomizers, and sets $\mathfrak{v}'_i = \mathfrak{v}_{\sigma(i)} \cdot \mathsf{enc}_{\mathsf{pk}}(0; s_i)$. The last mix-server (the $(M+1)$th one, usually implemented by using multi-party computation) is again a non-mixing server, who instead decrypts the results.

A mix-net clearly preserves the anonymity of voters, if at least one of the participating mix-servers is honest. To achieve security against an active attack (where some of the shuffles were not done correctly) is more difficult. In a nutshell, each server should prove in zero knowledge [24] that her shuffle was done correctly, i.e., prove that there exists a permutation σ and a vector $\boldsymbol{s}$, such that $\mathfrak{v}'_i = \mathfrak{v}_{\sigma(i)} \cdot \mathsf{enc}_{\mathsf{pk}}(0; s_i)$ for each i. The resulting zero-knowledge proof is usually called a *(zero-knowledge) shuffle argument*.

Moreover, to obtain active security of the whole mix-net, it is important that the outputs of incorrect shuffles are ignored. This means that each mix-server

(including the $(M+1)$th one) has to verify the correctness of each previous mix-server, and only apply her own shuffle to the output of the (multi-)shuffle where each previous server has been correct. Intuitively, this means that the verification time is the real bottleneck of mix-nets.

Substantial amount of work has been done on interactive zero-knowledge shuffle arguments. Random oracle model shuffle arguments are already quite efficient, see, e.g., [25]. However, an ever-growing amount of research [12, 23, 36, 6] has provided evidence that the random oracle model yields properties that are impossible to achieve in the standard model. (See [14] for recent progress on NIZK arguments in the random oracle model.)

Much less is known about shuffle arguments in the common reference string (CRS, [7]) model, without using random oracles. Based on earlier work [28, 33], Fauzi and Lipmaa recently proposed a shuffle argument in the CRS model [19]. Assuming that basic group operations are as efficient in both cases, and that a pairing is about 8 times slower than a group exponentiation (both assumptions should be taken with a caveat), the Fauzi-Lipmaa shuffle is about two times less efficient for the prover than the most efficient known shuffle argument in the random oracle model [25], while its verification is about 25 times less efficient.

The security of the Fauzi-Lipmaa shuffle argument is proven under a knowledge assumption [15] (PKE, [26]) and three computational assumptions (PCDH, TSDH, PSP). Knowledge assumptions are non-falsifiable [35], and their validity has to be very carefully checked in each application [5]. Moreover, the PSP assumption of Fauzi and Lipmaa [19] is novel (albeit closely related to SP, an earlier assumption of Groth and Lu [28]), and its security is proven in the generic bilinear group model [38, 34, 8].

The Fauzi-Lipmaa shuffle differs from the shuffle of Lipmaa and Zhang [33] in its security model. Briefly, in the security proof of the Lipmaa-Zhang shuffle argument it is assumed that the adversary obtains — by using knowledge assumptions — not only the secrets of the possibly malicious mix-server, but also the plaintexts and randomizers computed by all voters. This model was called *white-box soundness* by Fauzi and Lipmaa [19], where it was also criticized. Moreover, in the Lipmaa-Zhang shuffle argument [32], the plaintexts have to be small for the soundness proof to go through; for this, all voters should use efficient CRS-model range proofs [13, 20, 31].

On the other hand, the Fauzi-Lipmaa shuffle is proven culpably sound [28] though also under knowledge assumptions. Intuitively, culpable soundness means that if a cheating adversary produces an invalid (yet acceptable) shuffle together with the secret key, then one can break one of the underlying knowledge or computational assumptions.

Our Contribution. In all three results mentioned above [28, 33, 19], the authors based the soundness of their shuffle argument on some novel hardness assumptions, and then proved that the assumptions are secure in the generic bilinear group model (GBGM). It seems to be an obvious question whether one can obtain some efficiency benefit by bypassing the intermediate assumption and

proving the soundness of the shuffle argument directly in the GBGM. We show this is indeed the case. We improve on the efficiency of the previous CRS-based shuffle arguments by proving the security of our protocol in the GBGM and *without* using knowledge assumptions. Due to the use of GBGM, we must first define a sensible security model.

First, recall that in the GBGM, the adversary inputs some group elements $\mathfrak{G}_i = \mathfrak{g}^{\chi_i}$, where $\mathfrak{g}$ is a group generator and χ_i are various (not necessarily independent) random values. One assumes that each group element $\mathfrak{H}_j$ output by the adversary is of the form $\mathfrak{H}_j = \mathfrak{g}_z^{F_j(\boldsymbol{\chi})}$, where $F_j(\boldsymbol{X})$ are known linear polynomials and $\mathfrak{g}_z$ is a generator of the group $\mathbb{G}_z$, $z \in \{1, 2\}$. (Within this paper, $\boldsymbol{\chi}$ is a concrete instantiation of the indeterminate $\boldsymbol{X}$.) We call such values *admissible*. (In addition, elements output from the target group can also use the bilinear map, but in the current paper, we do not use this fact.)

One philosophical question when using the GBGM is what exactly is the input of the adversary. In our intended usage cases, the shuffle argument is a part of a mix-net. Clearly, the mix-net should remain secure against coalitions between parties (in the case of e-voting, either voters, or some of the mix-servers themselves) that create the input ciphertexts, and parties who perform the shuffling. It is a common practice to model such coalitions as a single adversary. In the GBGM, it is natural to model this single adversary — who may corrupt everybody who has produced any part of the input to the verifier — as a generic adversary. This means that an adversary, who has generated a (say, ILin [18]) ciphertext $\mathfrak{v}_i = (\mathfrak{v}_{i1}, \mathfrak{v}_{i2}, \mathfrak{v}_{i3})$, knows polynomials $V_{ij}(\boldsymbol{X})$ and $V'_{ij}(\boldsymbol{X})$, such that $\log \mathfrak{v}_{ij} = V_{ij}(\boldsymbol{\chi})$ and $\log \mathfrak{v}'_{ij} = V'_{ij}(\boldsymbol{\chi})$. This is somewhat similar to the approach taken in [33] who used knowledge assumptions to then obtain the random variables — more precisely, plaintexts and randomizers — hidden in $\mathfrak{v}$.

We will assume that the mix-net is structured as follows. First, the encrypters (e.g., voters) prove that their ciphertexts (e.g., *encrypted* ballots) are admissible. More precisely, by using a *validity argument*, a voter proves that each component (e.g., an ILin [18] ciphertext consists of three group elements) of her ciphertext is equal to $\mathfrak{g}_1^{F(\boldsymbol{\chi})}$, where the polynomial $F(\boldsymbol{X})$ has specific form. The validity argument guarantees that the input ciphertexts to the first mix-server have been computed only from certain, "allowed", elements of the CRS.

Each mix-server first verifies the validity of original (unshuffled) ciphertexts and the soundness of each previous shuffle argument. After that the mix-server produces her shuffle $(\mathfrak{v}'_i)_{i=1}^n$ together with her shuffle argument π_{sh}. This means that we consider shuffling a part of the shuffle argument.

Our generic approach in the shuffle argument is as follows. We first let the prover (a mix-server) choose a permutation matrix and then commit separately to its every row. The prover then proves that the committed matrix is a permutation matrix, by proving that each row is 1-sparse (i.e., it has at most one non-zero element) as in [33], while computing the last row explicitly, see Sect. 5. The 1-sparsity argument is based loosely on Square Span Programs [16]. Basically, to show that a vector $\boldsymbol{a}$ is 1-sparse, we construct $n + 1$ polynomials $(P_i(X))_{i=0}^n$ that interpolate a certain matrix (and a certain vector) connected

to the definition of 1-sparsity, and then commit to $\boldsymbol{a}$ by using a "polynomial" version of the extended Pedersen commitment scheme, $\mathfrak{c} \leftarrow \mathfrak{g}_2^{\sum a_i P_i(\chi)+r\varrho}$, for random secrets χ and ϱ.

To obtain the full shuffle argument, we use the same underlying idea as [28, 33, 19]. Namely, we construct a specific *consistency* verification equation that ensures that $(\mathfrak{v}_i)_{i=1}^n$ is permuted to $(\mathfrak{v}'_i)_{i=1}^n$ by using the same permutation matrix that was used to permute $(\mathfrak{g}_2^{P_i(\chi)})_{i=1}^n$ to $(\mathfrak{A}_{i2})_{i=1}^n$. This is done by using a pairing equation of type $\prod \hat{e}(\mathfrak{v}'_i, \mathfrak{g}_2^{P_i(\chi)}) / \prod \hat{e}(\mathfrak{v}_i, \mathfrak{A}_{i2}) = \mathfrak{R}$, where $\mathfrak{R}$ is a value that takes care of the rerandomization (i.e., it depends on the values $\boldsymbol{s}$ used to rerandomize $\mathfrak{v}$, but not on σ).

Both [28] and [19] had an additional problem here, namely it can be the case that a maliciously created $\mathfrak{v}'_i$ depends on $P_j(X)$ (in [28], one has $P_j(X_1, \ldots, X_n) = X_j$, where X_j are independent random variables) so $\log_{\mathfrak{g}_T} \hat{e}(\mathfrak{v}'_i, \mathfrak{g}_2^{P_i(\chi)})$ can depend on $P_j(X)P_i(X)$, for arbitrary i and j. In this case, this equation is not sufficient for soundness, since $\{P_i(X)P_j(X)\}_{i,j\in[1\,..\,n]}$ is not linearly independent (e.g., an adversary can cancel out $P_j(X)P_i(X)$ easily with $-P_i(X)P_j(X)$). Therefore, they had to go through additional complicated steps — that reduced the efficiency of their arguments — to achieve (culpable) soundness even in this case.

In our case, such complications are not needed, due to the validity argument. Since the validity argument guarantees that $\mathfrak{v}_i$ and $\mathfrak{v}'_i$ do not depend on $P_i(X)$, it means that the values $\log_{\mathfrak{g}_T} \hat{e}(\mathfrak{v}'_i, \mathfrak{g}_2^{P_i(\chi)})$ and $\log_{\mathfrak{g}_T} \hat{e}(\mathfrak{v}_i, \mathfrak{A}_{i2})$ do not depend on $P_i(X)P_j(X)$, which removes the problem evident in both [28] and [19]. On the other hand, [28, 19] solved this problem by proving culpable soundness only, while we prove that the new argument satisfies the standard soundness property.

We emphasize that the full GBGM soundness proof of the new shuffle argument is quite intricate. In particular, the verification of the permutation matrix argument results in a system of more than 20 polynomial equations. As some other recent papers like [3, 1], we use computer-based tools to solve the latter system. More precisely, we use a computer algebra system to find its Gröbner basis [11], and then continue to find solutions from there on. It is interesting that a simple shuffle argument has such a complicated security proof. On the other hand, both researchers and practitioners can write their own computer algebra code to verify the security proof; this is not possible in many other arguments.

We further optimize the verification by the use of batching techniques [4], thus replacing many pairings with less costly exponentiations. Batching has not been used before in the context of pairing-based shuffle arguments.

Tbl. 1 compares our work and known NIZK shuffle arguments in the CRS model. However, differently from other papers, [28] uses symmetric pairings, and thus its computational and communication complexity is not directly comparable. The prover's computational complexity and the communication includes the computation and sending of the ciphertexts themselves. (This is fair, since different shuffle arguments use different public-key cryptosystems that incur different overhead to these complexity measures.) The highlighted cells in each row are the

Table 1. A comparison of different NIZK shuffle arguments. We always consider shuffling to be a part of the communication and prover's computation. Units (the main parameter, a weighted sum of other parameters) are defined in Sect. 9

.

	Groth-Lu	Lipmaa-Zhang	Fauzi-Lipmaa	Current Work
Type of pairings	Symmetric	Asymmetric		
|CRS| in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$	$2n+8$	$(2n+2, 5n+4, 0)$	$(6n+8, 2n+8, 1)$	$(2n+6, n+7, 1)$
Communication	$18n+120$	$(8n+6, 4n+5, 0)$	$(7n+2, 2n, 0)$	$(4n+1, 3n+2, 0)$
Prover's computation				
Exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$54n+246$	$(16n+6, 12n+5)$	$(14n+3, 4n)$	$(9n+2, 9n+3)$
Units		36	19.8	24.3
Verifier's computation				
Exp. in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$	—	—	—	$(11n+5, 3n+6, 1)$
Pairings	$75n+282$	$28n+18$	$18n+6$	$3n+6$
Units		196	126	36.3
Knowl. assumpt-s	No	Yes	Yes	No
Relying on GBGM	PP, SP	Knowledge	Knowl., PSP	Complete
Random oracle	No			
Soundness	Culpable	Full	Culpable	Full

values with best efficiency, or best security properties. A more precise efficiency comparison is given in Sect. 9.

Finally, each of the CRS-model shuffle arguments relies substantially on the GBGM. The Groth-Lu and Fauzi-Lipmaa shuffles rely on the GBGM to prove security of complicated computational assumptions. The Lipmaa-Zhang shuffle relies on the GBGM to prove security of non-falsifiable knowledge assumptions. The current paper gives the full shuffle soundness proof in the GBGM. See Sect. 10 for a more thorough discussion of the GBGM security proof versus using knowledge assumptions.

2 Preliminaries

Let S_n be the symmetric group on n elements. For a (Laurent) polynomial or a rational function f and its monomial μ, denote by $\mathrm{coeff}_\mu(f)$ the coefficient of μ in f. We write $f(\kappa) \approx_\kappa g(\kappa)$, if $f(\kappa) - g(\kappa)$ is negligible as a function of κ.

Bilinear Maps. Let κ be the security parameter. Let q be a prime of length $O(\kappa)$ bits. Assume we use a secure bilinear group generator $\mathsf{genbp}(1^\kappa)$ that returns $\mathsf{gk} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three multiplicative groups of order q, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Within this paper, we denote the elements of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ as in $\mathfrak{g}_1$ (i.e., by using the Fraktur typeface). It is required that $\hat{e}$ is bilinear (i.e., $\hat{e}(\mathfrak{g}_1^a, \mathfrak{g}_2^b) = \hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^{ab}$), efficiently computable, and non-degenerate. We define $\hat{e}((\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3), \mathfrak{B}) = (\hat{e}(\mathfrak{A}_1, \mathfrak{B}), \hat{e}(\mathfrak{A}_2, \mathfrak{B}), \hat{e}(\mathfrak{A}_3, \mathfrak{B}))$ and $\hat{e}(\mathfrak{B}, (\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3)) = (\hat{e}(\mathfrak{B}, \mathfrak{A}_1), \hat{e}(\mathfrak{B}, \mathfrak{A}_2), \hat{e}(\mathfrak{B}, \mathfrak{A}_3))$. Assume that $\mathfrak{g}_i$ is a generator of $\mathbb{G}_i$ for $i \in \{1, 2\}$, and set $\mathfrak{g}_T \leftarrow \hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)$.

For $\kappa = 128$, the current recommendation is to use an optimal (asymmetric) Ate pairing over a subclass of Barreto-Naehrig curves. In that case, at security level of $\kappa = 128$, an element of $\mathbb{G}_1/\mathbb{G}_2/\mathbb{G}_T$ can be represented in respectively 256/512/3072 bits.

Zero Knowledge. A NIZK argument for a group-dependent language $\mathcal{L}$ consists of four algorithms, setup, gencrs, pro and ver. The setup algorithm setup takes as input 1^κ and n (the input length), and outputs the group description gk. The CRS generation algorithm gencrs takes as input gk and outputs the prover's CRS crs_p, the verifier's CRS crs_v, and a trapdoor td. The distinction between crs_p and crs_v is only important for efficiency. The prover pro takes as input gk and crs_p, a statement u, and a witness w, and outputs an argument π. The verifier ver takes as input gk and crs_v, a statement u, and an argument π, and either accepts or rejects.

Some of the properties of an argument are: (i) *perfect completeness* (honest verifier always accepts honest prover's argument), (ii) *perfect zero knowledge* (there exists an efficient simulator that can, given u, $(\mathsf{crs}_p, \mathsf{crs}_v)$ and td, output an argument that comes from the same distribution as the argument produced by the prover), (iii) *adaptive computational soundness* (if $u \notin \mathcal{L}$, then an arbitrary non-uniform probabilistic polynomial time prover has negligible probability of success in creating a satisfying argument), and (iv) *adaptive computational culpable soundness* [28, 29] (if $u \notin \mathcal{L}$, then an arbitrary NUPPT prover has negligible success in creating a satisfying argument together with a witness that $u \notin \mathcal{L}$). An argument is an *argument of knowledge*, if from an accepting argument it follows that the prover knows the witness. See App. A for formal definitions.

Generic Bilinear Group Model. We will prove the soundness of the new shuffle argument in the generic bilinear group model (GBGM, [38, 34, 8]). Our description of the GBGM is based on [34].

We start by picking a random asymmetric bilinear group $\mathsf{gk} := (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{genbp}(1^\kappa)$. Consider a black box $\mathbf{B}$ that can store values from groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ in internal state variables $\mathsf{cell}_1, \mathsf{cell}_2, \ldots$, where for simplicitly we allow the storage space to be infinite (this only increases the power of a generic adversary). The initial state consists of some values $(\mathsf{cell}_1, \mathsf{cell}_2, \ldots, \mathsf{cell}_{|inp|})$, which are set according to some probability distribution. Each state variable cell_i has an accompanying type $\mathsf{type}_i \in \{1, 2, T, \perp\}$. We assume initially $\mathsf{type}_i = \perp$ for $i > |inp|$. The black box allows computation operations on internal state variables and queries about the internal state. No other interaction with $\mathbf{B}$ is possible.

Let Π be the allowed set of computation operations. A computation operation consists of selecting a (say, t-ary) operation $f \in \Pi$ together with $t + 1$ indices $i_1, i_2, \ldots, i_{t+1}$. Assuming inputs have the correct type, $\mathbf{B}$ computes $f(\mathsf{cell}_{i_1}, \ldots, \mathsf{cell}_{i_t})$ and stores the result in $\mathsf{cell}_{i_{t+1}}$. For a set Σ of relations, a query consists of selecting a (say, t-ary) relation $\varrho \in \Sigma$ together with t indices

$i_1, i_2, \ldots, i_t$. Assuming inputs have the correct type, $\mathbf{B}$ replies to the query with $\varrho(\mathsf{cell}_{i_1}, \ldots, \mathsf{cell}_{i_t})$.

In the GBGM, we define $\Pi = \{\cdot, \hat{e}\}$ and $\Sigma = \{=\}$, where

1. On input $(\cdot, i_1, i_2, i_3)$: if $\mathsf{type}_{i_1} = \mathsf{type}_{i_2} \neq \perp$ then set $\mathsf{cell}_{i_3} \leftarrow \mathsf{cell}_{i_1} \cdot \mathsf{cell}_{i_2}$ and $\mathsf{type}_{i_3} \leftarrow \mathsf{type}_{i_1}$.
2. On input $(\hat{e}, i_1, i_2, i_3)$: if $\mathsf{type}_{i_1} = 1$ and $\mathsf{type}_{i_2} = 2$ then set $\mathsf{cell}_{i_3} \leftarrow \hat{e}(\mathsf{cell}_{i_1}, \mathsf{cell}_{i_2})$ and $\mathsf{type}_{i_3} \leftarrow T$.
3. On input $(=, i_1, i_2)$: if $\mathsf{type}_{i_1} = \mathsf{type}_{i_2} \neq \perp$ and $\mathsf{cell}_{i_1} = \mathsf{cell}_{i_2}$ then return 1. Otherwise return 0.

Since we are proving lower bounds, we will give a generic adversary adv additional power. We assume that all relation queries are for free. We also assume that adv is successful if after τ operation queries, he makes an equality query $(=, i_1, i_2)$, $i_1 \neq i_2$, that returns 1; at this point adv quits. Thus, if $\mathsf{type}_i \neq \perp$, then $\mathsf{cell}_i = F_1(\mathsf{cell}_1, \ldots, \mathsf{cell}_{|inp|})$ for a polynomial F_i known to adv.

The GBGM has proved itself to be very fruitful since its introduction, [8]. In particular, the generic (bilinear) group model is amenable to computerized analysis, and as such, has proven itself to be very useful say in the area of structure-preserving signature schemes [3]; see also [1].

Finally, Fischlin [21] and Dent [17] have pointed out that there exist constructions that are secure in (Shoup's version of) the generic group model but cannot be instantiated given any efficient instantiation of the group encoding. However, their constructions are utterly artificial; e.g., Dent constructed a signature scheme that under certain conditions outputs the signing key as a part of the signature.

Cryptosystems. A public-key cryptosystem Π is a triple $(\mathsf{genpkc}, \mathsf{enc}, \mathsf{dec})$ of efficient algorithms. The key generation algorithm $\mathsf{genpkc}(1^\kappa)$ returns a fresh public and secret key pair $(\mathsf{pk}, \mathsf{sk})$. The encryption algorithm $\mathsf{enc}_{\mathsf{pk}}(m; r)$, given a public key pk, a message m, and a randomizer r (from some randomizer space $\mathcal{R}$), returns a ciphertext. The decryption algorithm $\mathsf{dec}_{\mathsf{sk}}(c)$, given a secret key sk and a ciphertext c, returns a plaintext m. It is required that for each $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{genpkc}(1^\kappa)$ and each m, r, it holds that $\mathsf{dec}_{\mathsf{sk}}(\mathsf{enc}_{\mathsf{pk}}(m; r)) = m$. Informally, Π is *IND-CPA secure*, if the distributions of ciphertexts corresponding to any two plaintexts are computationally indistinguishable.

We will use the ILin cryptosystem from [18]; it is distinguished from other well-known cryptosystems like the BBS cryptosystem [9] by having shorter secret and public keys. Consider group $\mathbb{G}_k$, $k \in \{1, 2\}$. In this cryptosytem, where the secret key is $\mathsf{sk} = \gamma \leftarrow_r \mathbb{Z}_q \setminus \{0, -1\}$, the public key is $\mathsf{pk}_k \leftarrow (\mathfrak{g}_k, \mathfrak{h}_k) = (\mathfrak{g}_k, \mathfrak{g}_k^\gamma)$, and the encryption of a small $m \in \mathbb{Z}_q$ is

$$\mathsf{enc}_{\mathsf{pk}_k}(m; \boldsymbol{s}) := (\mathfrak{h}_k^{s_1}, (\mathfrak{g}_k \mathfrak{h}_k)^{s_2}, \mathfrak{g}_k^m \mathfrak{g}_k^{s_1+s_2})$$

for $\boldsymbol{s} \leftarrow_r \mathbb{Z}_q^{1\times 2}$. Denote $\mathfrak{P}_{k1} := (\mathfrak{h}_k, \mathfrak{1}_k, \mathfrak{g}_k)$ and $\mathfrak{P}_{k2} := (\mathfrak{1}_k, \mathfrak{g}_k \mathfrak{h}_k, \mathfrak{g}_k)$, thus $\mathsf{enc}_{\mathsf{pk}_k}(m; \boldsymbol{s}) = (\mathfrak{1}_k, \mathfrak{1}_k, \mathfrak{g}_k^m) \cdot \mathfrak{P}_{k1}^{s_1} \mathfrak{P}_{k2}^{s_2}$. Given $\mathfrak{v} \in \mathbb{G}_k^3$, the decryption sets

$$\mathsf{dec}_{\mathsf{sk}}(\mathfrak{v}) := \log_{\mathfrak{g}_k}(\mathfrak{v}_3 \mathfrak{v}_2^{-1/(\gamma+1)} \mathfrak{v}_1^{-1/\gamma}) \ ,$$

Decryption succeeds since $\mathfrak{v}_3\mathfrak{v}_2^{-1/(\gamma+1)}\mathfrak{v}_1^{-1/\gamma} = \mathfrak{g}_k^m\mathfrak{g}_k^{s_1+s_2} \cdot (\mathfrak{g}_k\mathfrak{h}_k)^{-s_2/(\gamma+1)} \cdot \mathfrak{h}_k^{-s_1/\gamma} = \mathfrak{g}_k^m\mathfrak{g}_k^{s_1+s_2} \cdot \mathfrak{g}_k^{-s_2/(\gamma+1)}\mathfrak{g}_k^{-s_2\cdot\gamma/(\gamma+1)} \cdot \mathfrak{g}_k^{-s_1} = \mathfrak{g}_k^m$. This cryptosystem is CPA-secure under the 2-Incremental Linear (2-ILin) assumption, see [18]. The ILin cryptosystem is *blindable*, $\mathsf{enc}_{\mathsf{pk}_k}(m;\boldsymbol{s}) \cdot \mathsf{enc}_{\mathsf{pk}_k}(0;\boldsymbol{s}') = \mathsf{enc}_{\mathsf{pk}}(m;\boldsymbol{s}+\boldsymbol{s}')$.

We use a variant of the ILin cryptosystem where each plaintext is encrypted twice, in group $\mathbb{G}_1$ and in $\mathbb{G}_2$ (but by using the same secret key an the same randomizer $\boldsymbol{s}$ in both). For technical reasons (relevant to the shuffle argument but not to the ILin cryptosystem), in group $\mathbb{G}_1$ we will use an auxiliary generator $\hat{\mathfrak{g}}_1 = \mathfrak{g}_1^{\varrho/\beta}$ instead of $\mathfrak{g}_1$, for $(\varrho,\beta) \leftarrow_r (\mathbb{Z}_q \setminus \{0\})^2$; both encryption and decryption are done as before but just using the secret key $\mathsf{sk} = (\varrho,\beta,\gamma)$ and the public key $\mathsf{pk}_1 = (\hat{\mathfrak{g}}_1, \mathfrak{h}_1 = \hat{\mathfrak{g}}_1^{\gamma})$; this also redefines $\mathfrak{P}_{k1}$. That is, $\mathsf{enc}_{\mathsf{pk}}(m;\boldsymbol{s}) = (\mathsf{enc}_{\mathsf{pk}_1}(m;\boldsymbol{s}), \mathsf{enc}_{\mathsf{pk}_2}(m;\boldsymbol{s}))$, where $\mathsf{pk}_1 = (\hat{\mathfrak{g}}_1, \mathfrak{h}_1 = \hat{\mathfrak{g}}_1^{\gamma})$, and $\mathsf{pk}_2 = (\mathfrak{g}_2, \mathfrak{h}_2 = \mathfrak{g}_2^{\gamma})$, and $\mathsf{dec}_{\mathsf{sk}}(\mathfrak{v}) := \log_{\hat{\mathfrak{g}}_1}(\mathfrak{v}_3\mathfrak{v}_2^{-1/(\gamma+1)}\mathfrak{v}_1^{-1/\gamma}) = \log_{\mathfrak{g}_1}(\mathfrak{v}_3\mathfrak{v}_2^{-1/(\gamma+1)}\mathfrak{v}_1^{-1/\gamma})/(\varrho/\beta)$ for $\mathfrak{v} \in \mathbb{G}_1^3$. We call this the *validity-enhanced* ILin cryptosystem.

In this case we denote the ciphertext in group k by $\mathfrak{v}_k$, and its jth component by $\mathfrak{v}_{kj}$. In the case when we have many ciphertexts, we denote the ith ciphertext by $\mathfrak{v}_i$ and the jth component of the ith ciphertext in group k by $\mathfrak{v}_{ikj}$.

3 Shuffle Argument

In the current section, we will give a full description of the new shuffle argument, followed by its efficiency analysis. Intuition behind its soundness will be given in Sect. 4. The full soundness proof is long, and postponed to Sect. 5, 6, and 7. Its zero knowledge property will be proven in Sect. 8.

Let $\Pi = (\mathsf{genpkc}, \mathsf{enc}, \mathsf{dec})$ be an additively homomorphic cryptosystem with randomizer space R; we assume henceworth that one uses the validity-enhanced ILin cryptosystem. Assume that $\mathfrak{v}_i$ and $\mathfrak{v}'_i$ are valid ciphertexts of Π. In a shuffle argument, the prover aims to convince the verifier in zero-knowledge that given $(\mathsf{pk}, (\mathfrak{v}_i, \mathfrak{v}'_i)_{i=1}^n)$, he knows a permutation $\sigma \in S_n$ and randomizers s_{ij}, $i \in [1\,..\,n]$ and $j \in [1\,..\,2]$, such that $\mathfrak{v}'_i = \mathfrak{v}_{\sigma(i)} \cdot \mathsf{enc}_{\mathsf{pk}}(0;\boldsymbol{s}_i)$ for $i \in [1\,..\,n]$. More precisely, we define the group-specific binary relation $\mathcal{R}_{sh,n}$ exactly as in [28, 33]:

$$\mathcal{R}_{sh,n} := \left(\begin{array}{l} (\mathsf{gk}, (\mathsf{pk}, \mathfrak{v}_i, \mathfrak{v}'_i)_{i=1}^n), (\sigma, \boldsymbol{s})) : \\ \sigma \in S_n \wedge \boldsymbol{s} \in R^{n\times 2} \wedge \left(\forall i : \mathfrak{v}'_i = \mathfrak{v}_{\sigma(i)} \cdot \mathsf{enc}_{\mathsf{pk}}(0;\boldsymbol{s}_i)\right) \end{array} \right) .$$

See Prot. 1 for the full description of the new shuffle argument.

We note that in the real mix-net, (γ, ϱ, β) is handled differently (in particular, γ — and possibly ϱ/β — will be known to the decrypting party while (ϱ,β) does not have to be known to anybody) than the real trapdoor (χ,α) that enables one to simulate the argument and thus cannot be known to anybody. Moreover, $(\mathfrak{g}_1,\mathfrak{g}_2)^{\sum P_i(\chi)}$ is in the CRS only to optimize computation. A precise efficiency analysis of this argument is given in Sect. 9.

In the rest of this section, we will explain the notion of batching and define non-batched versions (that are easier to read and analyse in the soundness proof) of the verification equations. We then state the main security theorem.

$\mathsf{gencrs}(1^\kappa, n \in \mathrm{poly}(\kappa))$**:** Call $\mathsf{gk} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{genbp}(1^\kappa)$. Let $P_i(X)$ for $i \in [0\,..\,n]$ be polynomials, chosen in Sect. 5. Set $\boldsymbol{\chi} = (\chi, \alpha, \varrho, \beta, \gamma) \leftarrow_r \mathbb{Z}_q^2 \times (\mathbb{Z}_q \setminus \{0\})^2 \times (\mathbb{Z}_q \setminus \{0, -1\})$. Let enc be the ILin cryptosystem with the secret key γ, and let $(\mathsf{pk}_1, \mathsf{pk}_2)$ be its public key. Set

$$\mathsf{crs} \leftarrow \begin{pmatrix} \mathsf{gk}, (\mathfrak{g}_1^{P_i(\chi)})_{i=1}^n, \mathfrak{g}_1^\varrho, \mathfrak{g}_1^{\alpha+P_0(\chi)}, \mathfrak{g}_1^{P_0(\chi)}, (\mathfrak{g}_1^{((P_i(\chi)+P_0(\chi))^2-1)/\varrho})_{i=1}^n, \\ \mathsf{pk}_1 = (\hat{\mathfrak{g}}_1 = \mathfrak{g}_1^{\varrho/\beta}, \mathfrak{h}_1 = \hat{\mathfrak{g}}_1^\gamma), \\ (\mathfrak{g}_2^{P_i(\chi)})_{i=1}^n, \mathfrak{g}_2^\varrho, \mathfrak{g}_2^{-\alpha+P_0(\chi)}, \mathsf{pk}_2 = (\mathfrak{g}_2, \mathfrak{h}_2 = \mathfrak{g}_2^\gamma), \mathfrak{g}_2^\beta, \\ \hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^{1-\alpha^2}, (\mathfrak{g}_1, \mathfrak{g}_2)^{\sum_{i=1}^n P_i(\chi)} \end{pmatrix}.$$

and $\mathsf{td} \leftarrow (\chi, \varrho)$. Return $(\mathsf{crs}, \mathsf{td})$.

$\mathsf{pro}(\mathsf{crs}; \mathfrak{v} \in (\mathbb{G}_1 \times \mathbb{G}_2)^{3n}; \sigma \in S_n, \boldsymbol{s} \in \mathbb{Z}_q^{n\times 2})$**:**

1. For $i = 1$ to $n-1$:
   (a) Set $r_i \leftarrow_r \mathbb{Z}_q$. Set $(\mathfrak{A}_{i1}, \mathfrak{A}_{i2}) \leftarrow (\mathfrak{g}_1, \mathfrak{g}_2)^{P_{\sigma^{-1}(i)}(\chi)+r_i\varrho}$.
2. Set $r_n \leftarrow -\sum_{i=1}^{n-1} r_i$.
3. Set $(\mathfrak{A}_{n1}, \mathfrak{A}_{n2}) \leftarrow (\mathfrak{g}_1, \mathfrak{g}_2)^{\sum_{i=1}^n P_i(\chi)} / \prod_{i=1}^{n-1} (\mathfrak{A}_{i1}, \mathfrak{A}_{i2})$.
4. For $i = 1$ to n: `/* Sparsity, for permutation matrix: */`
   (a) Set $\pi_{\mathsf{1sp}:i} \leftarrow (\mathfrak{A}_{i1}\mathfrak{g}_1^{P_0(\chi)})^{2r_i} (\mathfrak{g}_1^\varrho)^{-r_i^2} \mathfrak{g}_1^{((P_{\sigma^{-1}(i)}(\chi)+P_0(\chi))^2-1)/\varrho}$.
5. For $i = 1$ to n: `/* Shuffling itself */`
   (a) Set $(\mathfrak{v}'_{i1}, \mathfrak{v}'_{i2}) \leftarrow (\mathfrak{v}_{\sigma(i)1}, \mathfrak{v}_{\sigma(i)2}) \cdot (\mathsf{enc}_{\mathsf{pk}_1}(0; \boldsymbol{s}_i), \mathsf{enc}_{\mathsf{pk}_2}(0; \boldsymbol{s}_i))$.
6. Set `/* Consistency */`
   (a) For $k = 1$ to 2: Set $r_{s:k} \leftarrow_r \mathbb{Z}_q$. Set $\pi_{\mathsf{c1}:k} \leftarrow \mathfrak{g}_2^{\sum_{i=1}^n s_{ik}P_i(\chi)+r_{s:k}\varrho}$.
   (b) $(\boldsymbol{\pi}_{\mathsf{c2}:1}, \boldsymbol{\pi}_{\mathsf{c2}:2}) \leftarrow \prod_{i=1}^n (\mathfrak{v}_{i1}, \mathfrak{v}_{i2})^{r_i} \cdot (\mathsf{enc}_{\mathsf{pk}_1}(0; \boldsymbol{r}_s), \mathsf{enc}_{\mathsf{pk}_2}(0; \boldsymbol{r}_s))$.
7. Return $\pi_{sh} \leftarrow (\mathfrak{v}', (\mathfrak{A}_{i1}, \mathfrak{A}_{i2})_{i=1}^{n-1}, (\pi_{\mathsf{1sp}:i})_{i=1}^n, \pi_{\mathsf{c1}:1}, \pi_{\mathsf{c1}:2}, \boldsymbol{\pi}_{\mathsf{c2}:1}, \boldsymbol{\pi}_{\mathsf{c2}:2})$.

$\mathsf{ver}(\mathsf{crs}; \mathfrak{v}; \mathfrak{v}', (\mathfrak{A}_{i1}, \mathfrak{A}_{i2})_{i=1}^{n-1}, (\pi_{\mathsf{1sp}:i})_{i=1}^n, \pi_{\mathsf{c1}:1}, \pi_{\mathsf{c1}:2}, \boldsymbol{\pi}_{\mathsf{c2}:1}, \boldsymbol{\pi}_{\mathsf{c2}:2})$**:**

1. Set $(\mathfrak{A}_{n1}, \mathfrak{A}_{n2}) \leftarrow (\mathfrak{g}_1, \mathfrak{g}_2)^{\sum_{i=1}^n P_i(\chi)} / \prod_{i=1}^{n-1} (\mathfrak{A}_{i1}, \mathfrak{A}_{i2})$.
2. Set $(p_{1i}, p_{2j}, p_{3ij}, p_{4j})_{i\in[1\,..\,n], j\in[1\,..\,3]} \leftarrow_r \mathbb{Z}_q^{4n+6}$.
3. Check that `/* Permutation matrix: */`
$$\prod_{i=1}^n \hat{e}\left((\mathfrak{A}_{i1}\mathfrak{g}_1^{\alpha+P_0(\chi)})^{p_{1i}}, \mathfrak{A}_{i2}\mathfrak{g}_2^{-\alpha+P_0(\chi)}\right) = \hat{e}\left(\prod_{i=1}^n \pi_{\mathsf{1sp}:i}^{p_{1i}}, \mathfrak{g}_2^\varrho\right) \cdot \hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^{(1-\alpha^2)\sum_{i=1}^n p_{1i}}.$$
4. Check that `/* Validity: */`
$$\hat{e}\left(\mathfrak{g}_1^\varrho, \prod_{j=1}^3 \pi_{\mathsf{c2}:2j}^{p_{2j}} \cdot \prod_{i=1}^n \prod_{j=1}^3 (\mathfrak{v}'_{i2j})^{p_{3ij}}\right) = \hat{e}\left(\prod_{j=1}^3 \pi_{\mathsf{c2}:1j}^{p_{2j}} \cdot \prod_{i=1}^n \prod_{j=1}^3 (\mathfrak{v}'_{i1j})^{p_{3ij}}, \mathfrak{g}_2^\beta\right).$$
5. Set $\mathfrak{R} \leftarrow \hat{e}\left(\hat{\mathfrak{g}}_1, \pi_{\mathsf{c1}:2}^{p_{42}}(\pi_{\mathsf{c1}:1}\pi_{\mathsf{c1}:2})^{p_{43}}\right) \cdot \hat{e}\left(\mathfrak{h}_1, \pi_{\mathsf{c1}:1}^{p_{41}}\pi_{\mathsf{c1}:2}^{p_{42}}\right) / \hat{e}\left(\prod_{j=1}^3 \pi_{\mathsf{c2}:1j}^{p_{4j}}, \mathfrak{g}_2^\varrho\right)$.
6. Check that `/* Consistency: */`
$$\prod_{i=1}^n \hat{e}\left(\prod_{j=1}^3 (\mathfrak{v}'_{i1j})^{p_{4j}}, \mathfrak{g}_2^{P_i(\chi)}\right) / \prod_{i=1}^n \hat{e}\left(\prod_{j=1}^3 \mathfrak{v}_{i1j}^{p_{4j}}, \mathfrak{A}_{i2}\right) = \mathfrak{R}.$$

Protocol 1: The new shuffle argument

3.1 Batching

We assume that verifier checks that the batched version [4] of the equations (given in Prot. 1) hold. However, for soundness we need that the individual (non-batched) verification equations hold. We will show that we still have soundness even if the verifier checks batched versions of the equations.

We first prove the following lemma. We state it in the case where $f_i(\boldsymbol{X})$ are polynomials, but one can obviously transform it to the case where $f_i(\boldsymbol{X})$ are Laurent polynomials or even rational functions.

Lemma 1. *Assume $(p_i)_{i\in[1\,..\,k]}$ are values chosen uniformly random from $\mathbb{Z}_q^k$. Assume $\boldsymbol{\chi}$ are values chosen uniformly at random from $\mathbb{Z}_q$. Assume f_i are some polynomials of degree* $\mathrm{poly}(\kappa)$*. If the equation $\prod_{i=1}^{k} \hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^{f_i(\boldsymbol{\chi})p_i} = \mathfrak{1}_T$ holds, then with probability $\geq 1 - 1/q$ the k pairing equations $\hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^{f_i(\boldsymbol{\chi})} = \mathfrak{1}_T$, $i \in [1\,..\,k]$ also hold.*

Proof. As the pairing is non-degenerate, $\prod_{i=1}^{k} \hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^{f_i(\boldsymbol{\chi})p_i} = \mathfrak{1}_T$ iff $\sum_{i=1}^{k} f_i(\boldsymbol{\chi})p_i = 0$. By the Schwartz-Zippel lemma [37, 39], with probability $\geq 1 - 1/q$ this means $\sum_{i=1}^{k} f_i(\boldsymbol{\chi})Y_i = 0$ as a polynomial, where $(Y_i)_{i\in[1\,..\,k]}$ are random variables corresponding to p_i. Hence all individual coefficients of Y_i must be zero, i.e., $f_i(\boldsymbol{\chi}) = 0$ for $i \in [1\,..\,k]$. But then we have for $i \in [1\,..\,k]$ that $\hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^{f_i(\boldsymbol{\chi})} = \hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^0 = \mathfrak{1}_T$, as desired. □

The following corollary follows immediately from Lem. 1.

Corollary 1. *Assume $\boldsymbol{\chi} = (\chi, \alpha, \varrho, \beta, \gamma)$ is chosen uniformly random from $\mathbb{Z}_q^2 \times (\mathbb{Z}_q \setminus \{0\})^2 \times (\mathbb{Z}_q \setminus \{0, -1\})$. Assume $(p_{1i}, p_{2j}, p_{3ij}, p_{4j})_{i\in[1\,..\,n], j\in[1\,..\,3]}$ are values chosen uniformly random from $\mathbb{Z}_q^{4n+6}$. Consider the verification steps in Prot. 1.*

- *If the verification on Step 3 accepts, then (with probability $\geq 1 - 1/q$) for $i \in [1\,..\,n]$,*

$$\hat{e}\left(\mathfrak{A}_{i1}\mathfrak{g}_1^{\alpha+P_0(\chi)}, \mathfrak{A}_{i2}\mathfrak{g}_2^{-\alpha+P_0(\chi)}\right) = \hat{e}\left(\pi_{\mathsf{1sp}:i}, \mathfrak{g}_2^{\varrho}\right)\hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^{1-\alpha^2} \quad . \tag{1}$$

- *If the verification on Step 4 accepts, then with probability $\geq 1 - 1/q$,*

$$\hat{e}(\mathfrak{g}_1^{\varrho}, \pi_{\mathsf{c2}:2i}) = \hat{e}(\pi_{\mathsf{c2}:1i}, \mathfrak{g}_2^{\beta}) \quad , \quad i \in [1\,..\,3] \quad , \tag{2}$$

$$\hat{e}\left(\mathfrak{g}_1^{\varrho}, \mathfrak{v}'_{i2j}\right) = \hat{e}(\mathfrak{v}'_{i1j}, \mathfrak{g}_2^{\beta}) \quad , \quad i \in [1\,..\,n], j \in [1\,..\,3] \quad . \tag{3}$$

- *If the verification on Step 6 accepts, then with probability $\geq 1 - 1/q$,*

$$\prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}'_{i1}, \mathfrak{g}_2^{P_i(\chi)}\right) / \prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}_{i1}, \mathfrak{A}_{i2}\right) = \hat{e}\left(\mathfrak{P}_{11}, \pi_{\mathsf{c1}:1}\right)\hat{e}\left(\mathfrak{P}_{12}, \pi_{\mathsf{c1}:2}\right)/\hat{e}\left(\boldsymbol{\pi}_{\mathsf{c2}:1}, \mathfrak{g}_2^{\varrho}\right) \quad . \tag{4}$$

Proof. If the verification on Step 3 accepts, then we get that

$$\prod_{i=1}^{n} \left(\hat{e}\left(\mathfrak{A}_{i1}\mathfrak{g}_1^{\alpha+P_0(\chi)}, \mathfrak{A}_{i2}\mathfrak{g}_2^{-\alpha+P_0(\chi)}\right) / \left(\hat{e}(\pi_{\mathsf{1sp}:i}, \mathfrak{g}_2^{\varrho})\hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^{1-\alpha^2}\right)\right)^{p_{1i}}$$

$$= \prod_{i=1}^{n} \hat{e}\left((\mathfrak{A}_{i1}\mathfrak{g}_1^{\alpha+P_0(\chi)})^{p_{1i}}, \mathfrak{A}_{i2}\mathfrak{g}_2^{-\alpha+P_0(\chi)}\right) / \prod_{i=1}^{n}\left(\hat{e}(\pi_{\mathsf{1sp}:i}, \mathfrak{g}_2^{\varrho})\hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^{1-\alpha^2}\right)^{p_{1i}}$$
$$= \mathfrak{1}_T.$$

By Lem. 1, with probability $\geq 1 - 1/q$ we get

$$\hat{e}\left(\mathfrak{A}_{i1}\mathfrak{g}_1^{\alpha+P_0(\chi)}, \mathfrak{A}_{i2}\mathfrak{g}_2^{-\alpha+P_0(\chi)}\right) / \left(\hat{e}(\pi_{\mathsf{1sp}:i}, \mathfrak{g}_2^{\varrho})\hat{e}(\mathfrak{g}_1, \mathfrak{g}_2)^{1-\alpha^2}\right) = \mathfrak{1}_T,$$

for $i \in [1\,..\,n]$. Simplifying, this is precisely Eq. (1). The other cases are similar. □

This means that with probability $\geq 1 - 3/q$, checking the batched version of verification equations (as in Prot. 1) is equivalent to the checking of individual verification equations (as in Cor. 1).

We note that Cor. 1 also holds when $\boldsymbol{\chi}$ is chosen according to the distribution, stipulated in Prot. 1.

3.2 Statement of Security

Theorem 1 (Shuffle Security). *The shuffle argument from Prot. 1 is perfectly complete, computationally sound in the GBGM, and perfectly zero knowledge. More precisely, any generic adversary attacking the soundness of the new shuffle argument requires $\Omega(\sqrt{q/n})$ computation.*

Proof. Completeness: we deal with other verifications in later sections. Currently we only show that if the prover and the verifier are honest, then Eq. (4) (and thus also, the verification on step 6 in Prot. 1) accepts. Really, let $\mathfrak{v}'_{ik} = \mathfrak{v}_{\sigma(i)k} \cdot \mathsf{enc}_{\mathsf{pk}_k}(0; \boldsymbol{s}_i)$ and $\mathsf{pk}_1 = (\hat{\mathfrak{g}}_1, \mathfrak{h}_1)$ for some $\boldsymbol{s}_i \in \mathbb{Z}_q^{1\times 2}$. Then,

$$\begin{aligned}
\prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}'_{i1}, \mathfrak{g}_2^{P_i(\chi)}\right) &= \prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}_{\sigma(i)1} \cdot \mathsf{enc}_{\mathsf{pk}_1}(0; \boldsymbol{s}_i), \mathfrak{g}_2^{P_i(\chi)}\right) \\
&= \prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}_{\sigma(i)1}, \mathfrak{g}_2^{P_i(\chi)}\right) \cdot \prod_{i=1}^{n} \hat{e}\left(\mathsf{enc}_{\mathsf{pk}_1}(0; \boldsymbol{s}_i), \mathfrak{g}_2^{P_i(\chi)}\right) \\
&= \prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}_{i1}, \mathfrak{g}_2^{P_{\sigma^{-1}(i)}(\chi)}\right) \cdot \prod_{i=1}^{n} \hat{e}\left(\mathfrak{P}_{11}^{s_1}\mathfrak{P}_{12}^{s_2}, \mathfrak{g}_2^{P_i(\chi)}\right) \\
&= \prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}_{i1}, \mathfrak{g}_2^{P_{\sigma^{-1}(i)}(\chi)}\right) \cdot \hat{e}\left(\mathfrak{P}_{11}, \mathfrak{g}_2^{\sum_{i=1}^{n} s_{i1}P_i(\chi)}\right) \cdot \hat{e}\left(\mathfrak{P}_{12}, \mathfrak{g}_2^{\sum_{i=1}^{n} s_{i2}P_i(\chi)}\right)
\end{aligned}$$

and

$$\begin{aligned}
\prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}_{i1}, \mathfrak{A}_{i2}\right) &= \prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}_{i1}, \mathfrak{g}_2^{P_{\sigma^{-1}(i)}(\chi)+r_i\varrho}\right) \\
&= \prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}_{i1}, \mathfrak{g}_2^{P_{\sigma^{-1}(i)}(\chi)}\right) \cdot \hat{e}\left(\prod_{i=1}^{n} \mathfrak{v}_{i1}^{r_i}, \mathfrak{g}_2^{\varrho}\right) \quad .
\end{aligned}$$

Hence, as needed,

$$\begin{aligned}
&\prod_{i=1}^{n}\hat{e}\left(\mathfrak{v}'_{i1},\mathfrak{g}_2^{P_i(\chi)}\right)/\prod_{i=1}^{n}\hat{e}(\mathfrak{v}_{i1},\mathfrak{A}_{i2})\\
&=\hat{e}\left(\mathfrak{P}_{11},\mathfrak{g}_2^{\sum_{i=1}^{n}s_{i1}P_i(\chi)}\right)\cdot\hat{e}\left(\mathfrak{P}_{12},\mathfrak{g}_2^{\sum_{i=1}^{n}s_{i2}P_i(\chi)}\right)/\hat{e}\left(\prod_{i=1}^{n}\mathfrak{v}_{i1}^{r_i},\mathfrak{g}_2^{\varrho}\right)\\
&=\hat{e}\left(\mathfrak{P}_{11},\mathfrak{g}_2^{\sum_{i=1}^{n}s_{i1}P_i(\chi)+r_{s:1}\varrho}\right)\cdot\\
&\quad\hat{e}\left(\mathfrak{P}_{12},\mathfrak{g}_2^{\sum_{i=1}^{n}s_{i2}P_i(\chi)+r_{s:2}\varrho}\right)/\hat{e}\left(\mathfrak{P}_{11}^{r_{s:1}}\mathfrak{P}_{12}^{r_{s:2}}\cdot\prod_{i=1}^{n}\mathfrak{v}_{i1}^{r_i},\mathfrak{g}_2^{\varrho}\right)\\
&=\hat{e}\left(\mathfrak{P}_{11},\pi_{\mathsf{c1}:1}\right)\hat{e}\left(\mathfrak{P}_{12},\pi_{\mathsf{c1}:2}\right)/\hat{e}\left(\boldsymbol{\pi}_{\mathsf{c2}:1},\mathfrak{g}_2^{\varrho}\right)\quad.
\end{aligned}$$

Soundness. Intuition behind soundness will be given in Sect. 4. Soundness of this argument will be proven in Sect. 5, 6, and 7.

Zero-knowledge: The zero-knowledge property will be proven in Sect. 8. □

Since we work in the GBGM, where the adversary knows how all values were computed, Prot. 1 is actually an argument of knowledge.

4 Intuition Behind Soundness

Throughout this paper, we use a variation of the polynomial commitment scheme of type $\mathsf{com}_j(\boldsymbol{a};r):=\mathfrak{h}^{\sum_{i=1}^{n}a_iP_i(\chi)+r\varrho}$, where $\mathfrak{h}$ is a generator of $\mathbb{G}_j$, χ and ϱ are random values from $\mathbb{Z}_q$, and $P_i(X)$ are well-chosen polynomials. (The choice of $P_i(X)$ is fixed by the 1-sparsity argument, see Sect. 5.1.) Several variants of this commitment scheme are well-known to be perfectly hiding and computationally binding (under a suitable computational assumption, security of which is usually proved in the GBGM, [26, 30]). However, since we only rely on the security of this commitment scheme within the GBGM soundness proof of the shuffle, we will state neither the concrete assumption nor the security requirements (like hiding and binding) of a commitment scheme.

On the last three steps, see Prot. 1, the verifier executes four different verifications, restated in an easier to read format in Cor. 1. Each of these verifications has an intuitive meaning, resulting in a different subargument. However, since all of them have to use the same CRS and the soundness proof is in the GBGM, the subarguments interact strongly.

Our soundness proof in the GBGM uses the following idea. An adversary can only produce group elements from $\mathbb{G}_1$ or $\mathbb{G}_2$ that are products of the elements of the same group given in the CRS; elements of $\mathbb{G}_T$ can also be output by the pairing operation. Let $\boldsymbol{\chi}=(\chi,\alpha,\varrho,\beta,\gamma)$ be concrete (randomly chosen) values from $\mathbb{Z}_q$ and $\boldsymbol{X}=(X,X_\alpha,X_\varrho,X_\beta,X_\gamma)$ be the corresponding random variables. E.g., if $\mathcal{F}(\boldsymbol{X})=\{F_i(\boldsymbol{X})\}$ is the set of all rational functions such that $\mathfrak{g}_1^{\mathcal{F}(\boldsymbol{\chi})}=$

$\{\mathfrak{g}_1^{F_i(\boldsymbol{\chi})}\}$ is equal to the set of all CRS values in $\mathbb{G}_1$, then any value that the adversary creates in $\mathbb{G}_1$ must be of the form $\mathfrak{g}_1^{A(\boldsymbol{\chi})}$, where $A(\boldsymbol{X}) \in \operatorname{span} \mathcal{F}(\boldsymbol{X})$.

In this way, after taking a discrete logarithm, each verification equation can be written in the form $\mathcal{V}(\boldsymbol{\chi}) = 0$ for some polynomial $\mathcal{V}(\boldsymbol{X})$ known to the adversary. However, since the values in $\boldsymbol{\chi}$ were chosen uniformly random, from the Schwartz-Zippel lemma [37, 39] we can conclude that $\mathcal{V}(\boldsymbol{X}) = 0$ as a polynomial (or a rational function), except with negligible probability $O(n)/q$. From $\mathcal{V}(\boldsymbol{X}) = 0$, we deduce that all the coefficients of terms $X_\alpha^{i_1} X_\varrho^{i_2} X_\beta^{i_3} X_\gamma^{i_4}$ in $\mathcal{V}(\boldsymbol{X}) \cdot \mathcal{V}^*(\boldsymbol{X})$ (where $\mathcal{V}^*(\boldsymbol{X})$ is the denominator of $\mathcal{V}(\boldsymbol{X})$) are zero, giving us several equations related to the adversary's chosen values. From these equations and the linear independence of polynomials $P_i(X)$, we can deduce that the adversary's chosen values must be of a certain form, except with negligible probability $O(n)/q$.

More precisely, for symbolic values T and t, define (by following the definition of the CRS in Prot. 1)

$$\begin{aligned}
\mathsf{crs}_1(\boldsymbol{X}, T, t) =& t(X) + T_\varrho X_\varrho + T_\alpha \cdot (X_\alpha + P_0(X)) + T_0 P_0(X) + \frac{t^\dagger(X) Z(X)}{X_\varrho} + \\
& \frac{T_{\varrho\beta} X_\varrho}{X_\beta} + \frac{T_\gamma X_\varrho X_\gamma}{X_\beta} \ , \\
\mathsf{crs}_2(\boldsymbol{X}, T, t) =& t(X) + T_\varrho X_\varrho + T_\alpha \cdot (-X_\alpha + P_0(X)) + T_1 + T_\gamma X_\gamma + T_\beta X_\beta \ ,
\end{aligned}$$

where $t^\dagger(X)$ is in the span of $\{((P_i(X) + P_0(X)) - 1)^2 / Z(X)\}_{i=1}^n$ and $t(X)$ is in the span of $\{P_i(X)\}_{i=1}^n$. We will follow the same notation in the rest of the paper. In particular, all "daggered" polynomials (e.g., $b^\dagger(X)$) are in the span of $\{((P_i(X) + P_0(X)) - 1)^2 / Z(X)\}_{i=1}^n$. Since $\deg Z(X) = n + 1$, $\deg t^\dagger(X) \leq n - 1$, and $\deg t(X) \leq n$, then $\deg(\mathsf{crs}_1(\boldsymbol{X}, T, t) \cdot X_\varrho X_\beta) \leq (n-1) + (n+1) - 1 + 2 = 2n + 1$. (Multiplication with $X_\varrho X_\beta$ is needed to make $\mathsf{crs}_1(\boldsymbol{X}, T, t)$ a polynomial.) Analogously, $\deg \mathsf{crs}_2(\boldsymbol{X}, T, t) \leq n$. Importantly, $\{P_i(X)\}_{i=0}^n$ is linearly independent. In particular, $P_0(X)$ is linearly independent to all other polynomials present in $\mathsf{crs}_1(\boldsymbol{X})$ and $\mathsf{crs}_2(\boldsymbol{X})$, except the "daggered" polynomial $t^\dagger(X)$.

Since the shuffle argument adversary is a GBGM adversary (and one uses ILin encryption), she knows the following polynomials (in the case of crs_2-functions), Laurent polynomials (in the case of crs_1-functions) or rational functions (in the case of $M_{ij}(\boldsymbol{X})$, $M'_{ij}(\boldsymbol{X})$, and $M_{E:j}(\boldsymbol{X})$), where $\hat{\mathfrak{g}}_2 = \mathfrak{g}_2$:

$$\begin{aligned}
A(\boldsymbol{X}) =& \mathsf{crs}_1(\boldsymbol{X}, A, a) && \text{s.t. } \mathfrak{A}_1 = \mathfrak{g}_1^{A(\boldsymbol{\chi})} \ , \\
B(\boldsymbol{X}) =& \mathsf{crs}_2(\boldsymbol{X}, B, b) && \text{s.t. } \mathfrak{A}_2 = \mathfrak{g}_2^{B(\boldsymbol{\chi})} \ , \\
C(\boldsymbol{X}) =& \mathsf{crs}_1(\boldsymbol{X}, C, c) && \text{s.t. } \pi_{\mathsf{1sp}} = \mathfrak{g}_1^{C(\boldsymbol{\chi})} \ , \\
D_j(\boldsymbol{X}) =& \mathsf{crs}_2(\boldsymbol{X}, D_j, d_j) && \text{s.t. } \pi_{\mathsf{c1}:j} = \mathfrak{g}_2^{D_j(\boldsymbol{\chi})} \ , \\
E_{kj}(\boldsymbol{X}) =& \mathsf{crs}_k(\boldsymbol{X}, E_{kj}, e_{kj}) && \text{s.t. } \pi_{\mathsf{c2}:kj} = \mathfrak{g}_k^{E_{kj}(\boldsymbol{\chi})} \ , \\
V_{ikj}(\boldsymbol{X}) =& \mathsf{crs}_k(\boldsymbol{X}, V_{ikj}, v_{ikj}) && \text{s.t. } \mathfrak{v}_{ikj} = \hat{\mathfrak{g}}_k^{V_{ikj}(\boldsymbol{\chi})} \ ,
\end{aligned}$$

$$
\begin{aligned}
V'_{ikj}(\boldsymbol{X}) =& \mathsf{crs}_k(\boldsymbol{X}, V'_{ikj}, v'_{ikj}) && \text{s.t. } \mathfrak{v}'_{ikj} = \hat{\mathfrak{g}}_k^{V'_{ikj}(\boldsymbol{\chi})} \ , \\
M_{ij}(\boldsymbol{X}) =& V_{ij3}(\boldsymbol{X}) - V_{ij2}(\boldsymbol{X})/(X_\gamma + 1) - V_{ij1}/X_\gamma && \text{s.t. } \mathsf{dec}_{\mathsf{sk}}(\mathfrak{v}_{ij}) = M_{ij}(\boldsymbol{\chi}) \ , \\
M'_{ij}(\boldsymbol{X}) =& V'_{ij3}(\boldsymbol{X}) - V'_{ij2}(\boldsymbol{X})/(X_\gamma + 1) - V'_{ij1}/X_\gamma && \text{s.t. } \mathsf{dec}_{\mathsf{sk}}(\mathfrak{v}'_{ij}) = M'_{ij}(\boldsymbol{\chi}) \ , \\
M_{E:j}(\boldsymbol{X}) =& E_{j3}(\boldsymbol{X}) - E_{j2}(\boldsymbol{X})/(X_\gamma + 1) - E_{j1}/X_\gamma && \text{s.t. } \mathsf{dec}_{\mathsf{sk}}(\boldsymbol{\pi}_{\mathsf{c2}:j}) = M_{E:j}(\boldsymbol{\chi}) \ .
\end{aligned} \tag{5}
$$

We are now almost ready to explain the meaning of each individual verification equation. Before doing so, we emphasize that a major obstacle in proving soundness in the GBGM is that all subarguments must use the same CRS. In particular, a subargument that is sound by itself might stop being sound due to the elements in the CRS that are added because of other subarguments. Intuitively, we tackle this problem by introducing random variables α (that is only needed in Eq. (1)) and β (that is needed in Eq. (2) and Eq. (3)).

Briefly, the verifier makes three checks. Eq. (1), the "permutation matrix argument", guarantees that the prover has committed to a permutation matrix corresponding to some permutation σ. Eq. (3) and Eq. (2), the "validity argument", guarantee that the ciphertexts have not been formed in a devious way that would make the consistency argument to be unsound. Eq. (4), the "consistency argument", guarantees that the prover has used the same permutation σ to shuffle the ciphertexts.

Permutation Matrix Argument. Consider the subargument of Prot. 1, where the verifier just computes $(\mathfrak{A}_{n1}, \mathfrak{A}_{n2})$ and then performs the verification Eq. (1) for each $i = 1$ to n. We will call it the *permutation matrix argument*. In Sect. 5 we motivate this name, by showing that after the permutation matrix argument only, the verifier is convinced that $(\mathfrak{A}_{11}, \ldots, \mathfrak{A}_{n1})$ commits to a permutation matrix. For this, we first prove the security of its subargument — the 1-sparsity argument [33] — where the verifier performs the verification Eq. (1) for exactly one i.

To prove the security of permutation matrix argument, we have to solve a quite complicated system of polynomial equations. We do it by using a computer algebra system, see Sect. 5 for more details.

Validity Argument. As a subroutine in our argument, we make the verifier check the validity of all ciphertexts. This is done by checking Eq. (3) (and Eq. (2)). The main goal of the validity check is to show that the prover did not use "forbidden" terms $\mathfrak{g}_k^{P_i(\chi)}$ and $\mathfrak{g}_i^{\varrho}$ when computing the ciphertexts $\mathfrak{v}'_{ik}$ and $\boldsymbol{\pi}_{\mathsf{c2}:k}$. In the case of the Elgamal cryptosystem, the validity argument provides a proof that both $\mathfrak{v}'_{i1}$ and $\mathfrak{v}'_{i2}$ decrypt to a plaintext of form $M_i(\boldsymbol{X}) = \sum M_{ij} f_{ij}(\boldsymbol{X})$, for known coefficients M_{ij} and polynomials $f_{ij}(\boldsymbol{X})$, where none of the rational functions f_{ij} depends on either X or X_ϱ. (See Eq. (12).) Similar assurance is provided about the plaintext hidden in $\boldsymbol{\pi}_{\mathsf{c2}:k}$. Employing validity subarguments allows the consistency subargument to be more efficient than in [28, 19].

Consistency Argument. Finally, we show that performing all checks guarantees that $\mathsf{dec}_{\mathsf{sk}}(\mathfrak{v}'_i) = \mathsf{dec}_{\mathsf{sk}}(\mathfrak{v}_{\sigma(i)}) \neq \perp$ for some permutation $\sigma \in S_n$. The main observation is that a permutation of ciphertexts (without rerandomization) is invariant under multiplication: without rerandomizing the ciphertexts, the (non-batched) verification Eq. (4) would just be the identity $\hat{e}(\mathfrak{v}'_{i1}, \mathfrak{g}_2^{P_i(\chi)}) = \hat{e}(\mathfrak{v}_{i1}, \mathfrak{g}_2^{P_{\sigma^{-1}(i)}(\chi)})$, for all i. However, this trivially leaks the permutation σ, and hence is not secure. To ensure privacy, $\mathfrak{v}'_{i1}$ must be rerandomized, and $\mathfrak{g}_2^{P_{\sigma^{-1}(i)}(\chi)}$ must be replaced by a commitment to the unit vector $\boldsymbol{e}_{\sigma^{-1}(i)}$. This makes the final verification slightly more complicated, as we need extra terms to adjust it to the added random values.

A version of Eq. (4) was also used in [28, 33, 19]. However, the shuffle arguments from [28, 19] need to execute two versions of Eq. (4), once with $P_i(X)$ and once with different carefully chosen polynomials $\hat{P}_i(X)$ in $\mathbb{G}_2$. (See [28, 19] for an explanation.) In addition, one must prove that those two versions are consistent between each other (by providing a same-message argument, in the terminology of [19]). This makes the arguments of [28, 19] quite complicated.

Similarly to [33], we avoid this complication by having a validity argument on the ciphertexts. Since valid ciphertexts are not dependent of $P_i(X)$, it suffices for the verifier to execute just one version of Eq. (4).

5 Permutation Matrix Argument

In this section, we show that a subargument of Prot. 1, where the verifier only computes $\mathfrak{A}_{n1}$ as shown and then executes verification at Eq. (1) (for each $i \in [1\,..\,n]$) gives us a permutation matrix argument. This argument will be by far the most complex subargument that we use.

5.1 New 1-Sparsity Argument

In a 1-sparsity argument [33], the prover aims to convince the verifier that he knows how to open a commitment $\mathfrak{A}_1$ to $(\boldsymbol{a}, r)$, such that *at most* one coefficient a_I is non-zero. If, in addition, $a_I = 1$, then we have a unit vector argument [19]. A 1-sparsity argument can be constructed by using square span programs [16], an especially efficient variant of the quadratic span programs of [22]. We prove its security in the GBGM and therefore use a technique similar to that of [27], and this introduces some complications as we will demonstrate below. While we start using ideas behind the unit vector argument of [19], we only obtain a 1-sparsity argument. Then, in Sect. 5, we show how to obtain an efficient permutation matrix argument from it.

Clearly, $\boldsymbol{a} \in \mathbb{Z}_q^n$ is a unit vector iff the following $n+1$ conditions hold [19]:
- $a_i \in \{0, 1\}$ for $i \in [1\,..\,n]$ (i.e., $\boldsymbol{a}$ is Boolean), and
- $\sum_{i=1}^{n} a_i = 1$.

Let $\{0, 2\}^{n+1}$ denote the set of $(n+1)$-dimensional vectors where every coefficient is from $\{0, 2\}$, let $\circ$ denote the Hadamard (entry-wise) product of two vectors,

let $V := \begin{pmatrix} 2 \cdot I_{n \times n} \\ \mathbf{1}_n^\top \end{pmatrix} \in \mathbb{Z}_q^{(n+1) \times n}$ and $\boldsymbol{b} := \begin{pmatrix} \mathbf{0}_n \\ 1 \end{pmatrix} \in \mathbb{Z}_q^{n+1}$. Clearly, the above $n+1$ conditions hold iff $V\boldsymbol{a} + \boldsymbol{b} \in \{0,2\}^{n+1}$, i.e.,

$$(V\boldsymbol{a} + \boldsymbol{b} - \mathbf{1}_{n+1}) \circ (V\boldsymbol{a} + \boldsymbol{b} - \mathbf{1}_{n+1}) = \mathbf{1}_{n+1} \ . \tag{6}$$

Let ω_i, $i \in [1\,..\,n+1]$ be $n+1$ different values. Let

$$Z(X) := \prod_{i=1}^{n+1} (X - \omega_i)$$

be the unique degree $n+1$ monic polynomial, such that $Z(\omega_i) = 0$ for all $i \in [1\,..\,n+1]$. Let the ith Lagrange basis polynomial

$$\ell_i(X) := \prod_{j \in [1\,..\,n+1], j \neq i} ((X - \omega_j)/(\omega_i - \omega_j))$$

be the unique degree n polynomial, s.t. $\ell_i(\omega_i) = 1$ and $\ell_i(\omega_j) = 0$ for $j \neq i$.

For $i \in [1\,..\,n]$, let $P_i(X)$ be the polynomial that interpolates the ith column of the matrix V. That is,

$$P_i(X) = 2\ell_i(X) + \ell_{n+1}(X)$$

for $i \in [1\,..\,n]$. Let

$$P_0(X) = \ell_{n+1}(X) - 1$$

be the polynomial that interpolates $\boldsymbol{b} - \mathbf{1}_{n+1}$. In the rest of this paper, we will heavily use the following simple result.

Lemma 2. *$\{P_i(X)\}_{i=0}^n$ is linearly independent.*

Proof. Assume that $\sum_{i=0}^n b_i P_i(X) = 0$ for some constants b_i. Thus, $\sum_{i=0}^n b_i P_i(\omega_k) = 0$ for each k. Consider any $k \in [1\,..\,n]$. Then, $0 = b_0 P_0(\omega_k) + \sum_{i=1}^n b_i P_i(\omega_k) = b_0(\ell_{n+1}(\omega_k) - 1) + \sum_{i=1}^n b_i(2\ell_i(\omega_k) + \ell_{n+1}(\omega_k)) = -b_0 + 2b_k$. Thus, $b_k = b_0/2$ for $k \in [1\,..\,n]$. Consider now the case $k = n+1$, then $0 = b_0 P_0(\omega_{n+1}) + \sum_{i=1}^n b_i P_i(\omega_{n+1}) = b_0(\ell_{n+1}(\omega_{n+1}) - 1) + \sum_{i=1}^n b_i(2\ell_i(\omega_{n+1}) + \ell_{n+1}(\omega_{n+1})) = \sum_{i=1}^n b_i = n/2 \cdot b_0$. Thus $b_k = 0$ for $k \in [0\,..\,n]$. □

We arrive at the polynomial $Q(X) = (\sum_{i=1}^n a_i P_i(X) + P_0(X))^2 - 1 = (P_I(X) + P_0(X))^2 - 1$ (here, we used the fact that $\boldsymbol{a} = \boldsymbol{e_I}$ for some $I \in [1\,..\,n]$), such that $\boldsymbol{a}$ is a unit vector iff $Z(X) \mid Q(X)$. As in [27], to obtain privacy, we now add randomness $A_\varrho X_\varrho$ to $Q(X)$, arriving at the degree $2n$ polynomial

$$Q_{wi}(X, X_\varrho) = (P_I(X) + P_0(X) + A_\varrho X_\varrho)^2 - 1 \ . \tag{7}$$

Here, X_ϱ is a special independent random variable, and $A_\varrho \leftarrow_r \mathbb{Z}_q$. This means that we will use an instantiation of the polynomial commitment scheme (see Sect. 4) with $P_i(X)$ defined as in the current subsection.

The new 1-sparsity argument is the subargument of the shuffle argument on Prot. 1, where the verifier only executes verification step Eq. (1) for one concrete value of i.

Theorem 2. *Consider $i \in [1\,..\,n]$. The 1-sparsity argument is perfectly complete. The following holds in the GBGM, given that the generic adversary works in polynomial time. If the honest verifier accepts on Step 3 for this i, then there exists $I \in [1\,..\,n]$, such that*

$$\mathfrak{A}_{i1} = \mathfrak{g}_1^{a(\chi)+A_\varrho\varrho+A_\alpha(\alpha+P_0(\chi))} \quad , \tag{8}$$

where $a(X) = (1 + A_\alpha)P_I(X)$ for some constant A_α.

Proof. Completeness: For an honest prover, $\mathfrak{A}_{i1} = \mathfrak{g}_1^{A(\boldsymbol{\chi})}$, $\mathfrak{A}_{i2} = \mathfrak{g}_2^{B(\boldsymbol{\chi})}$, and $\pi_{\mathsf{1sp}:i} = \mathfrak{g}_1^{C(\boldsymbol{\chi})}$, where $A(\boldsymbol{X}) = B(\boldsymbol{X}) = P_I(X) + A_\varrho X_\varrho$ and $C(\boldsymbol{X}) = 2A_\varrho \cdot (A(\boldsymbol{X}) + P_0(X)) - A_\varrho^2 X_\varrho + Q_{wi}(X, X_\varrho)/X_\varrho$. Write

$$\mathcal{V}_{1sp}(\boldsymbol{X}) := (A(\boldsymbol{X})+X_\alpha+P_0(X))\cdot(B(\boldsymbol{X})-X_\alpha+P_0(X))-C(\boldsymbol{X})\cdot X_\varrho-(1-X_\alpha^2) \ . \tag{9}$$

The verification equation Eq. (1) assesses that $\mathcal{V}_{1sp}(\boldsymbol{\chi}) = 0$. This simplifies to $\mathcal{V}_{1sp}(\boldsymbol{X}) = (A_\varrho X_\varrho + P_I(X) + P_0(X))^2 - 1 - Q_{wi}(X, X_\varrho)$. Hence for an honest prover, it follows from Eq. (7) that $\mathcal{V}_{1sp}(\boldsymbol{\chi}) = 0$.

Soundness: Assume that the verifier has accepted inputs $\mathsf{cell}_{i_1} = A(\boldsymbol{X})$, $\mathsf{cell}_{i_2} = B(\boldsymbol{X})$, and $\mathsf{cell}_{i_3} = C(\boldsymbol{X})$, for some polynomials $A(\boldsymbol{X})$, $B(\boldsymbol{X})$, and $C(\boldsymbol{X})$. In the GBGM, the adversary knows all coefficients. (This corresponds to $\mathfrak{A}_{i1} = \mathfrak{g}_1^{A(\boldsymbol{\chi})}, \mathfrak{A}_{i2} = \mathfrak{g}_2^{B(\boldsymbol{\chi})}, \pi_{\mathsf{1sp}:i} = \mathfrak{g}_1^{C(\boldsymbol{\chi})}$.) Let $\mathcal{V}_{1sp}(\boldsymbol{X})$ then be as in Eq. (9) with $A(\boldsymbol{X})$, $B(\boldsymbol{X})$, and $C(\boldsymbol{X})$ as in Eq. (5). Let $\mathcal{V}^*_{1sp}(\boldsymbol{X})) := X_\varrho X_\beta$. Clearly, $\mathcal{V}_{1sp}(\boldsymbol{X}) \cdot \mathcal{V}^*_{1sp}(\boldsymbol{X})$ is a polynomial, with $\deg(\mathcal{V}_{1sp}(\boldsymbol{X}) \cdot \mathcal{V}^*_{1sp}(\boldsymbol{X})) \leq 3n+1$. Since the verifier accepts, $\mathcal{V}_{1sp}(\boldsymbol{X}) = 0$ as a polynomial.

In Tbl. 2, we enlist all the coefficients of $\mu(\boldsymbol{i}) = X_\alpha^{i_1} X_\varrho^{i_2} X_\beta^{i_3} X_\gamma^{i_4}$ in $\mathcal{V}_{1sp}(\boldsymbol{X}) \cdot \mathcal{V}^*_{1sp}(\boldsymbol{X})$. We remark that we found those polynomials by using a computer algebra system[1], but they can be verified manually.

Consider now each monomial of $\mathrm{coeff}_{\mu(\boldsymbol{i})}(\mathcal{V}_{1sp}(\boldsymbol{X}) \cdot \mathcal{V}^*_{1sp}(\boldsymbol{X})) = 0$ as a polynomial $F_i^*(\boldsymbol{Y})$ of formal variables $\boldsymbol{Y} := (a(X), A_\varrho, A_\alpha, \ldots, C_\gamma)$ (i.e., in all coefficients of $A(\boldsymbol{X})$, $B(\boldsymbol{X})$, and $C(\boldsymbol{X})$). We can now set $F_i^*(\boldsymbol{Y}) = 0$ for each monomial, and the solution set of this system of polynomial equations gives us all possible ways of "cheating" the adversary can do. However, the resulting polynomial equation system is too complicated, and moreover, it contains some formal variables that are *not* linearly independent, like $a(X)$ and $a^\dagger(X)$.

We hence execute two additional steps. First, we take into account (by using Lem. 2) that $P_0(X)$ is linearly independent of all other polynomials except "daggered" polynomials $a^\dagger(X)$ and $c^\dagger(X)$. This allows us to simplify some of the coefficients and gives some more polynomial equations. After that step, we get a new polynomial equation system $\{F_i(\boldsymbol{Y}) = 0\}$ for some polynomials F_i.

Second, we use a computer algebra system to derive a Gröbner basis [11] in variables in $\boldsymbol{Y}$ for the system $\{F_i(\boldsymbol{Y}) = 0\}$. By using lexicographic order (more precisely, we used the function `GroebnerBasis` of Mathematica, with parameters

[1] In the concrete case, Mathematica 9.0, but any other reasonably powerful system can be used. See [1] for references on the prior use of computer algebra systems to prove security in the generic (bilinear) group model

Table 2. $\mathrm{coeff}_{\mu(\boldsymbol{i})}(\mathcal{V}_{1sp}(\boldsymbol{X}) \cdot \mathcal{V}^*_{1sp}(\boldsymbol{X}))$, where $\mu(\boldsymbol{i}) = X_\alpha^{i_1} X_\varrho^{i_2} X_\beta^{i_3} X_\gamma^{i_4}$

$\{i_1, \ldots, i_4\}$	$\mathrm{coeff}_{\mu(\boldsymbol{i})}(\mathcal{V}_{1sp}(\boldsymbol{X}) \cdot \mathcal{V}^*_{1sp}(\boldsymbol{X}))$
$\{1,2,1,0\}$	$-A_\varrho(B_\alpha+1) + (A_\alpha+1)B_\varrho - C_\alpha$
$\{1,2,0,1\}$	$-A_\gamma(B_\alpha+1)$
$\{1,2,0,0\}$	$-A_{\varrho\beta}(B_\alpha+1)$
$\{1,1,2,0\}$	$(A_\alpha+1)B_\beta$
$\{1,1,1,1\}$	$(A_\alpha+1)B_\gamma$
$\{1,1,1,0\}$	$-a(X)(B_\alpha+1) + (A_\alpha+1)\left(b(X)+B_1\right) - A_0(B_\alpha+1)P_0(X)$
$\{1,0,1,0\}$	$-(B_\alpha+1)Z(X)a^\dagger(X)$
$\{0,3,1,0\}$	$A_\varrho B_\varrho - C_\varrho$
$\{0,3,0,1\}$	$A_\gamma B_\varrho - C_\gamma$
$\{0,3,0,0\}$	$A_{\varrho\beta}B_\varrho - C_{\varrho\beta}$
$\{0,2,2,0\}$	$A_\varrho B_\beta$
$\{0,2,1,1\}$	$A_\gamma B_\beta + A_\varrho B_\gamma$
$\{0,2,1,0\}$	$a(X)B_\varrho + A_\varrho\left(b(X)+B_1\right) + A_{\varrho\beta}B_\beta +$ $P_0(X)\left(A_\varrho(B_\alpha+1) + (A_\alpha+A_0+1)\,B_\varrho - C_\alpha - C_0\right) - c(X)$
$\{0,2,0,2\}$	$A_\gamma B_\gamma$
$\{0,2,0,1\}$	$A_\gamma\left(b(X)+B_1\right) + A_{\varrho\beta}B_\gamma + A_\gamma(B_\alpha+1)P_0(X)$
$\{0,2,0,0\}$	$A_{\varrho\beta}\left(b(X)+B_1\right) + A_{\varrho\beta}(B_\alpha+1)P_0(X)$
$\{0,1,2,0\}$	$a(X)B_\beta + (A_\alpha+A_0+1)\,B_\beta P_0(X)$
$\{0,1,1,1\}$	$a(X)B_\gamma + (A_\alpha+A_0+1)\,B_\gamma P_0(X)$
$\{0,1,1,0\}$	$-Z(X)c^\dagger(X) + P_0(X)\left(a(X)(B_\alpha+1) + (A_\alpha+A_0+1)\left(b(X)+B_1\right)\right) +$ $a(X)\left(b(X)+B_1\right) + (A_\alpha+A_0+1)\left(B_\alpha+1\right)P_0(X)^2 - 1 +$ $B_\varrho Z(X)a^\dagger(X)$
$\{0,0,2,0\}$	$B_\beta Z(X)a^\dagger(X)$
$\{0,0,1,1\}$	$B_\gamma Z(X)a^\dagger(X)$
$\{0,0,1,0\}$	$Z(X)\left(b(X)+B_1\right)a^\dagger(X) + (B_\alpha+1)P_0(X)Z(X)a^\dagger(X)$

`MonomialOrder -> Lexicographic` and `Method -> "Buchberger"`), we get the Gröbner basis $\{\mathcal{B}_i(\boldsymbol{Y})\}$ on Fig. 1.

The system of polynomial equations $\{\mathcal{B}_i(\boldsymbol{Y}) = 0\}$ can be solved manually. First, we simplify this system by setting $C_\gamma = 0$, $C_{\varrho\beta} = 0$, $B_\beta = 0$, $B_\gamma = 0$, $A_\gamma = 0$, $A_{\varrho\beta} = 0$, $a^\dagger(X) = 0$, $A_0 = 0$, $B_\alpha = -A_\alpha/(A_\alpha+1)$, $C_0 = 2A_\varrho/(A_\alpha+1)$, $b(X) = a(X)/(A_\alpha+1)^2 - B_1$, $C_\varrho = (A_\alpha+1)B_\varrho\left((A_\alpha+1)B_\varrho - C_\alpha\right)$. Then, we get a new system of polynomial equations, with the Gröbner basis $\{\mathcal{B}'_i(\boldsymbol{Y}) = 0\}$ as given on Fig. 2.

We can further simplify this system by noting that $C_\alpha = (A_\alpha+1)B_\varrho - A_\varrho/(A_\alpha+1)$ and thus $c(X) = a(X)\left(A_\varrho/(A_\alpha+1)^2 + B_\varrho\right)$. By inserting those two values to the Gröbner basis $\{\mathcal{B}'_i(\boldsymbol{Y})\}$, we get that the resulting system of polynomial equations has the following simple Gröbner basis $\{\mathcal{B}''_i(\boldsymbol{Y})\}$:

$$\left((A_\alpha+1)^2\left(-Z(X)c^\dagger(X) + P_0(X)^2 - 1\right) + 2a(X)(A_\alpha+1)P_0(X) + a(X)^2\right)$$

By solving $\mathcal{B}''_i(\boldsymbol{Y}) = 0$, we get

$$c^\dagger(X) = \frac{\left(\frac{a(X)}{A_\alpha+1} + P_0(X)\right)^2 - 1}{Z(X)} \ ,$$

$$\begin{pmatrix}
C_\gamma \\
C_{\varrho\beta} \\
4C_\varrho - C_0\left(C_0 + 2C_\alpha\right) \\
c(X)^2 + 2\left(C_0 + C_\alpha\right)P_0(X)c(X) + \left(C_0 + C_\alpha\right)^2\left(P_0(X)^2 - Z(X)c^\dagger(X) - 1\right) \\
B_\beta \\
B_\gamma \\
2B_\varrho - \left(B_\alpha + 1\right)\left(C_0 + 2C_\alpha\right) \\
\left(b(X) + B_1\right)\left(C_0 + C_\alpha\right) - c(X)\left(B_\alpha + 1\right) \\
b(X)c(X) + \left(B_1 + 2\left(B_\alpha + 1\right)P_0(X)\right)c(X) + \\
\qquad \left(B_\alpha + 1\right)\left(C_0 + C_\alpha\right)\left(P_0(X)^2 - Z(X)c^\dagger(X) - 1\right) \\
-\left(B_\alpha + 1\right)^2 - B_\alpha\left(B_\alpha + 2\right)Z(X)c^\dagger(X) - Z(X)c^\dagger(X) + \\
\qquad \left(b(X) + B_1 + \left(B_\alpha + 1\right)P_0(X)\right)^2 \\
A_\gamma \\
A_{\varrho\beta} \\
Z(X)a^\dagger(X) \\
A_0 \\
B_\alpha + A_\alpha\left(B_\alpha + 1\right) \\
2A_\varrho - (A_\alpha + 1)C_0 \\
a(X) - \left(A_\alpha + 1\right)^2\left(b(X) + B_1\right)
\end{pmatrix}$$

Fig. 1. Gröbner basis $\{\mathcal{B}_i(\boldsymbol{Y})\}$

$$\begin{pmatrix}
-\left(C_\alpha - 2(A_\alpha + 1)B_\varrho\right)^2\left(-Z(X)c^\dagger(X) + P_0(X)^2 - 1\right) + \\
\qquad 2c(X)P_0(X)\left(C_\alpha - 2(A_\alpha + 1)B_\varrho\right) - c(X)^2 \\
\frac{\left(C_\alpha - 2(A_\alpha+1)B_\varrho\right)^2\left(-Z(X)c^\dagger(X) + P_0(X)^2 - 1\right) + 2c(X)P_0(X)\left(2(A_\alpha+1)B_\varrho - C_\alpha\right) + c(X)^2}{A_\alpha + 1} \\
\left(A_\alpha + 1\right)\left(C_\alpha - (A_\alpha + 1)B_\varrho\right) + A_\varrho \\
\left(A_\alpha + 1\right)\Bigg(2(A_\alpha + 1)B_\varrho\left(-Z(X)c^\dagger(X) + P_0(X)^2 - 1\right) + \\
\qquad C_\alpha\left(Z(X)c^\dagger(X) - P_0(X)^2 + 1\right) + 2c(X)P_0(X)\Bigg) + a(X)c(X) \\
-(A_\alpha + 1)\left(c(X) - 2a(X)B_\varrho\right) - a(X)C_\alpha \\
a(X)\left(\frac{C_\alpha}{A_\alpha + 1} - 2B_\varrho\right) + c(X) \\
\left(A_\alpha + 1\right)^2\left(-Z(X)c^\dagger(X) + P_0(X)^2 - 1\right) + 2a(X)(A_\alpha + 1)P_0(X) + a(X)^2
\end{pmatrix} .$$

Fig. 2. Gröbner basis $\{\mathcal{B}'_i(\boldsymbol{Y})\}$

which is a witness that $a(X)/(A_\alpha + 1) = P_I(X)$ for some I.

Hence, if verification Step 3 in Prot. 1 succeeds for $j = i$, then, after replacing all coefficients with values derived in this proof, we get

$$\begin{aligned}
A(\boldsymbol{X}) =& a(X) + A_\varrho X_\varrho + A_\alpha\left(X_\alpha + P_0(X)\right) \ , \\
B(\boldsymbol{X}) =& \frac{a(X)}{(A_\alpha + 1)^2} + B_\varrho X_\varrho + \frac{A_\alpha(X_\alpha - P_0(X))}{A_\alpha + 1} \ .
\end{aligned}$$

Hence, Eq. (8) holds. □

5.2 Permutation Matrix Argument

Assume we explicitly compute $\mathfrak{A}_{n1} = \mathfrak{g}_1^{\sum_{i=1}^n P_i(\chi)} / \prod_{j=1}^{n-1} \mathfrak{A}_{j1}$ as in Prot. 1, and then apply the 1-sparsity argument to each $\mathfrak{A}_{i1}$, $i \in [1\,..\,n]$. Then, as in [33], we get that $(\mathfrak{A}_{11}, \ldots, \mathfrak{A}_{n1})$ commits to a permutation matrix. More precisely, according to Eq. (8), the ith commitment is represented by the polynomial

$$A_i(\boldsymbol{X}) = a_i(X) + A_{\varrho i} X_\varrho + A_{\alpha i} \cdot (X_\alpha + P_0(X)) \ ,$$

where $a_i(X)/(1 + A_{\alpha i}) = P_{f(i)}(X)$ for some f. Since $\sum_i A_i(\boldsymbol{X}) = \sum_i P_i(X)$, we get in particular that $\sum_i (A_{\alpha i} + 1) P_{f(i)}(X) = \sum_i P_i(X)$. Since due to Lem. 2, $\{P_i(X)\}_{i=0}^n$ is linearly independent, it means that $A_{\alpha i} = 0$ for each i, and $f = \sigma^{-1}$ is a permutation.

Theorem 3. *The described permutation matrix argument is perfectly complete. The following holds in the GBGM, assuming that the generic adversary works in polynomial time. If the honest verifier accepts Eq.* (1) *for all* $i \in [1\,..\,n]$*, and* $(\mathfrak{A}_{n1}, \mathfrak{A}_{n2})$ *is explicitly computed as in Prot. 1, then there exists a permutation* $\sigma \in S_n$ *and randomizers* $A_{\varrho i}$*, such that*

$$\mathfrak{A}_{i1} = \mathfrak{g}_1^{P_{\sigma^{-1}(i)}(\chi) + A_{\varrho i} \varrho} \quad \textit{for all } i \in [1\,..\,n] \ . \tag{10}$$

6 Validity Argument

The shuffle argument employs validity arguments for $(\boldsymbol{\pi}_{\mathsf{c2:1}}, \boldsymbol{\pi}_{\mathsf{c2:2}})$ and for each $(\mathfrak{v}'_{i1}, \mathfrak{v}'_{i2})$. We outline this argument for $(\boldsymbol{\pi}_{\mathsf{c2:1}}, \boldsymbol{\pi}_{\mathsf{c2:2}})$, the argument is the same for $(\mathfrak{v}'_{i1}, \mathfrak{v}'_{i2})$. More precisely, in the validity argument for $(\boldsymbol{\pi}_{\mathsf{c2:1}}, \boldsymbol{\pi}_{\mathsf{c2:2}})$, the verifier checks that $\hat{e}(\mathfrak{g}_1^\varrho, \pi_{\mathsf{c2:2}j}) = \hat{e}(\pi_{\mathsf{c2:1}j}, \mathfrak{g}_2^\beta)$ for $j \in [1\,..\,3]$. Thus, for

$$\mathcal{V}_{val:j}(\boldsymbol{X}) = E_{1j}(\boldsymbol{X}) X_\beta - X_\varrho E_{2j}(\boldsymbol{X}) \ ,$$

this argument guarantees that in the GBGM, $\mathcal{V}_{val:j}(\boldsymbol{X}) = 0$ for $j \in [1\,..\,3]$.

In this case, it is much easier to solve the resulting polynomial system of equations than it was in Sect. 5. First, we find the coefficients of $\mu(\boldsymbol{i}) = X_\alpha^{i_1} X_\varrho^{i_2} X_\beta^{i_3} X_\gamma^{i_4}$ in $\mathcal{V}_{val:j}(\boldsymbol{X}) \cdot X_\varrho X_\beta$, see Tbl. 3. Taking into account (see Lem. 2) that $\{P_i(X)\}_{i=0}^n$ are linearly independent and that $1 \notin \operatorname{span}\{P_i(X)\}_{i=1}^n$, we get from solving this polynomial system of equations that

$$\begin{aligned}
E_{1j}(\boldsymbol{X}) &= (E_{1j,\varrho\beta} + E_{2j,\beta} X_\beta + E_{2j,\gamma} X_\gamma) X_\varrho / X_\beta \ , \\
E_{2j}(\boldsymbol{X}) &= E_{1j,\varrho\beta} + E_{2j,\beta} X_\beta + E_{2j,\gamma} X_\gamma \ , \quad \text{and thus} \\
M_E(\boldsymbol{X}) &= M_{E:1}(\boldsymbol{X}) = M_{E:2}(\boldsymbol{X}) = E_{23}(\boldsymbol{X}) - E_{22}(\boldsymbol{X})/(X_\gamma + 1) - E_{21}(\boldsymbol{X})/X_\gamma \\
&= M_{E:1} + M_{E:2} X_\beta + M_{E:3} X_\gamma + \frac{M_{E:4}}{X_\gamma} + \frac{M_{E:5} X_\beta}{X_\gamma} + \frac{M_{E:6}}{X_\gamma + 1} + \frac{M_{E:7} X_\beta}{X_\gamma + 1}
\end{aligned} \tag{11}$$

Table 3. $\mathrm{coeff}_{\mu(\boldsymbol{i})}(\mathcal{V}_{val:j}(\boldsymbol{X}) \cdot X_\varrho X_\beta)$, where $\mu(\boldsymbol{i}) = X_\alpha^{i_1} X_\varrho^{i_2} X_\beta^{i_3} X_\gamma^{i_4}$

$\{i_1, \ldots, i_4\}$	$\mathrm{coeff}_{\mu(\boldsymbol{i})}(\mathcal{V}_{val:j}(\boldsymbol{X}) \cdot X_\varrho X_\beta)$
$\{1,2,1,0\}$	$E_{2j,\alpha}$
$\{1,1,2,0\}$	$E_{1j,\alpha}$
$\{0,3,1,0\}$	$-E_{2j,\varrho}$
$\{0,2,2,0\}$	$E_{1j,\varrho} - E_{2j,\beta}$
$\{0,2,1,1\}$	$E_{1j,\gamma} - E_{2j,\gamma}$
$\{0,2,1,0\}$	$E_{1j,\varrho\beta} - e_{2j}(X) - E_{2j,\alpha} P_0(X) - E_{2j,1}$
$\{0,1,2,0\}$	$e_{1j}(X) + (E_{1j,\alpha} + E_{1j,0})\, P_0(X)$
$\{0,0,2,0\}$	$Z(X) e_{1j}^\dagger(X)$

for some coefficients $M_{E:j}$ known to the adversary. Here, say $M_{E:2}(\boldsymbol{X}) = E_{23}(\boldsymbol{X}) - E_{22}(\boldsymbol{X})/(X_\gamma + 1) - E_{21}(\boldsymbol{X})/X_\gamma$; we will call such an operation a "generic decryption' in group $\mathbb{G}_k$.

Theorem 4. *The validity argument for* $(\boldsymbol{\pi}_{\mathsf{c2}:1}, \boldsymbol{\pi}_{\mathsf{c2}:2})$ *is perfectly complete. The following holds in the GBGM, assuming that the generic adversary works in polynomial time. If the honest verifier accepts Eq.* (2)*, then the generic adversary knows coefficients* $M_{E:j}$*, s.t.* $\mathsf{dec}_{\mathsf{sk}}(\boldsymbol{\pi}_{\mathsf{c2}}) = M_E(\boldsymbol{\chi})$ *where* $M_E(\boldsymbol{X})$ *is as in Eq.* (11)*.*

Assuming similarly that also validity of $V_{i1j}(\boldsymbol{X})$, $V_{i2j}(\boldsymbol{X})$, $V'_{i1j}(\boldsymbol{X})$, and $V'_{i2j}(\boldsymbol{X})$ is checked, we get that

$$
\begin{aligned}
V_{i1j}(\boldsymbol{X}) &= (V_{i1j,\varrho\beta} + V_{i2j,\beta} X_\beta + V_{i2j,\gamma} X_\gamma) X_\varrho / X_\beta \ , \\
V_{i2j}(\boldsymbol{X}) &= V_{i1j,\varrho\beta} + V_{i2j,\beta} X_\beta + V_{i2j,\gamma} X_\gamma \ , \\
V'_{i1j}(\boldsymbol{X}) &= (V'_{i1j,\varrho\beta} + V'_{i2j,\beta} X_\beta + V'_{i2j,\gamma} X_\gamma) X_\varrho / X_\beta \ , \\
V'_{i2j}(\boldsymbol{X}) &= V'_{i1j,\varrho\beta} + V'_{i2j,\beta} X_\beta + V'_{i2j,\gamma} X_\gamma \ , \quad \text{and thus} \\
M_i(\boldsymbol{X}) &= M_{i1}(\boldsymbol{X}) = M_{i2}(\boldsymbol{X}) = V_{i23}(\boldsymbol{X}) - V_{i22}(\boldsymbol{X})/(X_\gamma + 1) - V_{i21}(\boldsymbol{X})/X_\gamma \\
&= M_{i1} + M_{i2} X_\beta + M_{i3} X_\gamma + \frac{M_{i4}}{X_\gamma} + \frac{M_{i5} X_\beta}{X_\gamma} + \frac{M_{i6}}{X_\gamma + 1} + \frac{M_{i7} X_\beta}{X_\gamma + 1} \\
M'_i(\boldsymbol{X}) &= M'_{i1}(\boldsymbol{X}) = M'_{i2}(\boldsymbol{X}) = V'_{i23}(\boldsymbol{X}) - V'_{i22}(\boldsymbol{X})/(X_\gamma + 1) - V'_{i21}(\boldsymbol{X})/X_\gamma \\
&= M'_{i1} + M'_{i2} X_\beta + M'_{i3} X_\gamma + \frac{M'_{i4}}{X_\gamma} + \frac{M'_{i5} X_\beta}{X_\gamma} + \frac{M'_{i6}}{X_\gamma + 1} + \frac{M'_{i7} X_\beta}{X_\gamma + 1}
\end{aligned}
\tag{12}
$$

for some coefficients M_{ik}, $k \in [1\,..\,3]$, known to the adversary.

Corollary 2. *The validity argument for* $(\mathfrak{v}'_{i1}, \mathfrak{v}'_{i2})$ *is perfectly complete. The following holds in the GBGM, assuming that the generic adversary works in polynomial time. If the honest verifier accepts Eq.* (3) *for some* $i \in [1\,..\,n]$*, then the generic adversary knows coefficients* M'_{ij}*, s.t.* $\mathsf{dec}_{\mathsf{sk}}(\mathfrak{v}'_i) = M'_i(\chi)$ *where* $M'_i(\boldsymbol{X})$ *is as in Eq.* (12)*.*

7 Consistency Argument

We call the subargument of Prot. 1, where the verifier only executes the last verification (namely, Eq. (4)), the *consistency argument*. Intuitively, the consistency argument guarantees that the ciphertexts have been permuted by using the same permutation according to which the elements $\mathfrak{g}_k^{P_i(\chi)}$ were permuted inside the commitments $\mathfrak{A}_{i1}$.

According to Sect. 5 and Sect. 6, the permutation matrix argument and validity arguments are "sound". In what follows, we show that if the verifier executes all verification steps in Prot. 1, then this shuffle argument is sound in the GBGM. Now, we are finally able to finish the soundness proof of Thm. 1.

Proof (Of Thm. 1). Since all the batch verifications in Prot. 1 accept, by Cor. 1 we have that with probability $\geq 1-3/q$ all individual equations also hold. Since the permutation matrix argument and ciphertext validity are sound, Eq. (8) and Eq. (12) hold with overwhelming probability for all i.

Since we have a generic adversary, from the verification equation Eq. (4) we get that $\mathcal{V}_{cons:j}(\boldsymbol{X}) = 0$ for $j \in [1\,..\,3]$, where

$$\begin{aligned}
\mathcal{V}_{cons:1}(\boldsymbol{X}) =& \sum (V'_{i11}(\boldsymbol{X}) - V_{\sigma(i)11}(\boldsymbol{X}))P_i(X) - \sum V_{i11}(\boldsymbol{X})r_i X_\varrho \\
& - D_1(\boldsymbol{X})X_\gamma X_\varrho / X_\beta + E_{11}(\boldsymbol{X})X_\varrho \ , \\
\mathcal{V}_{cons:2}(\boldsymbol{X}) =& \sum (V'_{i12}(\boldsymbol{X}) - V_{\sigma(i)12}(\boldsymbol{X}))P_i(X) - \sum V_{i12}(\boldsymbol{X})r_i X_\varrho \\
& - D_2(\boldsymbol{X})(X_\gamma + 1)X_\varrho / X_\beta + E_{12}(\boldsymbol{X})X_\varrho \ , \\
\mathcal{V}_{cons:3}(\boldsymbol{X}) =& \sum (V'_{i13}(\boldsymbol{X}) - V_{\sigma(i)13}(\boldsymbol{X}))P_i(X) - \sum V_{i13}(\boldsymbol{X})r_i X_\varrho \\
& - D_1(\boldsymbol{X})X_\varrho / X_\beta - D_2(\boldsymbol{X})X_\varrho / X_\beta + E_{13}(\boldsymbol{X})X_\varrho
\end{aligned}$$

are rational functions. By doing a "generic decryption", define $M_i(\boldsymbol{X})$, $M'_i(\boldsymbol{X})$, and $M_E(\boldsymbol{X})$ as in Eq. (5). Then we get that $\mathcal{V}_{cons}(\boldsymbol{X}) = 0$, where

$$\begin{aligned}
\mathcal{V}_{cons}(\boldsymbol{X}) &= \frac{\mathcal{V}_{cons:3}(\boldsymbol{X})X_\beta}{X_\varrho} - \frac{\mathcal{V}_{cons:2}(\boldsymbol{X})X_\beta}{X_\varrho(X_\gamma+1)} - \frac{\mathcal{V}_{cons:1}(\boldsymbol{X})X_\beta}{X_\varrho X_\gamma} \\
&= \sum_i \left(M'_i(\boldsymbol{X}) - M_{\sigma(i)}(\boldsymbol{X})\right) P_i(X) - \left(\sum_i M_i(\boldsymbol{X})r_i - M_E(\boldsymbol{X})\right) X_\varrho = 0
\end{aligned}$$

is again a "generic decryption". Clearly, the last equality holds clearly independently of the shape of $D_j(\boldsymbol{X})$.

Now, since the validity argument is sound, $M_E(\boldsymbol{X})$ is as in Eq. (11) and $M_i(\boldsymbol{X})$ and $M'_i(\boldsymbol{X})$ are as in Eq. (12). Denote $\mathcal{V}^*_{cons}(\boldsymbol{X}) := X_\gamma(X_\gamma+1)$. Inserting the obtained representations of $M_i(\boldsymbol{X})$, $M'_i(\boldsymbol{X})$, and $M_E(\boldsymbol{X})$ to $\mathcal{V}_{cons}(\boldsymbol{X})$, we find the coefficients of $\mathcal{V}_{cons}(\boldsymbol{X}) \cdot \mathcal{V}^*_{cons}(\boldsymbol{X})$, as given in Tbl. 4.

Since $\{P_i(X)\}$ is linearly independent, this directly gives us $M_{\sigma(i)j} = M'_{ij}$, for each $j \in [2\,..\,5]$, and hence also for $j = 7$, as needed. In addition, we get that $M_{\sigma(i)1} + M_{\sigma(i)3} = M'_{i1} + M'_{i3}$ (and hence $M_{\sigma(i)1} = M'_{i1}$) and $M_{\sigma(i)1} + M_{\sigma(i)4} +$

Table 4. $\mathrm{coeff}_{\mu(\boldsymbol{i})}(\mathcal{V}_{cons}(\boldsymbol{X}) \cdot \mathcal{V}^*_{cons}(\boldsymbol{X}))$, where $\mu(\boldsymbol{i}) = X_\alpha^{i_1} X_\varrho^{i_2} X_\beta^{i_3} X_\gamma^{i_4}$

$\{i_1, \ldots, i_4\}$	$\mathrm{coeff}_{\mu(\boldsymbol{i})}(\mathcal{V}_{cons}(\boldsymbol{X}) \cdot \mathcal{V}^*_{cons}(\boldsymbol{X}))$
$\{0,1,1,2\}$	$M_{E:2} - \sum M_{i2} r_i$
$\{0,1,0,3\}$	$M_{E:3} - \sum M_{i3} r_i$
$\{0,1,0,0\}$	$M_{E:4} - \sum M_{i4} r_i$
$\{0,1,1,0\}$	$M_{E:5} - \sum M_{i5} r_i$
$\{0,1,1,1\}$	$M_{E:2} + M_{E:5} + M_{E:7} - \sum (M_{i2} + M_{i5} + M_{i7})\, r_i$
$\{0,1,0,2\}$	$M_{E:1} + M_{E:3} - \sum (M_{i1} + M_{i3})\, r_i$
$\{0,1,0,1\}$	$M_{E:1} + M_{E:4} + M_{E:6} - \sum (M_{i1} + M_{i4} + M_{i6})\, r_i$
$\{0,0,1,2\}$	$\sum (M'_{i2} - \sum M_{\sigma(i)2}) P_i(X)$
$\{0,0,0,3\}$	$\sum (M'_{i3} - \sum M_{\sigma(i)3}) P_i(X)$
$\{0,0,0,0\}$	$\sum (M'_{i4} - \sum M_{\sigma(i)4}) P_i(X)$
$\{0,0,1,0\}$	$\sum (M'_{i5} - \sum M_{\sigma(i)5}) P_i(X)$
$\{0,0,0,2\}$	$\sum \big((M'_{i1} + M'_{i3}) - \big(M_{\sigma(i)1} + M_{\sigma(i)3}\big)\big)\, P_i(X)$
$\{0,0,0,1\}$	$\sum \big((M'_{i1} + M'_{i4} + M'_{i6}) - \big(M_{\sigma(i)1} + M_{\sigma(i)4} + M_{\sigma(i)6}\big)\big)\, P_i(X)$
$\{0,0,1,1\}$	$\sum \big((M'_{i2} + M'_{i5} + M'_{i7}) - \big(M_{\sigma(i)2} + M_{\sigma(i)5} + M_{\sigma(i)7}\big)\big)\, P_i(X)$

$M_{\sigma(i)6} = M'_{i1} + M'_{i4} + M'_{i6}$ (and hence $M_{\sigma(i)6} = M'_{i6}$). Hence, we have proven that $M_{\sigma(i)}(\boldsymbol{X}) = M'_i(\boldsymbol{X})$ as a polynomial, which gives us soundness of the new shuffle argument in the GBGM.

Let us now compute a lower bound to the efficiency of a generic adversary. Assume that after some τ steps, the adversary has made a successful equality query $(=, i_1, i_2)$, i.e., $\mathsf{cell}_{i_1} = \mathsf{cell}_{i_2}$ for $i_1 \neq i_2$. Hence, she has found a collision $B_1(\boldsymbol{\chi}) = B_2(\boldsymbol{\chi})$ such that $B_1(\boldsymbol{X}) \neq B_2(\boldsymbol{X})$. If $\mathsf{type}_{i_1} \in \{1, T\}$ (this is not needed for group $\mathbb{G}_2$, since we do not have rational functions there), then redefine $B_j(\boldsymbol{X}) := B_j(\boldsymbol{X}) \cdot X_\varrho X_\beta$, this guarantees $B_j(\boldsymbol{X})$ is a polynomial. Thus,

$$B_1(\boldsymbol{\chi}) - B_2(\boldsymbol{\chi}) \equiv 0 \pmod{q} \ . \tag{13}$$

Note that

- If $\mathsf{type}_{i_1} = 1$, then $\deg B_j(\boldsymbol{X}) \leq 2n + 1 =: d_1$,
- If $\mathsf{type}_{i_1} = 2$, then $\deg B_j(\boldsymbol{X}) \leq n =: d_2$, and thus
- If $\mathsf{type}_{i_1} = T$, then $\deg B_j(\boldsymbol{X}) \leq (2n+1) + n = 3n + 1 =: d_T$.

Due to the Schwartz-Zippel lemma, since $\boldsymbol{\chi}$ is chosen uniformly random from $\mathbb{Z}_q^2 \times (\mathbb{Z}_q \setminus \{0\})^2 \times (\mathbb{Z}_q \setminus \{0, -1\})$, and since $B_1(\boldsymbol{X}) \neq B_2(\boldsymbol{X})$ as a polynomial, Eq. (13) holds with probability at most $\deg B_j(\boldsymbol{X})/(q-2) \leq d_{\mathsf{type}_{i_1}}/(q-2)$. Clearly, an adversary working in time τ can generate up to τ new group elements. Then the probability that there exists a collision between any two of those group elements is upper bounded by $\binom{\tau}{2} \cdot \deg B_j(\boldsymbol{X})/(q-2) \leq \binom{\tau}{2} \cdot d_{\mathsf{type}_{i_1}}/(q-2) \leq \tau^2/2 \cdot d_{\mathsf{type}_{i_1}}/(q-2)$. Thus, a successful adversary on average requires time at least

$$\tau^2 \geq 2(q-2)/d_{\mathsf{type}_{i_1}} \geq 2(q-2)/d_T = 2(q-2)/(3n+1)$$

to produce a collision. Simplifying, we get $\tau \in \Omega(\sqrt{q/n})$. □

8 Zero-Knowledge

Theorem 5. *The new shuffle argument is perfectly zero knowledge.*

Proof. Consider the simulator Sim that, given the CRS crs, the trapdoor $\mathsf{td} = (\chi, \varrho)$, and input $(\mathfrak{v}, \mathfrak{v}')$, simulates the prover in the shuffle argument. If the simulator can create an accepting argument with correct distribution for *any* $(\mathfrak{v}, \mathfrak{v}')$, this means that an accepting argument provides no information on $(\mathfrak{v}, \mathfrak{v}')$ or the relation between the two sets of ciphertext.

The complete simulator construction is as follows:

1. For $i = 1$ to $n-1$:
   (a) Set $r_i \leftarrow_r \mathbb{Z}_q$. Compute $(\mathfrak{A}_{i1}, \mathfrak{A}_{i2}) \leftarrow (\mathfrak{g}_1, \mathfrak{g}_2)^{P_i(\chi)+r_i\varrho}$.
2. Set $r_n \leftarrow -\sum_{i=1}^{n-1} r_i$.
3. Set $(\mathfrak{A}_{n1}, \mathfrak{A}_{n2}) \leftarrow (\mathfrak{g}_1, \mathfrak{g}_2)^{\sum_{i=1}^{n} P_i(\chi)} / \prod_{i=1}^{n-1} (\mathfrak{A}_{i1}, \mathfrak{A}_{i2})$.
4. For $i = 1$ to n: Compute
$$\pi_{\mathsf{1sp}:i} \leftarrow \left(\mathfrak{A}_{i1}\mathfrak{g}_1^{P_0(\chi)}\right)^{2r_i} (\mathfrak{g}_1^{\varrho})^{-r_i^2} \mathfrak{g}_1^{((P_i(X)+P_0(X))^2-1)/\varrho} \quad .$$
5. Set $\boldsymbol{r}_s \leftarrow_r \mathbb{Z}_q^2$. Set $\pi_{\mathsf{c1}:1} \leftarrow \mathfrak{g}_2^{r_{s:1}\varrho}$, $\pi_{\mathsf{c1}:2} \leftarrow \mathfrak{g}_2^{r_{s:2}\varrho}$. (I.e., they commit to $\mathbf{0}$.)
6. Compute $\boldsymbol{\pi}_{\mathsf{c2}:1} \leftarrow \prod_{i=1}^{n} (\mathfrak{v}_{i1}^{r_i+P_i(\chi)/\varrho} / (\mathfrak{v}'_{i1})^{P_i(\chi)/\varrho}) \cdot \mathsf{enc}_{\mathsf{pk}_1}(0; \boldsymbol{r}_s)$.
   Compute $\boldsymbol{\pi}_{\mathsf{c2}:2} \leftarrow \prod_{i=1}^{n} (\mathfrak{v}_{i2}^{r_i+P_i(\chi)/\varrho} / (\mathfrak{v}'_{i2})^{P_i(\chi)/\varrho}) \cdot \mathsf{enc}_{\mathsf{pk}_2}(0; \boldsymbol{r}_s)$.
7. Return $\pi_{sh} := (\mathfrak{v}', (\mathfrak{A}_{i1}, \mathfrak{A}_{i2})_{i=1}^{n-1}, (\pi_{\mathsf{1sp}:i})_{i=1}^{n}, \pi_{\mathsf{c1}:1}, \pi_{\mathsf{c1}:2}, \boldsymbol{\pi}_{\mathsf{c2}:1}, \boldsymbol{\pi}_{\mathsf{c2}:2})$.

The simulator calculates all values $(\mathfrak{A}_{i1}, \mathfrak{A}_{i2})_{i=1}^{n-1}, (\pi_{\mathsf{1sp}:i})_{i=1}^{n}, \boldsymbol{\pi}_{\mathsf{c1}}$ exactly as an honest prover would have when $\sigma = \mathrm{Id}$, and hence these values will have the same distribution as the same values computed by an honest prover. Since the commitment scheme is obviously perfectly hiding, these values have the same distribution independently of the choice of σ. Moreover, there is a unique pair of values $\boldsymbol{\pi}_{\mathsf{c2}:1}, \boldsymbol{\pi}_{\mathsf{c2}:2}$ that satisfy Eq. (2) and Eq (4). (Computing $\boldsymbol{\pi}_{\mathsf{c2}:1}$ and $\boldsymbol{\pi}_{\mathsf{c2}:2}$ is the only place in the simulation where one needs the trapdoor $\mathsf{td} = (\chi, \varrho)$.) Thus we are left to show that our chosen values satisfy these two equations.

But assuming the ciphertexts are valid, Eq. (2) trivially holds. We get $\hat{e}(\mathfrak{P}_{11}^{r_{s:1}}, \mathfrak{g}_2^{\varrho}) = \hat{e}(\mathfrak{P}_{11}, \pi_{\mathsf{c1}:1})$ and $\hat{e}(\mathfrak{P}_{12}^{r_{s:2}}, \mathfrak{g}_2^{\varrho}) = \hat{e}(\mathfrak{P}_{12}, \pi_{\mathsf{c1}:2})$. Hence,

$$\begin{aligned}
&\prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}'_{i1}, \mathfrak{g}_2^{P_i(\chi)}\right) / \prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}_{i1}, \mathfrak{A}_{i2}\right) \\
&\qquad = \prod_{i=1}^{n} \hat{e}\left((\mathfrak{v}'_{i1})^{P_i(\chi)}, \mathfrak{g}_2\right) / \prod_{i=1}^{n} \hat{e}\left(\mathfrak{v}_{i1}^{P_i(\chi)+r_i\varrho}, \mathfrak{g}_2\right) \\
&\qquad = \hat{e}\left(\prod_{i=1}^{n} ((\mathfrak{v}'_{i1})^{P_i(\chi)/\varrho} / \mathfrak{v}_{i1}^{P_i(\chi)/\varrho+r_i}), \mathfrak{g}_2^{\varrho}\right) \\
&\qquad = \hat{e}\left(\mathfrak{P}_{11}, \pi_{\mathsf{c1}:1}\right) \hat{e}\left(\mathfrak{P}_{12}, \pi_{\mathsf{c1}:2}\right) / \hat{e}\left(\boldsymbol{\pi}_{\mathsf{c2}:1}, \mathfrak{g}_2^{\varrho}\right) \quad ,
\end{aligned}$$

so the verifier will accept the shuffle argument. As the simulator did not know anything about the honest prover's permutation σ, the shuffle argument is thus perfectly zero knowledge. □

9 Efficiency

We use exponentiation speed records from [10] and pairing speed records from [2] for Barreto-Naehrig curves. According to Tbl. 4 in [10], a pairing, exponentiation in $\mathbb{G}_1$, exponentiation in $\mathbb{G}_2$, and exponentiation in $\mathbb{G}_T$ take respectively 7.0, 0.9, 1.8, and 3.1 million clock cycles on the Core i7-3520M CPU. This does *not* take into account the possible speed-ups by employing fast fixed-based exponentiation or multi-exponentiation algorithms. Thus, all following comparisons are imprecise, and just to give a gut feeling about the difference. They also depend on the known speed records on implementing pairings and exponentiations.

Prover's computation:
- Step 1: $n-1$ exponentiations in $\mathbb{G}_1$, $n-1$ exponentiations in $\mathbb{G}_2$.
- Step 4: $2n$ exponentiations in $\mathbb{G}_1$.
- Step 5a: $3n$ exponentiations in $\mathbb{G}_1$, $3n$ exponentiations in $\mathbb{G}_2$.
- Step 6a: $2n+2$ exponentiations in $\mathbb{G}_2$.
- Step 6b: $3n+3$ exponentiations in $\mathbb{G}_1$, $3n+3$ exponentiations in $\mathbb{G}_2$.

Hence, the prover executes $9n+2$ exponentiations in $\mathbb{G}_1$ and $9n+4$ exponentiations in $\mathbb{G}_2$. Here, *all* costly (i.e., at least n-wide) exponentiations can be written as either multi-exponentiations or fixed-base exponentiations — e.g., Step 5a — and are hence relatively cheap. The only exception is the computation of general exponentiation in $(\mathfrak{A}_{i1}\mathfrak{g}_1^{P_0(\chi)})^{2r_i}$ for $i \in [1\,..\,n]$. Taking n million clock cycles as the basic unit (and *not* taking into account possible speed-ups by employing fast multi-exponentiation and fixed-base exponentiation algorithms), the prover's computation is dominated by $9 \cdot 0.9 + 9 \cdot 1.8 = 24.3$ units.

Verifier's computation: by using batching techniques [4, 31], we reduced the number of pairings by introducing a number of exponentiations either in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$. The verifier does:
- Step 3: $2n$ exponentiations in $\mathbb{G}_1$, 1 exponentiation in $\mathbb{G}_T$, and $n+1$ pairings.
- Step 4: $3n+3$ exponentiations in $\mathbb{G}_1$, $3n+3$ exponentiations in $\mathbb{G}_2$, and 2 pairings.
- Step 5: 2 exponentiations in $\mathbb{G}_1$, 3 exponentiations in $\mathbb{G}_2$ ($\pi_{\mathsf{c1};2}^{p_{42}}$ is reused), and 3 pairings.
- Step 6: $3n+3n=6n$ exponentiations in $\mathbb{G}_1$, and $n+n=2n$ pairings.

In total, the verifier has to do $11n+5$ exponentiations in $\mathbb{G}_1$, $3n+6$ exponentiations in $\mathbb{G}_2$, 1 exponentiation in $\mathbb{G}_T$, and $3n+6$ pairings. Taking n million clock cycles as the basic unit, the verifier's computation is dominated by $11 \cdot 0.9 + 3 \cdot 1.8 + 3 \cdot 7.0 = 36.3$ units; around 58% (21 units) of this is the cost of pairings. Also here, most of the exponentiations are multi-exponentiations or fixed-base exponentiations.

Communication: $3n + (n-1) + n + 3 = 5n+2$ elements from $\mathbb{G}_1$ and $3n + (n-1) + 2 + 3 = 4n+4$ elements from $\mathbb{G}_2$, that is, $9n+6$ group elements.

CRS length (excluding gk): $2n + 6$ elements from $\mathbb{G}_1$, $n + 6$ elements from $\mathbb{G}_2$, and 1 element from $\mathbb{G}_T$, that is, $3n + 13$ group elements.

Comparison with prior work. To compare, the verifier's computation in [33] (resp., [19]) is dominated by $28 \cdot 7.0 = 196$ (resp., $18 \cdot 7.0 = 126$) units. Hence, the verification of the new shuffle is effectively about 5.4 (resp., 3.5) times faster than that of the Lipmaa-Zhang (resp., Fauzi-Lipmaa) shuffle. Since verification is a bottleneck of mix-nets, this constitutes of a major improvement.

In [33], the prover's computation is dominated by $28n + 11$ exponentiations, $16n + 6$ in $\mathbb{G}_1$ and $12n + 5$ in $\mathbb{G}_2$ (this also includes reshuffling) which yields $16 \cdot 0.9 + 12 \cdot 1.8 = 36$ units. In [19], the prover's computation is dominated by $18n + 3$ exponentiations, $14n + 3$ in $\mathbb{G}_1$ and $4n$ in $\mathbb{G}_2$ (this also includes reshuffling) which yields $14 \cdot 0.9 + 4 \cdot 1.8 = 19.8$ units. Hence, in the new protocol, the prover is about 1.5 times more efficient compared to [33], but about 1.2 times less efficient compared to [19].

As mentioned above, the most efficient shuffle scheme up to now [25] works in the random oracle model which allows to obtain better computational complexity both for the prover ($6 \cdot 0.9 = 5.4$ units) and verifier ($6 \cdot 0.9 = 5.4$ units), assuming that computation is done in $\mathbb{G}_1$. In reality, non pairing-friendly groups have usually somewhat faster arithmetic than pairing-friendly groups. Hence, there is still a significant gap.

10 On GBGM versus Knowledge Assumptions

A knowledge assumption guarantees that if an adversary, given an input (that includes the CRS and some auxiliary input), outputs some values then there exists an extractor running on the same input that outputs the same values together with some witness. Following [15], each input to the knowledge assumption has a well-defined knowledge component. Apart from that, the precise definition of a knowledge assumption is left to the imagination of its proposers. However, it is known that knowledge assumptions are unacceptable if the auxiliary input is not well chosen [5], and hence special care has to be taken when defining them.

In contrast, in the GBGM, the adversary can compute output values as a product or pairing of given inputs (and other previously computed values), so it is assumed that she knows a polynomial relationship between the discrete logarithms of its outputs and inputs. There is little need for imagination of how to define the GBGM, since this has been done before in sufficient detail [38, 34]. The known impossibility results about the generic (bilinear) group model [21, 17] use quite contrived constructions.

We think that GBGM is preferable to knowledge assumptions, hence Tbl. 1 has a highlighted cell for arguments that do not use knowledge assumptions. The validity of knowledge assumptions can and should be proven in the GBGM anyhow; indeed, one should be very suspicious of knowledge assumptions that cannot be proven in the GBGM. However, this should be done very carefully, taking into account the precise shape of the CRS and the adversary's auxiliary

input. To guarantee correct use of a knowledge assumption, we think that it is prudent that one proves in the GBGM the security of the knowledge assumption given the auxiliary string the adversary gets in the concrete application. This seems to hint that one should reprove in the GBGM the security of all used knowledge assumptions in each individual paper.

Instead of proving the security of non-falsifiable knowledge assumptions (on top of several novel computational assumptions like PP and SP [28] or PSP [19]) in the GBGM, and then using such assumptions in the security proof, we think it is more reasonable to work directly in the GBGM. Moreover, GBGM model arguments tend to be more efficient, in particular since there is a reduced need to compute the knowledge components.

In fact, most of the known knowledge assumptions make a very specific use of the power of the GBGM. E.g., reinterpreting the knowledge assumptions used in say [19] in the language of GBGM, one assumes for a *specific (unique!)* random variable X_k the following holds: for any polynomial F, if $F(X_1, \ldots, X_k, \ldots, X_m) = 0$ and $F(X_1, \ldots, X_k, \ldots, X_m)$ has μ_k as a coefficient of X_k then $\mu_k = 0$. It is questionable why this concrete coefficient is handled differently from all other coefficients; in the GBGM, from $F(X_1, \ldots, X_m) = 0$ one can derive that *all* coefficients $\mu_{i_1,\ldots,i_m}$ of $F(X_1, \ldots, X_m) = \sum \mu_{i_1,\ldots,i_m} X_1^{i_1} \ldots X_m^{i_m}$ are equal to 0.

Acknowledgment. We would like to thank Jens Groth for useful discussion. The authors were supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 653497 (project PANORAMIX), and by institutional research funding IUT2-1 of the Estonian Ministry of Education and Research.

A Preliminaries: 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 $\mathcal{L}_n$. Here, as in [28], since we argue about group elements, both $\mathcal{L}_n$ and $\mathcal{R}_n$ are group-dependent and thus we add gk as an input to $\mathcal{L}_n$ and $\mathcal{R}_n$. Let $\mathcal{R}_n(\mathsf{gk}) := \{(u, w) : (\mathsf{gk}, u, w) \in \mathcal{R}_n\}$.

A *non-interactive argument* for a group-dependent relation family $\mathcal{R}$ consists of four PPT algorithms: a setup algorithm setup, a common reference string (CRS) generator gencrs, a prover pro, and a verifier ver. For $\mathsf{gk} \leftarrow \mathsf{setup}(1^\kappa, n)$ (where n is the input length) and $(\mathsf{crs} = (\mathsf{crs}_p, \mathsf{crs}_v), \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk})$ (where td is not accessible to anybody but the simulator), $\mathsf{pro}(\mathsf{crs}_p; u, w)$ produces an argument π, and $\mathsf{ver}(\mathsf{crs}_v; u, \pi)$ outputs either 1 (accept) or 0 (reject). Here, crs_p (resp., crs_v) is the part of the CRS given to the prover (resp., the verifier). Distinction between crs_p and crs_v is not important from the security point of view, but in many cases crs_v is significantly shorter.

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

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{setup}(1^\kappa, n), ((\mathsf{crs}_p, \mathsf{crs}_v), \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), (u, w) \leftarrow \mathcal{R}_n(\mathsf{gk}) : \\ \mathsf{ver}(\mathsf{gk}, \mathsf{crs}_v; u, \mathsf{pro}(\mathsf{gk}, \mathsf{crs}_p; u, w)) = 1\end{array}\right] = 1 \ .$$

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

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{setup}(1^\kappa, n), ((\mathsf{crs}_p, \mathsf{crs}_v), \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), \\ (u, \pi) \leftarrow \mathsf{adv}(\mathsf{gk}, \mathsf{crs}_p, \mathsf{crs}_v) : (\mathsf{gk}, u) \notin \mathcal{L}_n \wedge \mathsf{ver}(\mathsf{gk}, \mathsf{crs}_v; u, \pi) = 1\end{array}\right] \approx_\kappa 0 \ .$$

We recall that in situations where the inputs have been committed by 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 proof culpable soundness or the argument of knowledge property.

Ψ is adaptively *computationally culpably sound* [28, 29] for $\mathcal{L}$, if for all $n = \text{poly}(\kappa)$, for all polynomial-time decidable binary relations $\mathcal{R}^{\mathsf{guilt}} = \{\mathcal{R}_n^{\mathsf{guilt}}\}$ consisting of elements from $\bar{\mathcal{L}}$ and witnesses w^{guilt}, and for all non-uniform probabilistic polynomial-time adv,

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{setup}(1^\kappa, n), ((\mathsf{crs}_p, \mathsf{crs}_v), \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), \\ (u, \pi, w^{\mathsf{guilt}}) \leftarrow \mathsf{adv}(\mathsf{gk}, \mathsf{crs}_p, \mathsf{crs}_v) : \\ (\mathsf{gk}, u, w^{\mathsf{guilt}}) \in \mathcal{R}_n^{\mathsf{guilt}} \wedge \mathsf{ver}(\mathsf{gk}, \mathsf{crs}_v; u, \pi) = 1\end{array}\right] \approx_\kappa 0 \ .$$

For algorithms adv and X_{adv}, we write $(y; y') \leftarrow (\mathsf{adv}||X_{\mathsf{adv}})(\chi)$ if adv on input χ outputs y, and X_{adv} on the same input (including the random tape of adv) outputs y'.

Ψ is *an argument of knowledge*, if for all $n = \text{poly}(\kappa)$ and every non-uniform probabilistic polynomial-time adv, there exists a non-uniform probabilistic polynomial-time extractor X, s.t. for every auxiliary input $\mathsf{aux} \in \{0,1\}^{\text{poly}(\kappa)}$,

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{setup}(1^\kappa, n), ((\mathsf{crs}_p, \mathsf{crs}_v), \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), \\ ((u, \pi); w) \leftarrow (\mathsf{adv}||X_{\mathsf{adv}})(\mathsf{crs}_p, \mathsf{crs}_v; \mathsf{aux}) : \\ (u, w) \notin \mathcal{R} \wedge \mathsf{ver}(\mathsf{crs}_v; u, \pi) = 1\end{array}\right] \approx_\kappa 0 \ .$$

Here, aux can be seen as the common auxiliary input to adv and X_{adv} that is generated by using benign auxiliary input generation [5].

Ψ is *perfectly zero-knowledge*, if there exists a probabilistic polynomial-time simulator $\mathcal{X}_\gamma$, such that for all stateful non-uniform probabilistic polynomial-time adversaries adv and $n = \text{poly}(\kappa)$,

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{setup}(1^\kappa, n), \\ ((\mathsf{crs}_p, \mathsf{crs}_v), \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), \\ (u, w) \leftarrow \mathsf{adv}(\mathsf{gk}, \mathsf{crs}_p, \mathsf{crs}_v), \\ \pi \leftarrow \mathsf{pro}(\mathsf{gk}, \mathsf{crs}_p; u, w) : \\ (\mathsf{gk}, u, w) \in \mathcal{R}_n \wedge \mathsf{adv}(\mathsf{gk}, \pi) = 1\end{array}\right] = \Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{setup}(1^\kappa, n), \\ ((\mathsf{crs}_p, \mathsf{crs}_v); \mathsf{td}) \leftarrow \mathsf{gencrs}(\mathsf{gk}), \\ (u, w) \leftarrow \mathsf{adv}(\mathsf{gk}, \mathsf{crs}_p, \mathsf{crs}_v), \\ \pi \leftarrow \mathcal{X}_\gamma(\mathsf{gk}, \mathsf{crs}_p, \mathsf{crs}_v; u, \mathsf{td}) : \\ (\mathsf{gk}, u, w) \in \mathcal{R}_n \wedge \mathsf{adv}(\mathsf{gk}, \pi) = 1\end{array}\right]$$

Here, the prover and the simulator use the same CRS. That is, we have *same-string zero knowledge*. A same-string statistical zero knowledge argument stay secure even when using the CRS an unbounded number of times.

References

1. Ambrona, M., Barthe, G., Schmidt, B.: Automated Unbounded Analysis of Cryptographic Constructions in the Generic Group Model. In: EUROCRYPT 2016. LNCS, vol. 9666, pp. 822–851
2. Aranha, D.F., Barreto, P.S.L.M., Longa, P., Ricardini, J.E.: The Realm of the Pairings. In: SAC 2013. LNCS, vol. 8282, pp. 3–25
3. Barthe, G., Fagerholm, E., Fiore, D., Scedrov, A., Schmidt, B., Tibouchi, M.: Strongly-Optimal Structure Preserving Signatures from Type II Pairings: Synthesis and Lower Bounds. In: PKC 2015. LNCS, vol. 9020, pp. 355–376
4. Bellare, M., Garay, J.A., Rabin, T.: Batch Verification with Applications to Cryptography and Checking. In: LATIN 1998. LNCS, vol. 1380, pp. 170–191
5. Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the Existence of Extractable One-Way Functions. In: STOC 2014, pp. 505–514
6. Bitansky, N., Dachman-Soled, D., Garg, S., Jain, A., Kalai, Y.T., Lopez-Alt, A., Wichs, D.: Why "Fiat-Shamir for Proofs" Lacks a Proof. In: TCC 2013. LNCS, vol. 7785, pp. 182–201
7. Blum, M., Feldman, P., Micali, S.: Non-Interactive Zero-Knowledge and Its Applications. In: STOC 1988, pp. 103–112
8. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456
9. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: CRYPTO 2004. LNCS, vol. 3152, pp. 41–55
10. Bos, J.W., Costello, C., Naehrig, M.: Exponentiating in Pairing Groups. In: SAC 2013. LNCS, vol. 8282, pp. 438–455
11. Buchberger, B.: An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal. PhD thesis, University of Innsbruck (1965)
12. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited. In: STOC 1998, pp. 209–218
13. Chaabouni, R., Lipmaa, H., Zhang, B.: A Non-Interactive Range Proof with Constant Communication. In: FC 2012. LNCS, vol. 7397, pp. 179–199
14. Ciampi, M., Persiano, G., Siniscalchi, L., Visconti, I.: A Transform for NIZK Almost as Efficient and General as the Fiat-Shamir Transform Without Programmable Random Oracles. In: TCC 2016-A (2). LNCS, vol. 9563, pp. 83–111
15. Damgård, I.: Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In: CRYPTO 1991. LNCS, vol. 576, pp. 445–456
16. Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square Span Programs with Applications to Succinct NIZK Arguments. In: ASIACRYPT 2014 (1). LNCS, vol. 8873, pp. 532–550
17. Dent, A.W.: Adapting the Weaknesses of the Random Oracle Model to the Generic Group Model. In: ASIACRYPT 2002. LNCS, vol. 2501, pp. 100–109
18. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An Algebraic Framework for Diffie-Hellman Assumptions. In: CRYPTO (2) 2013. LNCS, vol. 8043, pp. 129–147

19. Fauzi, P., Lipmaa, H.: Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles. In: CT-RSA 2016. LNCS, vol. 9610, pp. 200–216
20. Fauzi, P., Lipmaa, H., Zhang, B.: Efficient Modular NIZK Arguments from Shift and Product. In: CANS 2013. LNCS, vol. 8257, pp. 92–121
21. Fischlin, M.: A Note on Security Proofs in the Generic Model. In: ASIACRYPT 2000. LNCS, vol. 1976, pp. 458–469
22. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic Span Programs and NIZKs without PCPs. In: EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645
23. Goldwasser, S., Kalai, Y.T.: On the (In)security of the Fiat-Shamir Paradigm. In: FOCS 2003, pp. 102–113
24. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. In: STOC 1985, pp. 291–304
25. Groth, J.: A Verifiable Secret Shuffle of Homomorphic Encryptions. J. Cryptology **23**(4) (2010) pp. 546–579
26. Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340
27. Groth, J.: On the Size of Pairing-based Non-interactive Arguments. In: EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326
28. Groth, J., Lu, S.: A Non-interactive Shuffle with Pairing Based Verifiability. In: ASIACRYPT 2007. LNCS, vol. 4833, pp. 51–67
29. Groth, J., Ostrovsky, R., Sahai, A.: New Techniques for Noninteractive Zero-Knowledge. Journal of the ACM **59**(3) (2012)
30. Lipmaa, H.: Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In: TCC 2012. LNCS, vol. 7194, pp. 169–189
31. Lipmaa, H.: Prover-Efficient Commit-And-Prove Zero-Knowledge SNARKs. In: AFRICACRYPT 2016. LNCS, vol. 9646, pp. 185–206
32. Lipmaa, H., Zhang, B.: A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. In: SCN 2012. LNCS, vol. 7485, pp. 477–502
33. Lipmaa, H., Zhang, B.: A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. Journal of Computer Security **21**(5) (2013) pp. 685–719
34. Maurer, U.M.: Abstract Models of Computation in Cryptography. In: Cryptography and Coding 2005, pp. 1–12
35. Naor, M.: On Cryptographic Assumptions and Challenges. In: CRYPTO 2003. LNCS, vol. 2729, pp. 96–109
36. Nielsen, J.B.: Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In: CRYPTO 2002. LNCS, vol. 2442, pp. 111–126
37. Schwartz, J.T.: Fast Probabilistic Algorithms for Verification of Polynomial Identities. Journal of the ACM **27**(4) (1980) pp. 701–717
38. Shoup, V.: Lower Bounds for Discrete Logarithms and Related Problems. In: EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266
39. Zippel, R.: Probabilistic Algorithms for Sparse Polynomials. In: EUROSM 1979. LNCS, vol. 72, pp. 216–226