An extended abstract of this paper appears in ACM CCS '19. This is the full version.

Succinct Arguments for Bilinear Group Arithmetic: Practical Structure-Preserving Cryptography

Russell W. F. Lai
Friedrich-Alexander University
Erlangen-Nuremberg

Giulio Malavolta
Carnegie Mellon University

Viktoria Ronge
Friedrich-Alexander University
Erlangen-Nuremberg

ABSTRACT

In their celebrated work, Groth and Sahai [EUROCRYPT'08, SICOMP' 12] constructed non-interactive zero-knowledge (NIZK) proofs for general bilinear group arithmetic relations, which spawned the entire subfield of structure-preserving cryptography. This branch of the theory of cryptography focuses on modular design of advanced cryptographic primitives. Although the proof systems of Groth and Sahai are a powerful toolkit, their efficiency hits a barrier when the size of the witness is large, as the proof size is linear in that of the witness.

In this work, we revisit the problem of proving knowledge of general bilinear group arithmetic relations in zero-knowledge. Specifically, we construct a succinct zero-knowledge argument for such relations, where the communication complexity is logarithmic in the integer and source group components of the witness. Our argument has public-coin setup and verifier and can therefore be turned non-interactive using the Fiat-Shamir transformation in the random oracle model. For the special case of non-bilinear group arithmetic relations with only integer unknowns, our system can be instantiated in non-bilinear groups. In many applications, our argument system can serve as a drop-in replacement of Groth-Sahai proofs, turning existing advanced primitives in the vast literature of structure-preserving cryptography into practically efficient systems with short proofs.

KEYWORDS

succinct arguments, structure-preserving cryptography

1 INTRODUCTION

Non-interactive zero-knowledge proofs (NIZK) have been shown to be an extremely versatile and powerful tool in the construction of secure cryptographic protocols and have been the objective of a large body of research in the theory of cryptography. The seminal result of Blum, Feldman, and Micali [11] showed that all languages in NP admit a polynomial-time NIZK, assuming the existence of trapdoor permutations. This has spawned a very fruitful line of research that explores the feasibility of generic NIZKs under stronger definitions [51] and different assumptions [31]. The de-facto methodology to build such systems is to consider a specific NP-complete problem, e.g., Circuit Satisfiability, and build a proof system for it. This approach however comes at the intrinsic cost of transforming the statement via an NP-reduction, which is typically a very expensive step and is often the efficiency bottleneck.

1.1 NIZK for Bilinear Group Arithmetic

Motivated by this shortcoming and seeking for practically efficient system, many works have focused on designing NIZKs for specific (and practically relevant) languages, such as NIZKs for the knowledge of discrete logarithms [53], proofs of plaintext knowledge [16], range proofs [14] and many others. The most prominent example in this area is the breakthrough result of Groth and Sahai [32, 33], who constructed efficient non-interactive witness-indistinguishable (NIWI) and NIZK[1] proof systems for algebraic relations in bilinear groups, a recurrent structure in the design of group-based cryptographic objects [12, 26, 52]. The Groth-Sahai (GS) proofs were the first examples of practically efficient systems for an expressive language and had a tremendous impact: The whole subfield of *structure-preserving cryptography* (*e.g.*, [1–3, 15, 22, 42, 44]) specializes in designing basic cryptographic primitives (*e.g.*, digital signatures and encryption schemes) that consists exclusively of bilinear group operations, and compose them with Groth-Sahai proofs to construct more advanced primitives (*e.g.*, group signatures, anonymous credentials). The advantages of this modular approach are twofold:

(1) It allows one to avoid the high cost of NP reductions needed for using general purpose NIZK.
(2) It allows one to modularly compose cryptographic building blocks to construct larger systems, reducing the necessity for ad-hoc (and error-prone) solutions.

While GS proofs offer a very powerful toolkit, their efficiency hits a barrier when proving statements with large witnesses: The size of a proof grows linearly with the size of the underlying witness. This issue becomes especially relevant when the proof is required to be published on a bulletin board (e.g., a blockchain) of limited capacity and the proof size influences the monetary cost of making it publicly available. As an example, consider the scenario where a user wants to prove the knowledge of n message-signature pairs, where the signatures are possibly under different public keys. The combination of structure-preserving signatures and GS proofs would lead to proofs of size linear in n. We stress that the dependency on the witness size is not an artifact of GS proofs but seems to be inherent for all system based on standard (falsifiable) assumptions [24].

In this work we revisit the question of efficient zero-knowledge for bilinear group arithmetics and we propose an efficient *argument system* for such relations. In contrast to a proof, an argument is only computationally sound (*i.e.*, an unbounded prover could prove potentially wrong statements). On the brighter side, the relaxation in soundness allows to construct *succinct* non-interactive arguments (SNARG) [39, 49], whose size can be sublinear in the size of the corresponding witness.

1.2 Our Contributions

Argument for Bilinear Group Arithmetic Relations. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ be cyclic groups of order q equipped with a pairing $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$. We propose a zero-knowledge *succinct* argument system without trusted setup for bilinear group arithmetics. A bilinear group

[1]They construct NIZK for only a special subclass of relations.

arithmetic circuit

$$C:\mathbb{Z}_q^{\ell_0}\times\mathbb{G}_1^{\ell_1}\times\mathbb{G}_2^{\ell_2}\times\mathbb{G}_t^{\ell_t}\to\mathbb{Z}_q^{n_0}\times\mathbb{G}_1^{n_1}\times\mathbb{G}_2^{n_2}\times\mathbb{G}_t^{n_t}$$

consists of fan-in 2 $\mathbb{Z}_q$ multiplication gates, exponentiation gates for sources groups ($\mathbb{G}_1$,$\mathbb{G}_2$) and target group $\mathbb{G}_t$, and pairing gates, while linear operations are "for free" (see Section 2.5). Our argument system allows one to prove succinctly and in zero-knowledge that an assignment (expressed as a vector of integers and group elements) satisfies any given bilinear group arithmetic circuit and its outputs.

As for GS proofs, a main advantage of our system with respect to generic solutions is that it can directly handle bilinear group operations without using NP-reductions. The distinguishing feature of our approach, however, is that the size of the proof is logarithmic in the size of the $(\mathbb{Z}_q,\mathbb{G}_1,\mathbb{G}_2)$ components of the witness. Unlike GS proofs, our system also supports statements whose witnesses have a $\mathbb{G}_t$ component. For those applications, our proof size is still logarithmic in the $\mathbb{Z}_q$, $\mathbb{G}_1$, and $\mathbb{G}_2$ witness components, while being linear in the dimension of the $\mathbb{G}_t$ component[2].

Our argument satisfies the strong notion of extended witness emulation [30, 45] and special honest-verifier zero-knowledge. It has a public coin verifier and can be compiled to a non-interactive publicly-verifiable argument using the Fiat-Shamir transformation [20]. The common inputs of the prover and the verifier can be sampled with public coins, which means that no trusted party is required to initialize the system. Instead the public parameters can be sampled in a verifiable way using, *e.g.*, a random oracle. The soundness of the system is shown against the *generalized discrete logarithm representation* assumption: Loosely speaking, the assumption states that given a matrix of uniform group elements[3] ($[A_0]_t$,$[A_1]_2$,$[A_2]_1$), it is hard to find a non-trivial relation ($\mathbf{a}_0$,$[\mathbf{a}_1]_1$,$[\mathbf{a}_2]_2$) such that

$$[A_0]_t\mathbf{a}_0+[A_1]_2[\mathbf{a}_1]_1+[A_2]_1[\mathbf{a}_2]_2=[\mathbf{0}]_t.$$

We show that, for a certain regime of parameters, such an assumption is implied by the symmetric external Diffie-Hellman (SXDH) assumption. We refer the reader to Section 2.2 for a precise statement.

Argument for Non-Bilinear Group Arithmetic Relations. As a special case, our technique can be applied to prove the satisfiability of non-bilinear group arithmetic circuits with only $\mathbb{Z}_q$ inputs. In this setting, our argument system does not perform any pairing operations and therefore can be instantiated with non-bilinear groups, in which operations are generally more efficient than those in blinear groups.

In more detail, let $\mathbb{G}$ be a cyclic group of order q. Our argument system allows one to prove the satisfiability of a (non-bilinear) group arithmetic circuit

$$C:\mathbb{Z}_q^{\ell_0}\to\mathbb{Z}_q^{n_0}\times\mathbb{G}^{n_t}.$$

1.3 Technical Overview

At a technical level, our system follows the general structure of existing systems for arithmetic circuit satisfiability [13, 14], and is based on the interplay of structure-preserving additively homomorphic commitments [3], linear compressions of bilinear group operations, and succinct arguments for generalized inner-product relations[4]. Specifically, our contributions can be broken down into four main components.

1.3.1 Compressing Group Arithmetic Relations: We characterize a bilinear group arithmetic circuit as a system of generalized inner product relations, and use random linear combinations to compress the system into a smaller system of just 4 generalized inner product relations. The resulting system is of the form

$$\begin{cases}\langle \mathbf{a}_L+\boldsymbol{\beta}_L,\boldsymbol{\alpha}_0\circ\mathbf{a}_R+\boldsymbol{\beta}_R\rangle=\zeta_0 & (\text{type } \mathbb{Z}_q)\\ \langle[\mathbf{a}_1]_1,\boldsymbol{\alpha}_1\circ\mathbf{a}_R+\boldsymbol{\beta}_1\rangle=[\zeta_1]_1 & (\text{type } \mathbb{G}_1)\\ \langle\boldsymbol{\alpha}_2\circ\mathbf{a}_L+\boldsymbol{\beta}_2,[\mathbf{a}_2]_2\rangle=[\zeta_2]_2 & (\text{type } \mathbb{G}_2)\\ \left\langle\mathbf{a}_L,\begin{bmatrix}\boldsymbol{\beta}_t\\ \mathbf{a}_t\end{bmatrix}_t\right\rangle+\langle\boldsymbol{\alpha}_t\circ[\mathbf{a}_1]_1,[\mathbf{a}_2]_2\rangle=[\zeta_t]_t & (\text{type } \mathbb{G}_t)\end{cases}$$

where variables written in Greek are constants (*i.e.*, the statement to be proven), while those in Latin are the unknowns (*i.e.*, the witness).

1.3.2 Argument for Group Arithmetic Satisfiability: A crucial property of the above compression technique is that the witness of the simplified system can be derived *deterministically* from the original witness of group arithmetic satisfiability, and is valid for any compressed system derived from independent randomness. With this observation, we give a brief overview of our construction of an argument system for group arithmetic satisfiability, which we will refer to as the "outer protocol" for conciseness.

In the outer protocol, the prover derives the witness of the compressed system *deterministically* from the original witness, and commits to them using structure-preserving additively homomorphic commitments. It also commits to an equal amount of masking elements, which will be used to mask the witness when the (combined) commitments are opened later.

The verifier then sends sufficient randomness to the prover, so that they can both compress the original system into a simpler system of 4 generalized inner product relations described above.

With the compressed system specified, the prover proceeds to encode the components of the generalized inner product relations and their masks into inner products between (vector-valued) polynomials, and commits to the coefficients of these polynomials.

For example, consider the "type-$\mathbb{G}_1$" relation

$$\langle[\mathbf{a}_1]_1,\boldsymbol{\alpha}_1\circ\mathbf{a}_R+\boldsymbol{\beta}_1\rangle=[\zeta_1]_1.$$

The prover encodes $[\mathbf{a}_1]_1$ and its mask into a "left-polynomial" $[\mathbf{l}(X)]_1$, and encodes $\boldsymbol{\alpha}_1\circ\mathbf{a}_R+\boldsymbol{\beta}_1$ and its mask into a "right-polynomial" $\mathbf{r}(X)$. Suppose that $[\mathbf{a}_1]_1$ is encoded as the coefficient of the monomial X in $[\mathbf{l}(X)]_1$, and $\boldsymbol{\alpha}_1\circ\mathbf{a}_R+\boldsymbol{\beta}_1$ is encoded as the coefficient of the monomial X^2 in $\mathbf{r}(X)$. Then the value $[\zeta_1]_1$ would be encoded in the coefficient of X^3 in the "product-polynomial" $[p(X)]_1=\langle[\mathbf{l}(X)_1],\mathbf{r}(X)\rangle$. We remark that the encodings presented in this paper are chosen for conceptual simplicity and clarity of presentation, and are not necessarily the most compact.

The verifier then instructs the prover to evaluate these polynomials at a random point x. With the knowledge of ($[\mathbf{l}(x)]_1$,$\mathbf{r}(x)$,$[p(x)]_1$), the verifier can check that $[p(x)]_1=\langle[\mathbf{l}(x)_1],\mathbf{r}(x)\rangle$, and $[\zeta_1]_1$ is indeed encoded in the X^3 term of $[p(X)]_1$.

In a naive instantiation of the outer protocol, the prover would have to communicate vectors such as $[\mathbf{l}(x)]_1$ and $\mathbf{r}(x)$, which are

[2]This limitation seems to be inherent to our settings due to the fact that compressing structure-preserving commitments to target group elements are, in general, impossible [4].

[3]For a fixed generator of $\mathbb{G}_i$, denoted by $[1]_i$, we denote by $[a]_i$ the element in $\mathbb{G}_i$ whose discrete logarithm base $[1]_i$ is $a\in\mathbb{Z}_q$. This notation extends naturally to matrices of group elements. We refer to Section 2.1 for details.

[4]For example, a generalized inner-product relation with $\mathbb{G}_t$ output is of the form $\langle[\mathbf{a}_t]_t,\mathbf{a}_0\rangle+\langle[\mathbf{a}_1]_1,[\mathbf{a}_2]_2\rangle=[\zeta]_t$.

of the same length as the corresponding witness components such as $[\mathbf{a}_1]_1$ and $\mathbf{a}_R$ respectively. To achieve succinct communication, the prover would instead run another protocol, not necessarily zero-knowledge, to prove that $[p(x)]_1 = \langle [\mathbf{l}(x)_1], \mathbf{r}(x) \rangle$ and the commitment of $([\mathbf{l}(x)_1], \mathbf{r}(x))$ is valid. Such a protocol is described below.

1.3.3 Generalized Inner-Product Arguments: The above steps reduce the task of proving the satisfiability of a bilinear group arithmetic circuit in zero-knowledge to that of proving the knowledge of simple inner-product relations across integers and group elements without zero-knowledge. To complete the picture, we present a family of succinct arguments (which we refer to as "inner protocols" for conciseness) to prove statements of the following form: There exists a tuple of three n-dimensional vectors $(\mathbf{a}_0, [\mathbf{a}_1]_1, [\mathbf{a}_2]_2)$ such that

$$[\mathbf{p}]_t = [A_0]_t \cdot \mathbf{a}_0 + [A_1]_2 \cdot [\mathbf{a}_1]_1 + [A_2]_1 \cdot [\mathbf{a}_2]_2 \quad (1)$$

$$[c]_t = \langle \mathbf{a}_0, [\boldsymbol{\beta}]_t \rangle + \langle [\mathbf{a}_1]_1, [\mathbf{a}_2]_2 \rangle \quad (2)$$

where (1) ensures the commitment is valid and (2) ensures the inner product relation. Furthermore $([A_0]_t, [A_1]_2, [A_2]_1, [\boldsymbol{\beta}]_t, [\mathbf{p}]_t, [c]_t)$ are public elements.

The protocol proceeds in $\lceil \log n \rceil$ rounds where the dimensions of the witnesses are progressively halved by splitting the vectors and summing the two halfs, multiplied by a random scalar chosen by the verifier. In the last step of the recursion, the $O(1)$-dimensional witness is given in plain and the above relation is recomputed by the verifier. The argument is made non-interactive by sampling the challenges through a random oracle.

Our inner protocols are inspired by a recent work that considers similar relations in the settings of integer arithmetics [14], which we generalize to support pairing-product relations. One of the conceptual differences is that we allow part of the input to the inner-product, *e.g.*, $[\boldsymbol{\beta}]_t$, to be public. This seemingly unimportant technicality is crucial for the succinctness of our proofs. Otherwise, we would need to pad the $\mathbb{G}_t$ witness component to be as long as the component in other groups, which ruins succinctness as structure-preserving commitments of $\mathbb{G}_t$ elements cannot be compressing [4].

1.3.4 Performance Evaluation: Our work in this paper focuses on achieving logarithmic proof size. For a general group arithmetic circuit C, our argument system produces (zero-knowledge) proofs with $O(\log |C|) + 3\ell_t$ elements (in $\mathbb{Z}_q$, $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_t$ combined), where $|C|$ does not count linear operations and ℓ_t is the number of secret $\mathbb{G}_t$ inputs. For GS proofs, which only support a restrict class of group arithmetic circuits which do not have $\mathbb{G}_t$-input, a (witness-indistinguishable) proof is of size $O(|C|)$. Both systems have prover and verifier time linear in the size of the circuit, *i.e.*, $O(|C|)$.

To substantiate our claims we implemented our argument system and we compared its concrete efficiency with that of an existing implementation of GS proofs for proving the following language as an example:

$$\mathcal{L}_n^* = \{([\boldsymbol{\lambda}]_1, [\zeta]_t) : \exists [\mathbf{a}]_2 \ s.t. \ \langle [\boldsymbol{\lambda}]_1, [\mathbf{a}]_2 \rangle = [\zeta]_t \}$$

This type of relations captures several important applications of arguments for group arithmetic. For example, it captures the knowledge of correct decryption for the KSW predicate encryption scheme [37][5]. More specifically, the prover wants to prove its knowledge of a secret key

$$\{[k]_2, [k_{1,i}]_2, [k_{2,i}]_2\}_{i=1}^{\ell}$$

which correctly decrypts a ciphertext

$$\left([c']_t, [c]_1, \{[c_{1,i}]_1, [c_{2,i}]_1\}_{i=1}^{\ell}\right)$$

to the message $[m]_t$, *i.e.*,

$$[c]_1[k]_2 + \sum_{i=1}^{\ell} [c_{1,i}]_1 [k_{1,i}]_2 + \sum_{i=1}^{\ell} [c_{2,i}]_1 [k_{2,i}]_2 = [m]_t - [c']_t.$$

In our argument system, a proof for a statement in $\mathcal{L}^*$ consists of $6 \log n + 6$ total number of elements, while a GS proof consists of $2n+2$ or $3n+3$ total number of elements depending on the settings.

To compare the concrete prover and verifier efficiency, we instantiate both our argument system and GS proofs in two settings: 1) a "type A1" composite-order curve in the JPBC library; and 2) a Barreto-Naehrig prime-order curve. With almost no effort given to optimizing our implementation, the prover time and verifier time of our system over the type A1 curve are only 3 and 2 times longer than those of GS proofs respectively, for sufficiently large witness length $n \geq 32$. Over the Barreto-Naehrig curve, while our prover time is quite a bit longer, our verifier time is 0.6 times shorter when $n \geq 8$. We note that these constant factors of overhead are well in range of what can be improved by existing optimization techniques (*e.g.*, [14]). Another potential route to efficiency improvement is to modify the outer protocol to use a more compact encoding of inner-product relations into coefficients of polynomials.

1.4 Applications

Our argument system offers a general purpose tool for expressive relations: Any statement that can be encoded as a bilinear group arithmetic circuit admits an efficient and short proof in our framework. Our system can be seen as a drop-in replacement for GS proofs in many applications, especially for those cases where communication efficiency and proof size are of the essence. We stress that our proof system does not require a trusted setup and it is therefore suited for many real-life scenarios. In the following we exemplify the utility of our system by highlighting a few applications of interest.

1.4.1 Structure-Preserving Signatures. One of the motivating examples of the field of structure-preserving cryptography is the design of signature schemes where one can efficiently prove the knowledge of a signature. Roughly speaking, in a structure-preserving signature (SPS) scheme, both messages and signatures consist of group elements, and signature verification can be done by checking pairing product equations.

Our argument system allows one to prove the knowledge of a tuple of n SPS signatures on different messages and under (possibly) different public keys. The size of the resulting proof grows *logarithmically* with n. This feature becomes especially useful when the signatures are stored on a public ledger, such as a blockchain, where allocating space is costly for all participants.

As an example, many blockchains implement multi-signature addresses, where parties can spend coins only if the transaction is signed from all public-keys belonging to that address. Our framework allows one to give a succinct proof for *any* SPS scheme in an efficient way.

[5]The KSW scheme is originally described in symmetric composite order pairing groups. To compare our argument and GS proofs for proving relations about KSW, we can either instantiate both systems in such groups, or first transform the KSW scheme to the prime-order setting [5], and instantiate the systems in prime-order groups.

Due to the generality of our approach, our framework is also applicable to a variety of SPS schemes with extra features, such as linearly homomorphic signatures [43] and signatures on equivalence classes [35].

1.4.2 Attribute-Based Encryption. Attribute-Based Encryption (ABE) [26, 52] and Predicate Encryption (PE) [37] allow a manager to distribute keys for certain predicates. Everyone can encrypt a message with respect to a set of attributes. Decryption is possible if and only if the attributes encoded in the ciphertext satisfy the policy encoded in the key. PE offers the additional guarantee that the attributes are hidden.

ABE and PE have been the subject of a large body of research and most of the efficient schemes are based on bilinear maps. These schemes are natural candidates for composing with our argument system: We can, *e.g.*, prove well-formedness of ciphertexts computed with respect to large sets of (possibly secret) attributes, or the knowledge of a key that correctly decrypts a certain ciphertext. The proof scales logarithmically with the statement (*e.g.*, the number of attributes) being certified.

1.4.3 Attribute-Based Authentication. As an application of independent interest, our system allows us to efficiently prove the knowledge of attribute-based signatures (ABS) [46] based on bilinear maps. ABS enriches the syntax of standard digital signatures, allowing a signer, with a certified set of attributes, to sign messages with predicates that are satisfied by its attributes. ABS are useful in many contexts, such as attribute-based authentication and disclosure of secrets. Our argument can be used as a cheaper alternative to GS proofs to prove the knowledge of signatures for complex access structures.

1.5 Related Work

The past decade has seen an impressive development of succinct arguments for general NP relations. Different approaches achieve different trade-offs. We stress that although these argument systems support proving general NP relations, none of them can handle (bilinear) group arithmetic relations *natively* without relying on expensive reductions to the supported NP complete language (e.g., circuit satisfiability).

Direct Approach. Groth [28] gave a construction of a succinct argument for proving boolean circuits based on the DPAIR-assumption. Bootle *et al.* [13] and Bulletproof [14] express the satisfiability of an arithmetic circuit into a system of inner-product relations, and construct a succinct argument by recursively proving inner-product relations in the discrete-logarithm setting. The resulting proofs are of size logarithmic in the size of the verification circuit. Hyrax [55] follows a similar paradigm except that it achieves a milder form of succinctness: The size of the proof depends linearly on the depth of the verification circuit and logarithmically on its width. Ligero [6] builds sublinear proofs from multi-party computation protocols.

PCP/IOP-Based Approach. Pioneered by Kilian [39] and Micali [48, 49], a series of work (*e.g.*, [6, 8, 9]) focuses on building succinct arguments by combining an information-theoretic proof system, such as probabilistically checkable proofs (PCP) or in general interactive oracle proofs (IOP) [10], with a cryptographic compiler, such as a Merkle-tree [47] or in general a (sub)vector commitment [40].

Linear PCP-Based Approach. A beautiful line of work (*e.g.*, [23, 29]) builds SNARGs from bilinear pairing and linear PCPs. These schemes typically feature very short proofs and very efficient verifier, but has the intrinsic drawback of requiring a trusted setup.

2 PRELIMINARIES

Let $\lambda \in \mathbb{N}$ denote the security parameter. Let $\mathrm{poly}(\lambda)$ and $\mathrm{negl}(\lambda)$ denote the set of polynomials and negligible functions in λ respectively. For a positive integer n, $[n] := \{1,2,\ldots,n\}$.

2.1 Additive Notation for Group Operations

Let $\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_t$ be cyclic groups of order q equipped with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$. For each $i \in \{1,2\}$, we fix a generator of $\mathbb{G}_i$ and denote it by $[1]_i$. The element $[1]_t := [1]_1 \cdot [1]_2$ is then a generator of $\mathbb{G}_t$, where the operation $\cdot$ is defined below. For all $i \in \{1,2,t\}$ the identity element in $\mathbb{G}_i$ is denoted by $[0]_i$. The notation $[a]_i$ denotes a group element in $\mathbb{G}_i$ with discrete logarithm $a \in \mathbb{Z}_q$ with respect to $[1]_i$. Group operations are written additively, *i.e.*, $[a]_i + [b]_i := [a+b]_i$. Given an exponent $x \in \mathbb{Z}_q$, and a group element $[a]_i \in \mathbb{G}_i$, the exponentiation of $[a]_i$ to the power x is written as $[ax]_i := x \cdot [a]_i = [a]_i \cdot x$. Given $[a]_1 \in \mathbb{G}_1$ and $[b]_2 \in \mathbb{G}_2$, the pairing between them is written as $[ab]_t := [a]_1 \cdot [b]_2 = [b]_2 \cdot [a]_1$. The notation is extended naturally to vectors (denoted by bold symbols) and matrices (denoted by uppercase letters) of group elements, and the matrix-vector products, inner products, and Hadamard products (denoted by $\circ$) between them. For instance, if $A \in \mathbb{Z}_q^{m\times n}$ and $B \in \mathbb{Z}_q^{n\times k}$, then $[A]_1 \in \mathbb{G}_1^{m\times n}$ and $[B]_2 \in \mathbb{G}_2^{n\times k}$. Furthermore, $[A]_1[B]_2 := [AB]_t$.

2.2 Hardness Assumptions

We state a generalized version of all the assumptions which we will rely on throughout this work.

Definition 2.1 (Generalized Discrete Logarithm Representation Assumption). Let $m \geq 1$ and $n_0, n_1, n_2 \geq 0$ be not all zero. The (m,n_0,n_1,n_2)-GDLR assumption is said to hold in the bilinear groups $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_t,q,e)$ if for any PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} [B_0]_t\mathbf{a}_0 + [B_1]_2[\mathbf{a}_1]_1 + [B_2]_1[\mathbf{a}_2]_2 = [\mathbf{0}]_t \\ \wedge\, (\mathbf{a}_0,\mathbf{a}_1,\mathbf{a}_2) \neq (\mathbf{0},\mathbf{0},\mathbf{0}) : \\ B_i \leftarrow_{\$} \mathbb{Z}_q^{m\times n_i},\ \forall i \in \{0,1,2\} \\ (\mathbf{a}_0,[\mathbf{a}_1]_1,[\mathbf{a}_2]_2) \leftarrow \mathcal{A}\left(\begin{array}{c}\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_t,q,e, \\ [B_0]_t,[B_1]_2,[B_2]_1\end{array}\right)\end{array}\right] \leq \mathrm{negl}(\lambda).$$

The following lemma shows that without loss of generality we can assume that $n_0 = 0$. Due to space constraints, we defer the proof to Appendix A.1.

Lemma 2.2. *Let q be such that $1/q = \mathrm{negl}(\lambda)$. If the $(m,0,n_0+n_1,n_2)$-GDLR assumption holds, so does the (m,n_0,n_1,n_2)-GDLR assumption.*

We next discuss the case $n_1 = 0$ or $n_2 = 0$. The $(m,0,n_1,0)$-GDLR assumption is implied by the double-pairing (DBP) assumption over $\mathbb{G}_1$ for any $m \geq 1$ and any $n_1 \geq 2$. Similarly, $(m,0,0,n_2)$-GDLR assumption is implied by the DBP assumption over $\mathbb{G}_2$ (also called the "reverse DBP assumption") for any $m \geq 1$ and any $n_2 \geq 2$ not all zero.

The DBP assumption over $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) is implied by the decisional Diffie-Hellman (DDH) assumption over $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) (see, *e.g.*, [3]).

Next, we study the case $n_1 > 0$ and $n_2 > 0$. If $\mathbb{G}_1 = \mathbb{G}_2$ (so called "type-I pairing") or if there exists an efficient homomorphism $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$ (so called "type-II pairing"), then for any $m \geq 2$ and any $n_1 + n_2 \geq 2$, the $(m,0,n_1,n_2)$-GDLR assumption is implied by the simultaneous double pairing (SDP) assumption over $\mathbb{G}_1$, which is in turn implied by the decisional linear assumption (DLin) over $\mathbb{G}_1$ (see, *e.g.*, [17]).

We are not aware of any work that explicitly treats the general (m, n_0, n_1, n_2)-GDLR assumption in the case where no efficient homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ is known (type-III pairing). Indeed, most pairing based assumptions consider problems defined by elements in one source group (say $\mathbb{G}_1$) and / or the target group $\mathbb{G}_t$, while the solution consists of elements in the other source group $\mathbb{G}_2$. In the following, we show that the (m,n_0,n_1,n_2)-GDLR assumption is implied by the symmetric external Diffie-Hellman (SXDH) assumption, which states that the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$. For the proof we refer to Appendix A.2.

Theorem 2.3. *Let q be such that $1/q = \mathrm{negl}(\lambda)$. Let $m \geq 2$ and $n_0,n_1,n_2 \geq 0$ be not all zero. The (m,n_0,n_1,n_2)-GDLR assumption holds if the SXDH assumption holds.*

2.3 Commitment Schemes

Our constructions make use of various additively homomorphic commitments to elements in different domains. In some cases we require a commitment scheme to be both hiding and binding, while in some other cases we require only binding. All of the schemes that we consider in this work require using randomly sampled group elements as the public parameters of the scheme (henceforth referred to as the "basis" of the commitment). We stress that such a procedure can be done with public coins, e.g., using a random oracle, resulting in schemes without trusted setup.

2.3.1 Committing to $\mathbb{Z}_q$ elements. We use the Pedersen commitment scheme [50] for $\mathbb{Z}_q$ elements. Given the public parameters $\mathsf{pp} = ([g]_1,[\mathbf{b}^T]_1) \in \mathbb{G}_1 \times \mathbb{G}_1^\ell$ for some $\ell \in \mathbb{N}$, and some randomness $r \in \mathbb{Z}_q$, the commitment to the vector $\mathbf{a} \in \mathbb{Z}_q^\ell$ is computed as

$$\mathsf{Com}^{(0)}_{\mathsf{pp}}(\mathbf{a};r) := [g]_1 r + [\mathbf{b}^T]_1 \mathbf{a}.$$

Clearly, the Pedersen commitment is additively homomorphic in the sense that

$$\mathsf{Com}^{(0)}_{\mathsf{pp}}(\mathbf{a};r) + \mathsf{Com}^{(0)}_{\mathsf{pp}}(\mathbf{a}';r') = \mathsf{Com}^{(0)}_{\mathsf{pp}}(\mathbf{a}+\mathbf{a}';r+r').$$

If the randomness r is chosen uniformly from $\mathbb{Z}_q$, then $\mathsf{Com}^{(0)}_{\mathsf{pp}}(\mathbf{a};r)$ perfectly hides $\mathbf{a}$. Lastly, if pp is sampled uniformly, then $\mathsf{Com}^{(0)}_{\mathsf{pp}}$ is computationally binding under discrete logarithm assumption over $\mathbb{G}_1$.

2.3.2 Committing to $\mathbb{G}_1$ or $\mathbb{G}_2$ elements. We use a variant of the scheme of Abe *et al.* [3] for committing to $\mathbb{G}_1$ or $\mathbb{G}_2$ elements (but not both). We describe below the scheme for $\mathbb{G}_1$. The scheme for $\mathbb{G}_2$ is analogous and is omitted.

Given the public parameters $\mathsf{pp} = ([g]_t,[\mathbf{b}^T]_2) \in \mathbb{G}_t \times \mathbb{G}_2^\ell$ for some $\ell \in \mathbb{N}$, and some randomness $r \in \mathbb{Z}_q$, the commitment to the vector $[\mathbf{a}]_1 \in \mathbb{G}_1^\ell$ is computed as

$$\mathsf{Com}^{(1)}_{\mathsf{pp}}([\mathbf{a}]_1;r) := [g]_t r + [\mathbf{b}^T]_2 [\mathbf{a}]_1.$$

Clearly, the commitment scheme is additively homomorphic in the sense that

$$\mathsf{Com}^{(1)}_{\mathsf{pp}}([\mathbf{a}]_1;r) + \mathsf{Com}^{(1)}_{\mathsf{pp}}([\mathbf{a}']_1;r') = \mathsf{Com}^{(1)}_{\mathsf{pp}}([\mathbf{a}]_1 + [\mathbf{a}']_1;r+r').$$

If the randomness r is chosen uniformly from $\mathbb{Z}_q$, then $\mathsf{Com}^{(1)}_{\mathsf{pp}}([\mathbf{a}]_1;r)$ perfectly hides $[\mathbf{a}]_1$. Lastly, if the public parameters pp are sampled uniformly, then $\mathsf{Com}^{(1)}_{\mathsf{pp}}$ is computationally binding under the $(1,1,\ell,0)$-GDLR assumption.

2.3.3 Committing to $\mathbb{G}_t$ Elements. We use a "key-less" variant of the ElGamal encryption scheme [19] as a commitment scheme for $\mathbb{G}_t$ elements. Given the public parameters $([g]_t,[\mathbf{b}]_t) \in \mathbb{G}_t \times \mathbb{G}_t^\ell$ for some $\ell \in \mathbb{N}$, and some randomness $r \in \mathbb{Z}_q$, the commitment to the vector $[\mathbf{a}]_t \in \mathbb{G}_t^\ell$ is computed as

$$\mathsf{Com}^{(t)}_{\mathsf{pp}}([\mathbf{a}]_t;r) := \begin{pmatrix} [g]_t r \\ [\mathbf{b}]_t r + [\mathbf{a}]_t \end{pmatrix}.$$

This commitment scheme is also additively homomorphic in the sense that

$$\mathsf{Com}^{(t)}_{\mathsf{pp}}([\mathbf{a}]_t;r) + \mathsf{Com}^{(t)}_{\mathsf{pp}}([\mathbf{a}']_t;r') = \mathsf{Com}^{(t)}_{\mathsf{pp}}([\mathbf{a}]_t + [\mathbf{a}']_t;r+r').$$

The ElGamal commitment is perfectly binding. If the public parameters are chosen uniformly, then $\mathsf{Com}^{(t)}_{\mathsf{pp}}$ is computationally hiding under the DDH assumption over $\mathbb{G}_t$.

Unlike the previous commitment schemes where a commitment to a length-ℓ vector consists of a single group element, a commitment to a vector of ℓ $\mathbb{G}_t$ elements consists of $\ell+1$ $\mathbb{G}_t$ elements. The linear dependency of the vector length is however close to optimal: Abe, Haralambiev, and Ohkubo [4] have shown that a structure-preserving commitment to an ℓ-dimensional vector of $\mathbb{G}_t$ elements has size $\Omega(\ell)$.

"Basis" of Commitments. We call the vector $\mathbf{b}$ in the public parameters the "basis" for the commitment. To emphasize the basis $\mathbf{b}$ being used, we sometimes write it instead of pp in the subscript, *i.e.*,

$$\mathsf{Com}^{(t)}_{\mathbf{b}}([\mathbf{a}]_t;r).$$

Note that $\mathbf{b}$ is written without the bracket $[\cdot]_t$ just for clarity. The knowledge of $[\mathbf{b}]_t$ suffices to compute a commitment.

2.3.4 Committing to $\mathbb{Z}_q$, $\mathbb{G}_1$, and $\mathbb{G}_2$ Elements Simultaneously. Our constructions require an (additively) homomorphic commitment scheme which allows to commit to $\mathbb{Z}_q$, $\mathbb{G}_1$, and $\mathbb{G}_2$ elements simultaneously. For this purpose, we introduce the following commitment scheme, which is essentially a combination of the commitment schemes by Pedersen [50] (for $\mathbb{Z}_q$ elements) and Abe *et al.* [3] (for $\mathbb{G}_1$ and $\mathbb{G}_2$ elements).

The public parameters are of the form $\mathsf{pp} = ([\mathbf{b}]_t,[B_0]_t,[B_1]_2,[B_2]_1)$ where $\mathbf{b} \in \mathbb{Z}_q^2$, and $B_i \in \mathbb{Z}_q^{2\times\ell_i}$ for some $\ell_i \in \mathbb{N}$ for all $i \in \{0,1,2\}$. Given the public parameters pp, and some randomness $r \in \mathbb{Z}_q$, the commitment to the vectors $\mathbf{a}_0 \in \mathbb{Z}_q^{\ell_0}$, $[\mathbf{a}_1]_1 \in \mathbb{G}_1^{\ell_1}$, and $[\mathbf{a}_2]_2 \in \mathbb{G}_2^{\ell_2}$ is computed as

$$\begin{aligned}&\mathsf{Com}^{(\mathrm{mix})}_{\mathsf{pp}}(\mathbf{a}_0,[\mathbf{a}_1]_1,[\mathbf{a}_2]_2;r)\\ &:= [\mathbf{b}]_t \cdot r + [B_0]_t \mathbf{a}_0 + [B_1]_2[\mathbf{a}_1]_1 + [B_2]_1[\mathbf{a}_2]_2.\end{aligned}$$

This commitment is again additively homomorphic in the sense that

$$\begin{aligned}&\mathsf{Com}^{(\mathrm{mix})}_{\mathsf{pp}}(\mathbf{a}_0,[\mathbf{a}_1]_1,[\mathbf{a}_2]_2;r) + \mathsf{Com}^{(\mathrm{mix})}_{\mathsf{pp}}(\mathbf{a}'_0,[\mathbf{a}'_1]_1,[\mathbf{a}'_2]_2;r')\\ &= \mathsf{Com}^{(\mathrm{mix})}_{\mathsf{pp}}(\mathbf{a}_0+\mathbf{a}'_0,[\mathbf{a}_1]_1+[\mathbf{a}'_1]_1,[\mathbf{a}_2]_2+[\mathbf{a}'_2]_2;r+r').\end{aligned}$$

Assuming that pp are sampled uniformly, the commitment scheme is computationally hiding under the DDH assumption over

$\mathbb{G}_t$, and computationally binding under the $(2,\ell_0+1,\ell_1,\ell_2)$-GDLR assumption.

"Basis" of the Commitment. To emphasize the "basis" B_0, B_1, and B_2 used for the commitment, we sometimes write them instead of pp in the subscript, *i.e.*,

$$\mathsf{Com}^{(\mathrm{mix})}_{B_0,B_1,B_2}(\mathbf{a}_0,[\mathbf{a}_1]_1,[\mathbf{a}_2]_2;r).$$

Note that B_0, B_1 and B_2 are written without the brackets $[\cdot]_t$, $[\cdot]_2$ and $[\cdot]_1$ respectively just for clarity. The knowledge of $[B_0]_t$, $[B_1]_2$, and $[B_2]_1$ suffices to compute a commitment.

We state a simple fact about the relation between Hadamard products and matrix products, which will be useful for "changing the bases" of commitments.

Fact 1. *Let $B \in \mathbb{Z}_q^{m\times n}$ and $\boldsymbol{\alpha}, \mathbf{a} \in \mathbb{Z}_q^n$. Then $B(\boldsymbol{\alpha}\circ\mathbf{a}) = \left(B\circ(\mathbf{1}^m\boldsymbol{\alpha}^T)\right)\mathbf{a}$.*

Fact 2 (Changing Bases). *Extending Fact 1, we have*

$$\mathsf{Com}^{(\mathrm{mix})}_{B_0,B_1,B_2}(\boldsymbol{\alpha}_0\circ\mathbf{a}_0,\boldsymbol{\alpha}_1\circ[\mathbf{a}_1]_1,\boldsymbol{\alpha}_2\circ[\mathbf{a}_2]_2;r)$$
$$=\mathsf{Com}^{(\mathrm{mix})}_{B_0\circ(\mathbf{1}^2\boldsymbol{\alpha}_0^T),B_1\circ(\mathbf{1}^2\boldsymbol{\alpha}_1^T),B_2\circ(\mathbf{1}^2\boldsymbol{\alpha}_2^T)}(\mathbf{a}_0,[\mathbf{a}_1]_1,[\mathbf{a}_2]_2;r).$$

2.4 Arguments of Knowledge

In the following we give a formal characterization of argument systems and the corresponding properties. All the definitions are taken (almost) in verbatim from [14].

Definition 2.4 (Arguments). An argument system for a relation $\mathcal{R}$ is a triple of PPT algorithms $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ with the following syntax. On input 1^λ the setup algorithm Setup produces a common reference string crs. The prover $\mathcal{P}$ interacts with the verifier $\mathcal{V}$ to produce a transcript $\mathsf{tr}=\langle\mathcal{P}(\mathsf{crs},\mathsf{stmt},\mathsf{wit}),\mathcal{V}(\mathsf{crs},\mathsf{stmt})\rangle$, where $\langle.\rangle$ denotes the interaction between $\mathcal{P}$ and $\mathcal{V}$. After such interaction, $\mathcal{V}$ should be able to decide whether $(\mathsf{crs},\mathsf{stmt},\mathsf{wit})\in\mathcal{R}$. In this case, we say that tr is accepting.

Definition 2.5 (Perfect completeness). An argument system $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ has *perfect completeness* if for all non-uniform polynomial time algorithms $\mathcal{A}$

$$\Pr\left[\begin{array}{c|c}(\mathsf{crs},\mathsf{stmt},\mathsf{wit})\notin\mathcal{R}\vee & \mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda)\\ \mathsf{tr}\text{ is accepting} & (\mathsf{stmt},\mathsf{wit})\leftarrow\mathcal{A}(\mathsf{crs})\\ & \mathsf{tr}\leftarrow\langle\mathcal{P}(\mathsf{crs},\mathsf{stmt},\mathsf{wit}),\\ & \mathcal{V}(\mathsf{crs},\mathsf{stmt})\rangle\end{array}\right]=1.$$

Definition 2.6 (Computational Witness-Extended Emulation). An argument system $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ has *witness-extended emulation* if for all deterministic polynomial time $\mathcal{P}^*$ there exists an expected polynomial time emulator $\mathcal{E}$ such that for all pairs of adversaries $\mathcal{A}_1$, $\mathcal{A}_2$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\left|\begin{array}{l}\Pr\left[\begin{array}{c|c}\mathcal{A}_1(\mathsf{tr})=1 & \mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda),\\ & (\mathsf{stmt},\mathsf{wit})\leftarrow\mathcal{A}_2(\mathsf{crs}),\\ & \mathsf{tr}\leftarrow\langle\mathcal{P}^*(\mathsf{crs},\mathsf{stmt},\mathsf{wit}),\\ & \mathcal{V}(\mathsf{crs},\mathsf{stmt})\rangle\end{array}\right]-\\ \Pr\left[\begin{array}{c|c}\mathcal{A}_1(\mathsf{tr})=1\wedge & \mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda),\\ (\mathsf{tr}\text{ is accepting} & (\mathsf{stmt},\mathsf{wit})\leftarrow\mathcal{A}_2(\mathsf{crs}),\\ \Rightarrow(\mathsf{crs},\mathsf{stmt},\mathsf{wit}')\in\mathcal{R}) & (\mathsf{tr},\mathsf{wit}')\leftarrow\mathcal{E}^{\mathcal{O}}(\mathsf{crs},\mathsf{stmt})\end{array}\right]\end{array}\right|\le\mathsf{negl}(\lambda)$$

where the oracle is given by $\mathcal{O}=\langle\mathcal{P}^*(\mathsf{crs},\mathsf{stmt},\mathsf{wit}),\mathcal{V}(\mathsf{crs},\mathsf{stmt})\rangle$, and permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards. If the adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ are restricted to run in polynomial time, then we say $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ has *computational witness-extended emulation.*

Definition 2.7 (Public coin). An argument system $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ is called *public coin* if the Setup algorithm is executed using public randomness and all messages sent from the verifier to the prover are chosen uniformly at random and independently of the prover's messages, *i.e.*, the challenges correspond to the verifier's randomness ρ.

Definition 2.8 (Computational Special Honest-Verifier Zero-Knowledge). A public-coin argument system $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ is *computationally special honest-verifier zero knowledge (SHVZK)* for $\mathcal{R}$ if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all PPT adversaries $\mathcal{A}_1$ and non-uniform polynomial time algorithms $\mathcal{A}_2$

$$\left|\begin{array}{l}\Pr\left[\begin{array}{c|c} & \mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda),\\ (\mathsf{crs},\mathsf{stmt},\mathsf{wit})\in\mathcal{R}\wedge & (\mathsf{stmt},\mathsf{wit},\rho)\leftarrow\mathcal{A}_2(\mathsf{crs}),\\ \mathcal{A}_1(\mathsf{tr})=1 & \mathsf{tr}\leftarrow\langle\mathcal{P}(\mathsf{crs},\mathsf{stmt},\mathsf{wit}),\\ & \mathcal{V}(\mathsf{crs},\mathsf{stmt};\rho)\rangle\end{array}\right]\\ -\Pr\left[\begin{array}{c|c}(\mathsf{crs},\mathsf{stmt},\mathsf{wit})\in\mathcal{R}\wedge & \mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda),\\ \mathcal{A}_1(\mathsf{tr})=1 & (\mathsf{stmt},\mathsf{wit},\rho)\leftarrow\mathcal{A}_2(\mathsf{crs}),\\ & \mathsf{tr}\leftarrow\mathcal{S}(\mathsf{stmt},\rho)\end{array}\right]\end{array}\right|\le\mathsf{negl}(\lambda).$$

2.5 Encoding Group Arithmetic Circuits

Consider a circuit

$$C:\mathbb{Z}_q^{\ell_0}\times\mathbb{G}_1^{\ell_1}\times\mathbb{G}_2^{\ell_2}\times\mathbb{G}_t^{\ell_t}\to\mathbb{Z}_q^{n_0}\times\mathbb{G}_1^{n_1}\times\mathbb{G}_2^{n_2}\times\mathbb{G}_t^{n_t}$$

which consists of fan-in 2 $\mathbb{Z}_q$ multiplication gates, $\mathbb{G}_i$ exponentiation gates for $i\in\{1,2,t\}$, and pairing gates. Linear operations in $\mathbb{Z}_q$, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ respectively are performed "for free" in the sense that an input to a multiplication, exponentiation, or pairing gate can be a linear combination of the outputs from other gates of the compatible type. We call such a circuit a (bilinear) group arithmetic circuit.

Let C be a group arithmetic circuit with m_0 $\mathbb{Z}_q$ multiplication gates, m_i $\mathbb{G}_i$ exponentiation gates for $i\in\{1,2,t\}$, and m_{12} pairing gates. The satisfiability of C for a given output is equivalent to the existence of a solution of a system of equations of the following form, where unknowns are written in Latin and constants in Greek:

$$\forall i\in[m_0],\qquad \left\langle\mathbf{a}_L,\boldsymbol{\alpha}_{0,i}\circ\mathbf{a}_R\right\rangle=0 \tag{3}$$
$$\forall i\in[m_1],\qquad \left\langle[\mathbf{a}_1]_1,\boldsymbol{\alpha}_{1,i}\circ\mathbf{a}_R\right\rangle=[0]_1 \tag{4}$$
$$\forall i\in[m_2],\qquad \left\langle\boldsymbol{\alpha}_{2,i}\circ\mathbf{a}_L,[\mathbf{a}_2]_2\right\rangle=[0]_2 \tag{5}$$
$$\forall i\in[n_t],\qquad \left\langle\mathbf{a}_{E,i},\begin{bmatrix}\boldsymbol{\beta}_t\\ \mathbf{a}_t\end{bmatrix}_t\right\rangle+\left\langle\boldsymbol{\alpha}_{t,i}\circ[\mathbf{a}_1]_1,[\mathbf{a}_2]_2\right\rangle=[\zeta_{t,i}]_t \tag{6}$$
$$\forall i\in[q_0],\qquad \left\langle\mathbf{a}_L,\boldsymbol{\beta}_{R,i}\right\rangle+\left\langle\boldsymbol{\beta}_{L,i},\mathbf{a}_R\right\rangle+\sum_{j=1}^{n_t}\left\langle\mathbf{a}_{E,j},\boldsymbol{\beta}_{E,i,j}\right\rangle=\zeta_{0,i} \tag{7}$$
$$\forall i\in[q_1],\qquad \left\langle[\mathbf{a}_1]_1,\boldsymbol{\beta}_{1,i}\right\rangle=[\zeta_{1,i}]_1 \tag{8}$$
$$\forall i\in[q_2],\qquad \left\langle\boldsymbol{\beta}_{2,i},[\mathbf{a}_2]_2\right\rangle=[\zeta_{2,i}]_2 \tag{9}$$

In the above, for each $i\in[m_0]$, Equation (3) encodes the input-output relation of the i-th $\mathbb{Z}_q$ multiplication gate as follows. The vector $\mathbf{a}_L$ (resp. $\mathbf{a}_R$) consists of, among other values, the concatenation of all left-inputs (resp. right-inputs) to all $\mathbb{Z}_q$ multiplication gates. $\mathbf{a}_R$ also consists of the concatenation of all outputs of all $\mathbb{Z}_q$ multiplication gates. The public vector $\boldsymbol{\alpha}_{0,i}$ consists of mostly zeros, except for the positions corresponding to the right-input and output of the

i-th $\mathbb{Z}_q$ multiplication gate, which are set to 1 and -1 respectively. For example, suppose that the circuit C specifies that the i-th Z_q multiplication gate multiplies the k_1-th entries of $\mathbf{a}_L$ and $\mathbf{a}_R$ to get the k_2-th entry of $\mathbf{a}_R$. Then $\mathbf{a}_L$ would have the k_2-th entry set to 1 (which will be enforced by Equation (7)), and $\boldsymbol{\alpha}_{0,i}$ is a vector with the k_1-th entry being 1, the k_2-th entry being -1, and zero everywhere else.

Similarly, Equations (4) and (5) encode the input-output relations of $\mathbb{G}_1$ and $\mathbb{G}_2$ exponentiation gates respectively.

The treatment for $\mathbb{G}_t$ relations is somewhat different, as the prover of the satisfiability of C eventually needs to commit to the $\mathbb{G}_t$ unknowns, and the commitment string has to be as long as the number of $\mathbb{G}_t$ unknowns. The encoding is therefore designed to minimize the number of $\mathbb{G}_t$ unknowns in the relations. Concretely, for $i \in [n_t]$ (one per $\mathbb{G}_t$ output), Equation (6) encodes the relation between the i-th $\mathbb{G}_t$ output of C, the $\mathbb{G}_t$ inputs to C, and other values which are either public or can be committed to succinctly. The vector $\mathbf{a}_{E,i}$ consists of the concatenation of all $\mathbb{Z}_q$-inputs to all $\mathbb{G}_t$ exponentiation gates contributing to the i-th $\mathbb{G}_t$-output of C. The vector $[\boldsymbol{\beta}_t]_t$ consists of the concatenation of all $\mathbb{G}_t$-inputs to all $\mathbb{G}_t$ exponentiation gates with constant $\mathbb{G}_t$-input. The vector $[\mathbf{a}_t]_t$ consists of the concatenation of all $\mathbb{G}_t$-inputs to all $\mathbb{G}_t$ exponentiation gates with variable $\mathbb{G}_t$-input. The vectors $[\mathbf{a}_1]_1$ and $[\mathbf{a}_2]_2$ consist of, among other values, all the $\mathbb{G}_1$ and $\mathbb{G}_2$ inputs to the pairing gates. The vector $\boldsymbol{\alpha}_{t,i}$ selects which elements of $[\mathbf{a}_1]_1$ and $[\mathbf{a}_2]_2$ should be paired. Finally, the value $[\zeta_{t,i}]_t$ denotes the i-th $\mathbb{G}_t$-output of C.

For each $i \in [q_0]$ where $q_0 \le 4m_0 + 2m_1 + 2m_2 + 2m_{12} + m_t n_t$, Equation (7) encodes the i-th linear relation between the $\mathbb{Z}_q$ unknowns. Similarly, Equation (8) encode the linear relations between the $\mathbb{G}_1$ and $\mathbb{G}_2$ elements respectively, where $q_1, q_2 \le 2m_0 + m_1 + m_2 + m_{12}$.

Note that all vectors are padded so that they have the appropriate lengths for the inner products.

2.6 Encoding Compression

The goal of the paper is to design an argument system for the satisfiability of group arithmetic circuits. One way of doing so is to design a system where the prover convinces the verifier that a system of equations of the form defined above can be satisfied. However, the form of the system is unwieldy. In the following, we recall standard techniques of compressing the systems into much smaller ones.

2.6.1 Compressing Relations of the Same Type of Gates. Using well-known random linear combination techniques[6], to convince a verifier that, say, Equation (3) holds for all $i \in [m_0]$, it suffices for the prover to show that the relation obtained by a random linear combination of the m_0 equations holds, where the randomness used for the linear combination is chosen by the verifier. Applying the technique to all equations, we obtain a system of the following form:

$$\langle \mathbf{a}_L, \boldsymbol{\alpha}_0 \circ \mathbf{a}_R \rangle = 0 \tag{10}$$

$$\langle [\mathbf{a}_1]_1, \boldsymbol{\alpha}_1 \circ \mathbf{a}_R \rangle = [0]_1 \tag{11}$$

$$\langle \boldsymbol{\alpha}_2 \circ \mathbf{a}_L, [\mathbf{a}_2]_2 \rangle = [0]_2 \tag{12}$$

$$\left\langle \mathbf{a}_E, \begin{bmatrix} \boldsymbol{\beta}_t \\ \mathbf{a}_t \end{bmatrix}_t \right\rangle + \langle \boldsymbol{\alpha}_t \circ [\mathbf{a}_1]_1, [\mathbf{a}_2]_2 \rangle = [\zeta_t]_t \tag{13}$$

$$\langle \mathbf{a}_L, \boldsymbol{\beta}_R \rangle + \langle \boldsymbol{\beta}_L, \mathbf{a}_R \rangle + \langle \mathbf{a}_E, \boldsymbol{\beta}_E \rangle + \sum_{j=1}^{n_t} \langle \mathbf{a}_{E,j}, \boldsymbol{\beta}_{E,j} \rangle = \zeta_0 \tag{14}$$

$$\langle [\mathbf{a}_1]_1, \boldsymbol{\beta}_1 \rangle = [\zeta_1]_1 \tag{15}$$

$$\langle \boldsymbol{\beta}_2, [\mathbf{a}_2]_2 \rangle = [\zeta_2]_2 \tag{16}$$

In the above, $\boldsymbol{\alpha}_0$ is the result of a random linear combination of $\{\boldsymbol{\alpha}_{0,1}, \ldots, \boldsymbol{\alpha}_{0,m_0}\}$. The vectors $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\beta}_1$, and $\boldsymbol{\beta}_2$ are defined analogously. The way of obtaining $\mathbf{a}_E$, the $\boldsymbol{\beta}$'s, and $\boldsymbol{\alpha}_t$ is slightly more complicated. First, a random linear combination of Equation (6) is performed over $\{\mathbf{a}_{E,i}\}_i$ and $\{\boldsymbol{\alpha}_{t,i}\}_i$ to obtain $\mathbf{a}_E$ and $\boldsymbol{\alpha}_t$ respectively. Note that $\mathbf{a}_E$ is a new "dummy" unknown $\mathbb{Z}_q$ vector, whose integrity must be guaranteed by inducing additional (linear) relations. These relations can be written in a form similar to that of Equation (7), with the new unknown $\mathbf{a}_E$. The extended Equation (7) is then combined using a random linear combination to obtain Equation (14).

2.6.2 Further Compressing Relations over the Same Group. To further simplify the system of relations that the prover has to prove, the prover and the verifier can compress the above system again by performing random linear combination over equations in the same group. By doing so, we end up with a system of 4 equations: 1 over each of $\mathbb{Z}_q$, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_t$. In addition to this, we append the vector $\mathbf{a}_E$ to $\mathbf{a}_L$ (which results in a new and longer $\mathbf{a}_L$), separate $\{\mathbf{a}_{E,i}\}_i$ into two parts, and append the parts to $\mathbf{a}_L$ and $\mathbf{a}_R$ respectively so that $|\mathbf{a}_L| = |\mathbf{a}_R|$. The constant vectors encoding the constraints are also appended accordingly. The system is of the following form:

$$\begin{cases} \langle \mathbf{a}_L + \boldsymbol{\beta}_L, \boldsymbol{\alpha}_0 \circ \mathbf{a}_R + \boldsymbol{\beta}_R \rangle = \zeta_0 & (\text{type } \mathbb{Z}_q) \\ \langle [\mathbf{a}_1]_1, \boldsymbol{\alpha}_1 \circ \mathbf{a}_R + \boldsymbol{\beta}_1 \rangle = [\zeta_1]_1 & (\text{type } \mathbb{G}_1) \\ \langle \boldsymbol{\alpha}_2 \circ \mathbf{a}_L + \boldsymbol{\beta}_2, [\mathbf{a}_2]_2 \rangle = [\zeta_2]_2 & (\text{type } \mathbb{G}_2) \\ \left\langle \mathbf{a}_L, \begin{bmatrix} \boldsymbol{\beta}_t \\ \mathbf{a}_t \end{bmatrix}_t \right\rangle + \langle \boldsymbol{\alpha}_t \circ [\mathbf{a}_1]_1, [\mathbf{a}_2]_2 \rangle = [\zeta_t]_t & (\text{type } \mathbb{G}_t) \end{cases}$$

Taking all transformations into account, the length of the witness components are (overestimatedly) $|\mathbf{a}_L| = |\mathbf{a}_R| = |[\mathbf{a}_1]_1| = |[\mathbf{a}_2]_2| \le 2m_0 + m_1 + m_2 + m_{12} + m_t n_t$ and $|\mathbf{a}_t| = \ell_t$.

2.6.3 A Formal Description. Putting everything together, suppose that a prover wishes to prove its knowledge about the input $(\mathbf{x}_0, [\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{x}_t]_t)$ such that $C(\mathbf{x}_0, [\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{x}_t]_t) = (\mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t)$. A crucial observation is that, throughout the compression process, the satisfying assignment $(\mathbf{a}_L, \mathbf{a}_R, [\mathbf{a}_1]_1, [\mathbf{a}_2]_2, [\mathbf{a}_t]_t)$ of the final linear system is uniquely determined by the circuit C and its assignment $(\mathbf{x}_0, [\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{x}_t]_t)$, regardless of the randomness used in the random linear combinations.

To formally describe the above compression procedures, we define the following algorithms CompressStatement, WitnessLength and CompressWitness.

- CompressWitness(C, $\mathbf{x}_0$, $[\mathbf{x}_1]_1$, $[\mathbf{x}_2]_2$, $[\mathbf{x}_t]_t$) = wit: This *deterministic* algorithm inputs a group arithmetic circuit C

[6]A naive option is to use uniformly random linear combinations, which require a large number of fresh randomness. Alternatively, one can use linear combinations where the coefficients are distinct (multivariate) monomials, *e.g.*, $x_1^m, x_1^{m-1}x_2, x_1^{m-2}x_2^2, \ldots$, and the variables $x_1, x_2, \ldots$ are chosen uniformly at random.

with its inputs, and outputs the tuple

$$\mathsf{wit} = (\mathbf{a}_L, \mathbf{a}_R, [\mathbf{a}_1]_1, [\mathbf{a}_2]_2, [\mathbf{a}_t]_t)$$

which corresponds to a satisfying assignment to the system of linear equations.

- $\mathsf{WitnessLength}(C, \mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t) \to (\ell_{(\mathrm{mix})}, \ell_t)$: This *deterministic* algorithm inputs a group arithmetic circuit C with its outputs, and outputs the tuple $(\ell_{(\mathrm{mix})}, \ell_t)$ specifying the length of the witness. Specifically, it holds that $|\mathbf{a}_L| = |\mathbf{a}_R| = |\mathbf{a}_1| = |\mathbf{a}_2| = \ell_{(\mathrm{mix})} \leq 2m_0 + m_1 + m_2 + m_{12} + m_t n_t$, and $|\mathbf{a}_t| = \ell_t$.
- $\mathsf{CompressStatement}(C, \mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t) \to \mathsf{stmt}$: This *probabilistic* algorithm inputs a group arithmetic circuit C with its outputs, and outputs the tuple

$$\mathsf{stmt} = (\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_t, \boldsymbol{\beta}_L, \boldsymbol{\beta}_R, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2, [\boldsymbol{\beta}_t]_t, \zeta_0, [\zeta_1]_1, [\zeta_2]_2, [\zeta_t]_t)$$

which specifies the system of linear equations defined in Section 2.6.2.

By construction, if $C(\mathbf{x}_0, [\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{x}_t]_t) = (\mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t)$ and $\mathsf{wit} = \mathsf{CompressWitness}(C, \mathbf{x}_0, [\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{x}_t]_t)$, then for any $\mathsf{stmt} \in \mathsf{CompressStatement}(C, \mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t)$, it must be the case that wit is a satisfying assignment to the linear system defined by stmt.

Conversely, if wit is a satisfying assignment for the linear system defined by stmt for sufficiently many[7] $\mathsf{stmt} \in \mathsf{CompressStatement}(C, \mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t)$ sampled with independent randomness, then one can recover the original witness $(\mathbf{x}_0, [\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{x}_t]_t)$ from wit (otherwise it would lead to the contradiction that an over-determined linear system of equations has more than one solution). If we consider C of polynomial size, then polynomially many independent randomnesses are sufficient.

3 ARGUMENT FOR GROUP ARITHMETIC (OUTER PROTOCOL)

We construct an interactive argument for the satisfiability of group arithmetic circuits. Following the structure of [13, 14], we first describe a naive protocol, called the outer protocol, which will have communication complexity linear in the size of the circuit. In Section 4, we present a family of efficient protocols for generalized inner product relations, called the inner protocols, which can be composed with the outer protocol to obtain an argument system with communication logarithmic in the circuit size.

3.1 Construction

Formally, we present an interactive argument for the following language $\mathcal{L}$:

$$\mathcal{L} := \left\{ \begin{array}{c} (C, \mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t) : \exists (\mathbf{x}_0, [\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{x}_t]_t) \ s.t. \\ C(\mathbf{x}_0, [\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{x}_t]_t) = (\mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t) \end{array} \right\}$$

where the circuits C are group arithmetic circuits defined in Section 2.5.

(Public-Coin) Setup.

- Compute $(\ell_{(\mathrm{mix})}, \ell_t) = \mathsf{WitnessLength}(C, \mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t)$
- Sample $\mathsf{pp}_0 \leftarrow_{\$} \mathbb{G}_1^2$ (for committing to 1 $\mathbb{Z}_q$ element).
- Sample $\mathsf{pp}_i \leftarrow_{\$} \mathbb{G}_t \times \mathbb{G}_{3-i}$ for $i \in \{1,2\}$ (for committing to 1 $\mathbb{G}_i$ element).
- Sample $\mathsf{pp}_t \leftarrow_{\$} \mathbb{G}_t^2$ (for committing to 1 $\mathbb{G}_t$ element).
- Sample $b \leftarrow_{\$} \mathbb{Z}_q$ and $\mathbf{b}_t \leftarrow_{\$} \mathbb{Z}_q^{\ell_t}$ (for committing to ℓ_t $\mathbb{G}_t$ elements).
- Sample $\mathbf{b}^{(\mathrm{mix})} \leftarrow_{\$} \mathbb{Z}_q^2$, $B_i \leftarrow_{\$} \mathbb{Z}_q^{2\times\ell_{(\mathrm{mix})}}$ for $i \in \{L, R, 1, 2\}$ (for committing to mixed elements).
- Output $\mathsf{pp} := \begin{pmatrix} \{\mathsf{pp}_i\}_{i \in \{0,1,2,t\}}, ([b]_t, [\mathbf{b}_t]_t), \\ ([\mathbf{b}^{(\mathrm{mix})}]_t, [B_L]_t, [B_R]_t, [B_1]_2, [B_2]_1) \end{pmatrix}$.

Main Protocol.

- $\mathcal{P}$: Compute $(\mathbf{a}_L, \mathbf{a}_R, [\mathbf{a}_1]_1, [\mathbf{a}_2]_2, [\mathbf{a}_t]_t) = \mathsf{CompressWitness}(C, \mathbf{x}_0, [\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{x}_t]_t)$.
- $\mathcal{P}$: Commit to the witness $(\mathbf{a}_L, \mathbf{a}_R, [\mathbf{a}_1]_1, [\mathbf{a}_2]_2, [\mathbf{a}_t]_t)$ by computing the following:
  - $[\hat{\mathbf{a}}_L]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_L}(\mathbf{a}_L; r_L)$
  - $[\hat{\mathbf{a}}_R]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_R}(\mathbf{a}_R; r_R)$
  - $[\hat{\mathbf{a}}_1]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_1}([\mathbf{a}_1]_1; r_1)$
  - $[\hat{\mathbf{a}}_2]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_2}([\mathbf{a}_2]_2; r_2)$
  - $[\hat{\mathbf{a}}_t]_t = \mathsf{Com}^{(t)}_{\mathbf{b}_t}([\mathbf{a}_t]_t; r_t)$

  The randomness used in all commitments are sampled uniformly from $\mathbb{Z}_q$.
- $\mathcal{P}$: Sample masking elements (one masking element per element in the witness, denoted by $\mathbf{s}$ with the appropriate subscripts and brackets) from the appropriate domains, and commit to the masking elements by computing:
  - $[\hat{\mathbf{s}}_0]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_L \| B_R}\left(\begin{pmatrix} \mathbf{s}_{L,0} \\ \mathbf{s}_{R,0} \end{pmatrix}; s_0\right)$
  - $[\hat{\mathbf{s}}_1]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_R, B_1}(\mathbf{s}_{R,1}, [\mathbf{s}_{1,1}]_1; s_1)$
  - $[\hat{\mathbf{s}}_2]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_L, B_2}(\mathbf{s}_{L,2}, [\mathbf{s}_{2,2}]_1; s_2)$
  - $[\hat{\mathbf{s}}_t^{(\mathrm{mix})}]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_L, B_1, B_2}\left(\mathbf{s}_{L,t}, [\mathbf{s}_{1,t}]_1, [\mathbf{s}_{2,t}]_2; s_t^{(\mathrm{mix})}\right)$
  - $[\hat{\mathbf{s}}_t^{(t)}]_t = \mathsf{Com}^{(t)}_{\mathbf{b}_t}\left([\mathbf{s}_t]_t; s_t^{(t)}\right)$

  The randomness used in all commitments are sampled uniformly from $\mathbb{Z}_q$.
- $\mathcal{P} \to \mathcal{V}$: Send all the commitments $\begin{pmatrix} [\hat{\mathbf{a}}_L]_t, [\hat{\mathbf{a}}_R]_t, [\hat{\mathbf{a}}_1]_t, [\hat{\mathbf{a}}_2]_t, [\hat{\mathbf{a}}_t]_t, \\ [\hat{\mathbf{s}}_0]_t, [\hat{\mathbf{s}}_1]_t, [\hat{\mathbf{s}}_2]_t, [\hat{\mathbf{s}}_t^{(\mathrm{mix})}]_t, [\hat{\mathbf{s}}_t^{(t)}]_t \end{pmatrix}$.
- $\mathcal{V}$: Sample the randomness θ (uniformly at random from the appropriate domain) to be used in CompressStatement.
- $\mathcal{P} \leftarrow \mathcal{V}$: Send θ.
- $\mathcal{P}, \mathcal{V}$: Compute $\begin{pmatrix} \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_t, \\ \boldsymbol{\beta}_L, \boldsymbol{\beta}_R, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \\ [\boldsymbol{\beta}_t]_t, \zeta_0, [\zeta_1]_1, [\zeta_2]_2, [\zeta_t]_t \end{pmatrix} \leftarrow \mathsf{CompressStatement}(C, \mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t; \theta)$.
- $\mathcal{P}$: (Encoding type $\mathbb{Z}_q$ relation) Prepare the coefficients of the polynomials (in X_0)

$$\mathbf{l}_0(X_0) := (\mathbf{a}_L + \boldsymbol{\beta}_L) \cdot X_0 + \mathbf{s}_{L,0}$$
$$\mathbf{r}_0(X_0) := (\boldsymbol{\alpha}_0 \circ \mathbf{a}_R + \boldsymbol{\beta}_R) \cdot X_0^2 + \boldsymbol{\alpha}_0 \circ \mathbf{s}_{R,0}$$
$$p_0(X_0) := \langle \mathbf{l}_0(X_0), \mathbf{r}_0(X_0) \rangle := \sum_{i=0}^{3} p_{0,i} X_0^i$$

Note that $p_{0,3} = \langle \mathbf{a}_L + \boldsymbol{\beta}_L, \boldsymbol{\alpha}_0 \circ \mathbf{a}_R + \boldsymbol{\beta}_R \rangle = \zeta_0$ (known by $\mathcal{V}$).

[7] It suffices for the witness to satisfy a number of statements equal to the number of random coefficients used to derive a compressed statement, which is polynomial in the size of the original statement. For example, using the multivariate monomial technique, a number equal to the number of distinct monomials suffices.

- $\mathcal{P}$: (Encoding type $\mathbb{G}_1$ relation) Prepare the coefficients of the polynomials (in X_1)

$$[\mathbf{l}_1(X_1)]_1 := [\mathbf{a}_1]_1 \cdot X_1 + [\mathbf{s}_{1,1}]_1$$
$$\mathbf{r}_1(X_1) := (\boldsymbol{\alpha}_1 \circ \mathbf{a}_R + \boldsymbol{\beta}_1) \cdot X_1^2 + \boldsymbol{\alpha}_1 \circ \mathbf{s}_{R,1}$$
$$[p_1(X_1)]_1 := \langle [\mathbf{l}_1(X_1)]_1, \mathbf{r}_1(X_1) \rangle := \sum_{i=0}^{3} [p_{1,i}]_1 X_1^i$$

Note that $[p_{1,3}]_1 = \langle [\mathbf{a}_1]_1, \boldsymbol{\alpha}_1 \circ \mathbf{a}_R + \boldsymbol{\beta}_1 \rangle = [\zeta_1]_1$.
- $\mathcal{P}$: (Encoding type $\mathbb{G}_2$ relation) Prepare the coefficients of the polynomials (in X_2)

$$\mathbf{l}_2(X_2) := (\boldsymbol{\alpha}_2 \circ \mathbf{a}_L + \boldsymbol{\beta}_2) \cdot X_2^2 + \boldsymbol{\alpha}_2 \circ \mathbf{s}_{L,2}$$
$$[\mathbf{r}_2(X_2)]_2 := [\mathbf{a}_2]_2 \cdot X_2 + [\mathbf{s}_{2,2}]_1$$
$$[p_2(X_2)]_2 := \langle \mathbf{l}_2(X_2), [\mathbf{r}_2(X_2)]_2 \rangle := \sum_{i=0}^{3} [p_{2,i}]_2 X_2^i$$

Note that $[p_{2,3}]_2 = \langle \boldsymbol{\alpha}_2 \circ \mathbf{a}_L + \boldsymbol{\beta}_2, [\mathbf{a}_2]_2 \rangle = [\zeta_2]_2$.
- $\mathcal{P}$: (Encoding type $\mathbb{G}_t$ relation) Prepare the coefficients of the polynomials (in X_t)

$$\mathbf{l}_{t,0}(X_t) := \mathbf{a}_L \cdot X_t + \mathbf{s}_{L,t}$$
$$[\mathbf{r}_{t,t}(X_t)]_t := \begin{pmatrix} [\mathbf{r}_{t,t,0}(X_t)]_t \\ [\mathbf{r}_{t,t,1}(X_t)]_t \end{pmatrix} := \begin{pmatrix} [\boldsymbol{\beta}_t]_t \\ [\mathbf{a}_t]_t \end{pmatrix} \cdot X_t^4 + \begin{pmatrix} [\mathbf{0}]_t \\ [\mathbf{s}_t]_t \end{pmatrix}$$
$$[\mathbf{l}_{t,1}(X_t)]_1 := \boldsymbol{\alpha}_t \circ [\mathbf{a}_1]_1 \cdot X_t^2 + \boldsymbol{\alpha}_t \circ [\mathbf{s}_{1,t}]_1$$
$$[\mathbf{r}_{t,2}(X_t)]_2 := [\mathbf{a}_2]_2 \cdot X_t^3 + [\mathbf{s}_{2,t}]_2$$
$$[p_t(X_t)]_t := \langle \mathbf{l}_{t,0}(X_t), [\mathbf{r}_{t,t}(X_t)]_t \rangle + \langle [\mathbf{l}_{t,1}(X_t)]_1, [\mathbf{r}_{t,2}(X_t)]_2 \rangle := \sum_{i=0}^{5} [p_{t,i}]_t X_t^i$$

Note that $[p_{t,5}]_2 = \left\langle \mathbf{a}_L, \begin{bmatrix} \boldsymbol{\beta}_t \\ \mathbf{a}_t \end{bmatrix}_t \right\rangle + \langle \boldsymbol{\alpha}_t \circ [\mathbf{a}_1]_1, [\mathbf{a}_2]_2 \rangle = [\zeta_t]_t$.
- $\mathcal{P}$: Commit to those coefficients of the polynomials $p_0(X_0)$, $[p_1(X_1)]_1$, $[p_1(X_1)]_1$, and $[p_t(X_t)]_t$ which are not constant. That is, the prover computes the following:
  - $[\hat{p}_{0,i}]_1 = \mathsf{Com}^{(0)}_{\mathsf{pp}_0}(p_{0,i}; z_{0,i})$ for $i \in \{0,1,2\}$
  - $[\hat{p}_{1,i}]_t = \mathsf{Com}^{(1)}_{\mathsf{pp}_1}([p_{1,i}]_1; z_{1,i})$ for $i \in \{0,1,2\}$
  - $[\hat{p}_{2,i}]_t = \mathsf{Com}^{(2)}_{\mathsf{pp}_2}([p_{2,i}]_2; z_{2,i})$ for $i \in \{0,1,2\}$
  - $[\hat{\mathbf{p}}_{t,i}]_t = \mathsf{Com}^{(t)}_{\mathsf{pp}_t}([p_{t,i}]_t; z_{t,i})$ for $i \in \{0,1,\ldots,4\}$

  As always, the randomness used in all commitments are sampled uniformly from $\mathbb{Z}_q$.
- $\mathcal{P} \rightarrow \mathcal{V}$: Send all the commitments

$$\left(\{[\hat{p}_{0,i}]_1\}_{i=0}^2, \{[\hat{p}_{1,i}]_t\}_{i=0}^2, \{[\hat{p}_{2,i}]_t\}_{i=0}^2, \{[\hat{\mathbf{p}}_{t,i}]_t\}_{i=0}^4 \right).$$

- $\mathcal{V}$: Sample challenges x_0, x_1, x_2, and x_t uniformly from $\mathbb{Z}_q$.
- $\mathcal{P} \leftarrow \mathcal{V}$: Send (x_0, x_1, x_2, x_t).
- $\mathcal{P}$: Compute the openings of the homomorphically evaluated commitments. That is, the prover computes the following:

$$e_0 = s_0 + r_L \cdot x_0 + r_R \cdot x_0^2$$
$$e_1 = s_1 + r_1 \cdot x_1 + r_R \cdot x_1^2$$
$$e_2 = s_2 + r_2 \cdot x_2 + r_L \cdot x_2^2$$
$$e_t^{(\mathrm{mix})} = s_t^{(\mathrm{mix})} + r_L \cdot x_t + r_1 \cdot x_t^2 + r_2 \cdot x_t^3$$
$$e_t^{(t)} = s_t^{(t)} + r_t \cdot x_t^4$$
$$f_i = \sum_{j=0}^{2} z_{i,j} \cdot x_i^j, \forall i \in \{0,1,2\}$$
$$f_t = \sum_{i=0}^{4} z_{t,i} \cdot x_t^i$$
$$(\tilde{\mathbf{l}}_0, \tilde{\mathbf{r}}_0, \tilde{p}_0) = (\mathbf{l}_0(x_0), \mathbf{r}_0(x_0), p_0(x_0))$$
$$([\tilde{\mathbf{l}}_1]_1, \tilde{\mathbf{r}}_1, [\tilde{p}_1]_1) = ([\mathbf{l}_1(x_1)]_1, \mathbf{r}_1(x_1), [p_1(x_1)]_1)$$
$$(\tilde{\mathbf{l}}_2, [\tilde{\mathbf{r}}_2]_2, [\tilde{p}_2]_2) = (\mathbf{l}_2(x_2), [\mathbf{r}_2(x_2)]_2, [p_2(x_2)]_2)$$
$$\begin{pmatrix} \tilde{\mathbf{l}}_{t,0}, [\tilde{\mathbf{r}}_{t,t,1}]_t, \\ [\tilde{\mathbf{l}}_{t,1}]_1, [\tilde{\mathbf{r}}_{t,2}]_2, [\tilde{p}_t]_t \end{pmatrix} = \begin{pmatrix} \mathbf{l}_{t,0}(x_t), [\mathbf{r}_{t,t,1}(x_t)]_t, \\ [\mathbf{l}_{t,1}(x_t)]_1, [\mathbf{r}_{t,2}(x_t)]_2, [p_t(x_t)]_t \end{pmatrix}$$

- $\mathcal{P} \rightarrow \mathcal{V}$: Send all the openings:

$$\begin{pmatrix} e_0, f_0, \tilde{\mathbf{l}}_0, \tilde{\mathbf{r}}_0, \tilde{p}_0, \\ e_1, f_1, [\tilde{\mathbf{l}}_1]_1, \tilde{\mathbf{r}}_1, [\tilde{p}_1]_1, \\ e_2, f_2, \tilde{\mathbf{l}}_2, [\tilde{\mathbf{r}}_2]_2, [\tilde{p}_2]_2, \\ e_t^{(\mathrm{mix})}, e_t^{(t)}, f_t, \tilde{\mathbf{l}}_{t,0}, [\tilde{\mathbf{r}}_{t,t,1}]_t, [\tilde{\mathbf{l}}_{t,1}]_1, [\tilde{\mathbf{r}}_{t,2}]_2, [\tilde{p}_t]_t \end{pmatrix} \tag{17}$$

- $\mathcal{V}$: Compute $[\tilde{\mathbf{r}}_{t,t,0}]_t = [\boldsymbol{\beta}_t]_t \cdot x_t^4$.
- $\mathcal{V}$: Set $[\tilde{\mathbf{r}}_{t,t}]_t := \begin{pmatrix} [\tilde{\mathbf{r}}_{t,t,0}]_t \\ [\tilde{\mathbf{r}}_{t,t,1}]_t \end{pmatrix}$.
- $\mathcal{V}$: (Checking type $\mathbb{Z}_q$ relation) Let

$$(B'_{L,0}, B'_{R,0}) := (B_L \| B_R \circ (\mathbf{1}^2 (\boldsymbol{\alpha}_0^{\circ -1})^T)) \tag{18}$$
$$[\mathbf{p}_0]_t := -[\mathbf{b}^{(\mathrm{mix})}]_t \cdot e_0 + [\hat{\mathbf{s}}_0]_t + \left([\hat{\mathbf{a}}_L]_t + [B'_{L,0}]_t \boldsymbol{\beta}_L \right) \cdot x_0 + \left([\hat{\mathbf{a}}_R]_t + [B'_{R,0}]_t \boldsymbol{\beta}_R \right) \cdot x_0^2 \tag{19}$$

Check the following :

$$[\mathbf{p}_0]_t \stackrel{?}{=} [B'_{L,0}]_t \tilde{\mathbf{l}}_0 + [B'_{R,0}]_t \tilde{\mathbf{r}}_0 \tag{20}$$
$$\tilde{p}_0 \stackrel{?}{=} \left\langle \tilde{\mathbf{l}}_0, \tilde{\mathbf{r}}_0 \right\rangle \tag{21}$$
$$\mathsf{Com}^{(0)}_{\mathsf{pp}_0}(\tilde{p}_0; f_0) \stackrel{?}{=} \sum_{i=0}^{2} [\hat{p}_{0,i}]_1 \cdot x_0^i + \mathsf{Com}^{(0)}_{\mathsf{pp}_0}(\zeta_0; 0) \cdot x_0^3 \tag{22}$$

- $\mathcal{V}$: (Checking type $\mathbb{G}_1$ relation) Let

$$(B'_{1,1}, B'_{R,1}) := (B_R \circ (\mathbf{1}^2 (\boldsymbol{\alpha}_1^{\circ -1})^T), B_1) \tag{23}$$
$$[\mathbf{p}_1]_t := -[\mathbf{b}^{(\mathrm{mix})}]_t \cdot e_1 + [\hat{\mathbf{s}}_1]_t + [\hat{\mathbf{a}}_1]_t \cdot x_1 + ([\hat{\mathbf{a}}_R]_t + [B'_{1,1}]_t \boldsymbol{\beta}_1) \cdot x_1^2 \tag{24}$$

Check the following :

$$[\mathbf{p}_1]_t \stackrel{?}{=} [B'_{1,1}]_t \tilde{\mathbf{r}}_1 + [B'_{R,1}]_2 [\tilde{\mathbf{l}}_1]_1 \tag{25}$$
$$[\tilde{p}_1]_1 \stackrel{?}{=} \left\langle [\tilde{\mathbf{l}}_1]_1, \tilde{\mathbf{r}}_1 \right\rangle \tag{26}$$
$$\mathsf{Com}^{(1)}_{\mathsf{pp}_1}([\tilde{p}_1]_1; f_1) \stackrel{?}{=} \sum_{i=0}^{2} [\hat{p}_{1,i}]_t \cdot x_1^i + \mathsf{Com}^{(1)}_{\mathsf{pp}_1}([\zeta_1]_1; 0) \cdot x_1^3 \tag{27}$$

- $\mathcal{V}$: (Checking type $\mathbb{G}_2$ relation) Let

$$(B'_{L,2}, B'_{2,2}) := (B_L \circ (\mathbf{1}^2 (\boldsymbol{\alpha}_2^{\circ -1})^T), B_2) \quad (28)$$

$$[\mathbf{p}_2]_t := -[\mathbf{b}^{(\text{mix})}]_t \cdot e_2 + [\hat{\mathbf{s}}_2]_t + [\hat{\mathbf{a}}_2]_t \cdot x_2 + ([\hat{\mathbf{a}}_L]_t + [B'_{L,2}]_t \boldsymbol{\beta}_2) \cdot x_2^2 \quad (29)$$

Check the following :

$$[\mathbf{p}_2]_t \stackrel{?}{=} [B'_{L,2}]_t \tilde{\mathbf{l}}_2 + [B'_{2,2}]_1 [\tilde{\mathbf{r}}_2]_2 \quad (30)$$

$$[\tilde{p}_2]_2 \stackrel{?}{=} \left\langle \tilde{\mathbf{l}}_2, [\tilde{\mathbf{r}}_2]_2 \right\rangle \quad (31)$$

$$\mathsf{Com}^{(2)}_{\mathsf{pp}_2}([\tilde{p}_2]_2; f_2) \stackrel{?}{=} \sum_{i=0}^{2} [\hat{p}_{2,i}]_t \cdot x_2^i + \mathsf{Com}^{(2)}_{\mathsf{pp}_2}([\zeta_2]_2; 0) \cdot x_2^3 \quad (32)$$

- $\mathcal{V}$: (Checking type $\mathbb{G}_t$ relation) Let

$$(B'_{L,t}, B'_{1,t}, B'_{2,t}) := (B_L, B_1 \circ (\mathbf{1}^2 (\boldsymbol{\alpha}_t^{\circ -1})^T), B_2) \quad (33)$$

$$[\mathbf{p}_t]_t := -[\mathbf{b}^{(\text{mix})}]_t \cdot e_t^{(\text{mix})} + [\hat{\mathbf{s}}_t^{(\text{mix})}]_t + [\hat{\mathbf{a}}_L]_t \cdot x_t + [\hat{\mathbf{a}}_1]_t \cdot x_t^2 + [\hat{\mathbf{a}}_2]_t \cdot x_t^3 \quad (34)$$

Check the following :

$$\begin{bmatrix} 0 \\ \tilde{\mathbf{r}}_{t,t,1} \end{bmatrix}_t \stackrel{?}{=} -\begin{bmatrix} g \\ \mathbf{b}_t \end{bmatrix}_t e_t^{(t)} + [\hat{\mathbf{s}}_t^{(t)}]_t + [\hat{\mathbf{a}}_t]_t \cdot x_t^4 \quad (35)$$

$$[\mathbf{p}_t]_t \stackrel{?}{=} [B'_{L,t}]_t \tilde{\mathbf{l}}_{t,0} + [B'_{1,t}]_2 [\tilde{\mathbf{l}}_{t,1}]_1 + [B'_{2,t}]_1 [\tilde{\mathbf{r}}_{t,2}]_2 \quad (36)$$

$$[\tilde{p}_t]_t \stackrel{?}{=} \left\langle \tilde{\mathbf{l}}_{t,0}, [\tilde{\mathbf{r}}_{t,t}]_t \right\rangle + \left\langle [\tilde{\mathbf{l}}_{t,1}]_1, [\tilde{\mathbf{r}}_{t,2}]_2 \right\rangle \quad (37)$$

$$\mathsf{Com}^{(t)}_{\mathsf{pp}_t}([\tilde{p}_t]_t; f_t) \stackrel{?}{=} \sum_{i=0}^{4} [\hat{\mathbf{p}}_{t,i}]_t \cdot x_t^i + \mathsf{Com}^{(t)}_{\mathsf{pp}_t}([\zeta_t]_t; 0) \cdot x_t^5 \quad (38)$$

- $\mathcal{V}$: Output 1 if all the above checks are passed. Otherwise output 0.

In Appendix B we give a concise summary of the message flow of the protocol.

3.2 Security

In the following we show that our scheme is computationally honest-verifier zero-knowledge and has computational witness-extended emulation. We recall that the scheme can be made non-interactive via the standard Fiat-Shamir transformation [20].

Theorem 3.1. *The interactive argument described above is perfectly complete. If* $\mathsf{Com}^{(0)}$, $\mathsf{Com}^{(1)}$, $\mathsf{Com}^{(2)}$, $\mathsf{Com}^{(t)}$, *and* $\mathsf{Com}^{(\text{mix})}$ *are computationally hiding, then it is computationally special honest-verifier zero-knowledge.*

For the proof we refer to Appendix B.1

Theorem 3.2. *If* $\mathsf{Com}^{(0)}$, $\mathsf{Com}^{(1)}$, $\mathsf{Com}^{(2)}$, $\mathsf{Com}^{(t)}$, *and* $\mathsf{Com}^{(\text{mix})}$ *are computationally binding, and the* $(2, n_0, n_1, n_2)$*-GDLR assumption holds for* $n_0, n_1, n_2 = \text{poly}(\lambda)$*, then the interactive argument described above has computational witness-extended emulation.*

For the proof we refer to Appendix B.2

3.3 Achieving Logarithmic Communication

We show how to modify the naive protocol above, such that it can be composed with the inner protocols described in Section 4 to yield a succinct protocol.

In Equation (17) of the naive protocol, the prover has to send the vectors

$$\begin{pmatrix} \tilde{\mathbf{l}}_0, \tilde{\mathbf{r}}_0, [\tilde{\mathbf{l}}_1]_1, \tilde{\mathbf{r}}_1, \tilde{\mathbf{l}}_2, [\tilde{\mathbf{r}}_2]_2, \\ \tilde{\mathbf{l}}_{t,0}, [\tilde{\mathbf{r}}_{t,t,1}]_t, [\tilde{\mathbf{l}}_{t,1}]_1, [\tilde{\mathbf{r}}_{t,2}]_2 \end{pmatrix}$$

whose total size are linear in the size of the witness.

To achieve logarithmic communication, the prover would instead send the commitments of the vectors

$$[\mathbf{p}_0]_t := [B'_{L,0}]_t \tilde{\mathbf{l}}_0 + [B'_{R,0}]_t \tilde{\mathbf{r}}_0$$

$$[\mathbf{p}_1]_t := [B'_{R,1}]_t \tilde{\mathbf{r}}_1 + [B'_{1,1}]_2 [\tilde{\mathbf{l}}_1]_1$$

$$[\mathbf{p}_2]_t := [B'_{L,2}]_t \tilde{\mathbf{l}}_2 + [B'_{2,2}]_1 [\tilde{\mathbf{r}}_2]_2$$

$$[\mathbf{p}_t]_t := [B'_{L,t}]_t \tilde{\mathbf{l}}_{t,0} + [B'_{1,t}]_2 [\tilde{\mathbf{l}}_{t,1}]_1 + [B'_{2,t}]_1 [\tilde{\mathbf{r}}_{t,2}]_2$$

where the bases are defined in Equations (18), (23), (28) and (33).

Correspondingly, instead of having the verifier $\mathcal{V}$ check the equations such as Equations (20) and (21) for type $\mathbb{Z}_q$ relation in plain, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ engage in the inner protocol described in Section 4.1. To check Equations (25), (26), (30) and (31) for type $\mathbb{G}_1$ and $\mathbb{G}_2$ relations, they engage in the inner protocols described in Section 4.2. Finally, to check Equations (36) and (37), they run the protocol in Section 4.3. As we will show in Section 4, all these inner protocols have logarithmic communication complexity, so does the composed protocol.

Note that the commitments $([\mathbf{p}_0]_t, [\mathbf{p}_1]_t, [\mathbf{p}_2]_t, [\mathbf{p}_t]_t)$ need not be hiding and the inner protocols need not be zero-knowledge, as the committed vectors are sent in plain in Equation (17). Due to the witness-extended emulation property of the inner protocols, the composed protocol still has witness-extended emulation.

4 ARGUMENTS FOR GENERALIZED INNER PRODUCTS (INNER PROTOCOLS)

In this section, we present a family of arguments for generalized inner product relations (inner protocols).

4.1 Protocol for Type $\mathbb{Z}_q$ Inner Products

As a warm-up, we present the protocol for "type $\mathbb{Z}_q$" inner products. While this protocol follows with minor modifications from existing work, its presentation is instrumental to familiarize with the notation and for a more incremental exposition of the subsequent arguments. Specifically, we give a succinct argument for the following language

$$\mathcal{L}_0 := \left\{ \begin{array}{l} ([B_0 \| B_1]_t, [\mathbf{p}]_t, c) \ s.t.\ \exists (\mathbf{a}_0, \mathbf{a}_1): \\ [\mathbf{p}]_t = [B_0]_t \mathbf{a}_0 + [B_1]_t \mathbf{a}_1 \wedge c = \langle \mathbf{a}_0, \mathbf{a}_1 \rangle \end{array} \right\}$$

where $[B_0 \| B_1]_t \in \mathbb{G}_t^{2 \times 2n}$, $[\mathbf{p}]_t \in \mathbb{G}_t^2$, $\mathbf{a}_0 \in \mathbb{Z}_q^n$, and $\mathbf{a}_1 \in \mathbb{Z}_q^n$. Similarly as what done in [14], we actually design a protocol for the following related language

$$\tilde{\mathcal{L}}_0 := \left\{ \begin{array}{l} ([B_0 \| B_1 \| \mathbf{b}]_t, [\mathbf{p}]_t) \text{ s.t.} \exists (\mathbf{a}_0, \mathbf{a}_1): \\ [\mathbf{p}]_t = [B_0]_t \mathbf{a}_0 + [B_1]_t \mathbf{a}_1 + [\mathbf{b}]_t \langle \mathbf{a}_0, \mathbf{a}_1 \rangle \end{array} \right\}$$

where $[\mathbf{b}]_t \leftarrow_{\$} \mathbb{G}_t^2$. An argument system for $\tilde{\mathcal{L}}_0$ can be shown to imply an argument for $\mathcal{L}_0$ via a simple conversion similar to that in [14]. The protocol consists of a parallel execution of the inner product argument in [14], as our commitment to the witness now consists of 2 elements instead of 1.

Protocol 1.

- If $n = 1$, then $\mathcal{P}$ simply sends $(\mathbf{a}_0, \mathbf{a}_1)$. $\mathcal{V}$ outputs 1 if

$$[\mathbf{p}]_t = [B_0]_t \mathbf{a}_0 + [B_1]_t \mathbf{a}_1 + [\mathbf{b}]_t \cdot \langle \mathbf{a}_0, \mathbf{a}_1 \rangle.$$

- Else set $\tilde{n} = n/2$ and parse $B_0, B_1, \mathbf{a}_0, \mathbf{a}_1$ as

$$B_{00} \| B_{01} = B_0 \qquad B_{10} \| B_{11} = B_1$$

$$(\mathbf{a}_{00}^T \| \mathbf{a}_{01}^T) = \mathbf{a}_0^T \qquad (\mathbf{a}_{10}^T \| \mathbf{a}_{11}^T) = \mathbf{a}_1^T$$

  where $B_{00}, B_{01}, B_{10}, B_{11} \in \mathbb{Z}_q^{2\times\tilde{n}}$ and $\mathbf{a}_{00}, \mathbf{a}_{01}, \mathbf{a}_{10}, \mathbf{a}_{11} \in \mathbb{Z}_q^{\tilde{n}}$.
- $\mathcal{P}$: Compute
  - $c_L := \langle \mathbf{a}_{00}, \mathbf{a}_{11} \rangle$
  - $c_R := \langle \mathbf{a}_{01}, \mathbf{a}_{10} \rangle$
  - $[\mathbf{l}]_t := [B_{01}]_t \mathbf{a}_{00} + [B_{10}]_t \mathbf{a}_{11} + [\mathbf{b}]_t \cdot c_L$
  - $[\mathbf{r}]_t := [B_{00}]_t \mathbf{a}_{01} + [B_{11}]_t \mathbf{a}_{10} + [\mathbf{b}]_t \cdot c_R$
- $\mathcal{P} \to \mathcal{V}$: Send $([\mathbf{l}]_t, [\mathbf{r}]_t)$.
- $\mathcal{V}$: Sample $x \leftarrow_{\$} \mathbb{Z}_q$.
- $\mathcal{P} \leftarrow \mathcal{V}$: Send x.
- $\mathcal{P}, \mathcal{V}$: Compute
  - $[\tilde{B}_0]_t := [B_{00}]_t \cdot x^{-1} + [B_{01}]_t \cdot x$
  - $[\tilde{B}_1]_t := [B_{10}]_t \cdot x + [B_{11}]_t \cdot x^{-1}$
  - $[\tilde{\mathbf{p}}]_t := [\mathbf{l}]_t \cdot x^2 + [\mathbf{p}]_t + [\mathbf{r}]_t \cdot x^{-2}$
- $\mathcal{P}$: Compute
  - $\tilde{\mathbf{a}}_0 := \mathbf{a}_{00} \cdot x + \mathbf{a}_{01} \cdot x^{-1}$
  - $\tilde{\mathbf{a}}_1 := \mathbf{a}_{10} \cdot x^{-1} + \mathbf{a}_{11} \cdot x$
- $\mathcal{P}$, $\mathcal{V}$: Recursively engage in Protocol 1 on the statement $\left([\tilde{B}_0 \| \tilde{B}_1 \| \mathbf{b}]_t, [\tilde{\mathbf{p}}]_t\right)$ with $(\tilde{\mathbf{a}}_0, \tilde{\mathbf{a}}_1)$ as the witness.

Theorem 4.1. *If the* (2, 2*n*, 0, 0)*-GDLR assumption holds, then Protocol 1 has computational witness-extended emulation.*

For the proof we refer to Appendix C.1.

4.2 Protocol for Type $\mathbb{G}_1$ and $\mathbb{G}_2$ Inner Products

Generalizing Protocol 1, we now present an argument for "type-$\mathbb{G}_1$" inner product relations. The protocol for "type-$\mathbb{G}_2$" relations can be obtained analogously, and is omitted. Specifically, for type-$\mathbb{G}_1$, we give a succinct argument for the following language

$$\mathcal{L}_1 := \left\{ \begin{array}{l} ([B_1]_2, [B_0]_t, [\mathbf{p}]_t, [c]_1) \text{ s.t.} \exists ([\mathbf{a}_1]_1, \mathbf{a}_0) : \\ [\mathbf{p}]_t = [B_1]_2 [\mathbf{a}_1]_1 + [B_0]_t \mathbf{a}_0 \wedge [c]_1 = \langle [\mathbf{a}_1]_1, \mathbf{a}_0 \rangle \end{array} \right\}$$

where $[B_1]_2 \in \mathbb{G}_2^{2\times n}$, $[B_0]_t \in \mathbb{G}_t^{2\times n}$, $[\mathbf{p}]_t \in \mathbb{G}_t^2$, $[\mathbf{a}_1]_1 \in \mathbb{G}_1^n$, and $\mathbf{a}_0 \in \mathbb{Z}_q^n$. This is done by describing an argument for the following related language

$$\tilde{\mathcal{L}}_1 := \left\{ \begin{array}{l} ([B_1 \| \mathbf{b}]_2, [B_0]_t, [\mathbf{p}]_t) \text{ s.t.} \exists ([\mathbf{a}_1]_1, \mathbf{a}_0) : \\ [\mathbf{p}]_t = [B_1]_2 [\mathbf{a}_1]_1 + [B_0]_t \mathbf{a}_0 + [\mathbf{b}]_2 \langle [\mathbf{a}_1]_1, \mathbf{a}_0 \rangle \end{array} \right\}$$

where $[\mathbf{b}]_2 \leftarrow_{\$} \mathbb{G}_2^2$. The two can be shown to be equivalent (except with negligible probability) via a standard transformation. The protocol follows the same blueprint as the argument described above, except that it generalizes modular exponentiations to pairing products.

Protocol 2.

- If $n = 1$, then $\mathcal{P}$ simply sends $([\mathbf{a}_1]_1, \mathbf{a}_0)$. $\mathcal{V}$ outputs 1 if

$$[\mathbf{p}]_t = [B_1]_2 \cdot [\mathbf{a}_1]_1 + [B_0]_t \mathbf{a}_0 + [\mathbf{b}]_2 \cdot \langle [\mathbf{a}_1]_1, \mathbf{a}_0 \rangle.$$

- Else set $\tilde{n} = n/2$ and parse $B_1, B_0, \mathbf{a}_1, \mathbf{a}_0$ as

$$B_{00} \| B_{01} = B_0 \qquad B_{10} \| B_{11} = B_1$$

$$(\mathbf{a}_{00}^T \| \mathbf{a}_{01}^T) = \mathbf{a}_0^T \qquad (\mathbf{a}_{10}^T \| \mathbf{a}_{11}^T) = \mathbf{a}_1^T$$

  where $B_{00}, B_{01}, B_{10}, B_{11} \in \mathbb{Z}_q^{2\times\tilde{n}}$ and $\mathbf{a}_{00}, \mathbf{a}_{01}, \mathbf{a}_{10}, \mathbf{a}_{11} \in \mathbb{Z}_q^{\tilde{n}}$.
- $\mathcal{P}$: Compute
  - $[c_L]_1 := \langle [\mathbf{a}_{10}]_1, \mathbf{a}_{01} \rangle$
  - $[c_R]_1 := \langle [\mathbf{a}_{11}]_1, \mathbf{a}_{00} \rangle$
  - $[\mathbf{l}]_t := [B_{11}]_2 [\mathbf{a}_{10}]_1 + [B_{00}]_t \mathbf{a}_{01} + [\mathbf{b}]_2 \cdot [c_L]_1$
  - $[\mathbf{r}]_t := [B_{10}]_2 [\mathbf{a}_{11}]_1 + [B_{01}]_t \mathbf{a}_{00} + [\mathbf{b}]_2 \cdot [c_R]_1$
- $\mathcal{P} \to \mathcal{V}$: Send $([\mathbf{l}]_t, [\mathbf{r}]_t)$.
- $\mathcal{V}$: Sample $x \leftarrow_{\$} \mathbb{Z}_q$.
- $\mathcal{P} \leftarrow \mathcal{V}$: Send x.
- $\mathcal{P}, \mathcal{V}$: Compute
  - $[\tilde{B}_0]_t := [B_{00}]_t \cdot x + [B_{01}]_t \cdot x^{-1}$
  - $[\tilde{B}_1]_2 := [B_{10}]_2 \cdot x^{-1} + [B_{11}]_2 \cdot x$
  - $[\tilde{\mathbf{p}}]_t := [\mathbf{l}]_t \cdot x^2 + [\mathbf{p}]_t + [\mathbf{r}]_t \cdot x^{-2}$
- $\mathcal{P}$: Compute
  - $\tilde{\mathbf{a}}_0 := \mathbf{a}_{00} \cdot x^{-1} + \mathbf{a}_{01} \cdot x$
  - $[\tilde{\mathbf{a}}_1]_1 := [\mathbf{a}_{10}]_1 \cdot x + [\mathbf{a}_{11}]_1 \cdot x^{-1}$
- $\mathcal{P}$, $\mathcal{V}$: Recursively engage in Protocol 2 on the statement $\left([\tilde{B}_1 \| \mathbf{b}]_2, [\tilde{B}_0]_t, [\tilde{\mathbf{p}}]_t\right)$ with $([\tilde{\mathbf{a}}_0]_1, \tilde{\mathbf{a}}_1)$ as the witness.

Theorem 4.2. *If the* (2, *n*, *n*, 0)*-GDLR assumption holds, then Protocol 2 has computational witness-extended emulation.*

For the proof we refer to Appendix C.2.

4.3 Protocol for Type $\mathbb{G}_t$ Inner Products

We further generalize the technique in Protocol 2 to obtain a succinct argument for the following language

$$\mathcal{L}_t := \left\{ \begin{array}{l} ([B_1]_2, [B_2]_1, [B_0]_t, [\mathbf{p}]_t, [\mathbf{a}_t]_t, [c]_t) \text{ s.t.} \exists ([\mathbf{a}_1]_1, [\mathbf{a}_2]_2, \mathbf{a}_0) : \\ [\mathbf{p}]_t = [B_1]_2 [\mathbf{a}_1]_1 + [B_2]_1 [\mathbf{a}_2]_2 + [B_0]_t \mathbf{a}_0 \\ \quad \wedge [c]_t = \langle [\mathbf{a}_1]_1, [\mathbf{a}_2]_2 \rangle + \langle \mathbf{a}_0, [\mathbf{a}_t]_t \rangle \end{array} \right\}$$

where $[B_1]_2 \in \mathbb{G}_2^{2\times n}$, $[B_2]_1 \in \mathbb{G}_1^{2\times n}$, $[B_0]_t \in \mathbb{G}_t^{2\times n}$, $[\mathbf{p}]_t \in \mathbb{G}_t^2$, $[c] \in \mathbb{G}_t$, $[\mathbf{a}_1]_1 \in \mathbb{G}_1^n$, $[\mathbf{a}_2]_2 \in \mathbb{G}_2^n$, $\mathbf{a}_0 \in \mathbb{Z}_q^n$, and $[\mathbf{a}_t]_t \in \mathbb{G}_t^n$. An important difference of $\mathcal{L}_t$ from the other similar languages is that, part of the input, *i.e.*, $[\mathbf{a}_t]_t$, to the inner product is given in the clear: This allows us to handle relations in $\mathbb{G}_t$ without compromising the overall communication complexity.

Protocol 3.

- If $n = 1$, then $\mathcal{P}$ simply sends $([\mathbf{a}_1]_1, [\mathbf{a}_2]_2, \mathbf{a}_0)$. $\mathcal{V}$ outputs 1 if
  - $[\mathbf{p}]_t = [B_1]_2 \cdot [\mathbf{a}_1]_1 + [B_2]_1 \cdot [\mathbf{a}_2]_2 + [B_0]_t \cdot \mathbf{a}_0$ and
  - $[c]_t = \langle [\mathbf{a}_1]_1, [\mathbf{a}_2]_2 \rangle + \langle \mathbf{a}_0, [\mathbf{a}_t]_t \rangle$
- Else set $\tilde{n} = n/2$ and parse $B_0, B_1, B_2, \mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_t$ as

$$B_{00} \| B_{01} = B_0 \qquad B_{10} \| B_{11} = B_1 \qquad B_{20} \| B_{21} = B_2$$

$$(\mathbf{a}_{10}^T \| \mathbf{a}_{11}^T) = \mathbf{a}_1^T \qquad (\mathbf{a}_{20}^T \| \mathbf{a}_{21}^T) = \mathbf{a}_2^T$$

$$(\mathbf{a}_{00}^T \| \mathbf{a}_{01}^T) = \mathbf{a}_0^T \qquad (\mathbf{a}_{t0}^T \| \mathbf{a}_{t1}^T) = \mathbf{a}_t^T$$

  where $B_{00}, B_{01}, B_{10}, B_{11}, B_{21}, B_{21} \in \mathbb{Z}_q^{2\times\tilde{n}}$ and $\mathbf{a}_{10}, \mathbf{a}_{11}, \mathbf{a}_{20}, \mathbf{a}_{21}, \mathbf{a}_{00}, \mathbf{a}_{01}, \mathbf{a}_{t0}, \mathbf{a}_{t1} \in \mathbb{Z}_q^{\tilde{n}}$.
- $\mathcal{P}$: Compute
  - $[c_L]_t := \langle [\mathbf{a}_{10}]_1, [\mathbf{a}_{21}]_2 \rangle + \langle \mathbf{a}_{00}, [\mathbf{a}_{t1}]_t \rangle$
  - $[c_R]_t := \langle [\mathbf{a}_{11}]_1, [\mathbf{a}_{20}]_2 \rangle + \langle \mathbf{a}_{01}, [\mathbf{a}_{t0}]_t \rangle$
  - $[\mathbf{l}]_t := [B_{01}]_2 [\mathbf{a}_{10}]_1 + [B_{10}]_1 [\mathbf{a}_{21}]_2 + [B_{21}]_t \mathbf{a}_{00}$
  - $[\mathbf{r}]_t := [B_{00}]_2 [\mathbf{a}_{11}]_1 + [B_{11}]_1 [\mathbf{a}_{20}]_2 + [B_{20}]_t \mathbf{a}_{01}$
- $\mathcal{P} \to \mathcal{V}$: Send $([\mathbf{l}]_t, [\mathbf{r}]_t, [c_L]_t, [c_R]_t)$.
- $\mathcal{V}$: Sample $x \leftarrow_{\$} \mathbb{Z}_q$.
- $\mathcal{P} \leftarrow \mathcal{V}$: Send x.
- $\mathcal{P}, \mathcal{V}$: Compute
  - $[\tilde{B}_1]_2 := [B_{10}]_2 \cdot x^{-1} + [B_{11}]_2 \cdot x$
  - $[\tilde{B}_2]_1 := [B_{20}]_1 \cdot x + [B_{21}]_1 \cdot x^{-1}$
  - $[\tilde{B}_0]_t := [B_{00}]_t \cdot x^{-1} + [B_{01}]_t \cdot x$
  - $[\tilde{\mathbf{p}}]_t := [\mathbf{l}]_t \cdot x^2 + [\mathbf{p}]_t + [\mathbf{r}]_t \cdot x^{-2}$
  - $[\tilde{c}]_t := [c_L]_t \cdot x^2 + [c]_t + [c_R]_t \cdot x^{-2}$
  - $[\tilde{\mathbf{a}}_t]_t := [\mathbf{a}_{t0}]_t \cdot x^{-1} + [\mathbf{a}_{t1}]_t \cdot x$

	Outer Protocol			Inner $\mathbb{Z}_q$ Protocol		
	$\mathbb{G}_{1/2}$	$\mathbb{G}_t$	Pairing	$\mathbb{G}_{1/2}$	$\mathbb{G}_t$	Pairing
Prover	$16 \cdot l_{mix} + 6$	$16 \cdot l_{mix} + 9 \cdot l_t + 29$	$14 \cdot l_{mix} + 6$	–	$18 \cdot \log(l_{mix})$	–
Verifier	$2 \cdot l_{mix} + 6$	$21 l_{mix} + 8 \cdot l_t + 46$	$9 \cdot l_{mix} + 4$	–	$(4 \cdot l_{mix} + 4) \cdot \log(l_{mix}) + 5$	–

	Inner $\mathbb{G}_{1/2}$ Protocol			Inner $\mathbb{G}_t$ Protocol		
	$\mathbb{G}_{1/2}$	$\mathbb{G}_t$	Pairing	$\mathbb{G}_{1/2}$	$\mathbb{G}_t$	Pairing
Prover	–	$(10 \cdot l_{mix} + 8) \cdot \log(l_{mix})$	–	–	$(11 \cdot l_{mix} + 8) \cdot \log(l_{mix})$	$5 \cdot l_{mix} \cdot \log(l_{mix})$
Verifier	1	$(4 \cdot l_{mix} + 4) \cdot \log(l_{mix}) + 4$	1	–	$13 \cdot l_{mix} \cdot \log(l_{mix}) + 2$	5

Table 1: Number of operations with respect to the compressed witness length $(\ell_{(\text{mix})}, \ell_t)$

- $\mathcal{P}$: Compute
  - $[\tilde{\mathbf{a}}_0]_1 := [\mathbf{a}_{10}]_1 \cdot x + [\mathbf{a}_{11}]_1 \cdot x^{-1}$
  - $[\tilde{\mathbf{a}}_1]_2 := [\mathbf{a}_{20}]_2 \cdot x^{-1} + [\mathbf{a}_{21}]_2 \cdot x$
  - $\tilde{\mathbf{a}}_2 := \mathbf{a}_{00} \cdot x + \mathbf{a}_{01} \cdot x^{-1}$
- $\mathcal{P}$, $\mathcal{V}$: Recursively engage in Protocol 3 on the statement $\left([\tilde{B}_1]_2, [\tilde{B}_2]_1 [\tilde{B}_0]_t, [\tilde{\mathbf{p}}]_t, [\tilde{\mathbf{a}}_t]_t, [\tilde{c}]_t\right)$ with $([\tilde{\mathbf{a}}_1]_1, [\tilde{\mathbf{a}}_2]_2, \tilde{\mathbf{a}}_0)$ as the witness.

Theorem 4.3. *If the $(2, n, n, n)$-GDLR assumption holds, then Protocol 3 has computational witness-extended emulation.*

For the proof we refer to Appendix C.3.

5 PERFORMANCE EVALUATION

We first count the proof size of our argument system and GS proofs for general statements. A (zero-knowledge) proof in our system consists of $16 \log \ell_{(\text{mix})} + 3\ell_t + 71$ total number of elements (combining the counts in $\mathbb{Z}_q$, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_t$), where $\ell_{(\text{mix})} \leq 2m_0 + m_1 + m_2 + m_{12} + m_t n_t$. For GS proofs, which can only support statements with $\ell_t = 0$, a (witness-indistinguishable) proof consists of $2(\ell_0 + \ell_1 + \ell_2) + 4(n_0 + q_0 + q_1 + q_2) + 6(n_1 + n_2) + 8n_t$ total number of elements (combining the counts in $\mathbb{Z}_q$, $\mathbb{G}_1$, and $\mathbb{G}_2$) in the asymmetric pairing setting, and $3(\ell_0 + \ell_1 + \ell_2) + 6(n_0 + q_0 + q_1 + q_2) + 9(n_1 + n_2 + n_t)$ total number of elements in the symmetric pairing setting. Note that $q_0 \leq 4m_0 + 2m_1 + 2m_2 + 2m_{12} + m_t n_t$ and $q_1, q_2 \leq 2m_0 + m_1 + m_2 + m_{12}$.

Table 1 counts the number of exponentiations in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ and pairing operations needed in the protocols, with respect to the length of the compressed witness $(\ell_{(\text{mix})}, \ell_t)$. Here we assume $\mathbb{G}_1 = \mathbb{G}_2$ or at least the cost of exponentiation in both groups are approximately the same. Therefore we add up the number of exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_2$.

For concrete performance evaluation we have implemented our construction and compared it with an existing implementation[8] of GS NIZK proofs [32], which adopts some of the optimizations suggested in subsequent work [18, 25, 27]. Both implementations are written in Java and are based on the JPBC-Library[9]. We also slightly modified the GS proofs implementation so that it takes equal advantage of the optimization done in the JPBC-Library as our implementation.

We instantiate both schemes over two different elliptic curves. The first is a "type A1" curve supported in the JPBC-Library, which specifies a tuple of symmetric pairing groups, where the order of the groups is a product of three 517-bit primes[10]. Although in this setting we are working with symmetric pairing groups, *i.e.*, $\mathbb{G}_1 = \mathbb{G}_2$, for the clarity of presentation we will still denote left- and right-inputs to a pairing gate with the notation $[\cdot]_1$ and $[\cdot]_2$ respectively.

Our second choice is a tuple of prime-order asymmetric pairing groups chosen from the well-studied Barreto-Naehrig (BN) [7] family with a 256-bit order. The expected security is 128-bit.

All experiments are performed with an Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz processor, without parallelization. Numerical figures are obtained by averaging over those in ten independent instances.

5.1 Choice of Statements to be Proven

For a meaningful comparison between our scheme and GS proofs, we need a family of languages parameterized by the length of the witness. We choose the following simple language $\mathcal{L}_n^*$ as an example:

$$\mathcal{L}_n^* := \{([\boldsymbol{\lambda}]_1, [\zeta]_t) : \exists [\mathbf{a}]_2 \ s.t. \ \langle [\boldsymbol{\lambda}]_1, [\mathbf{a}]_2 \rangle = [\zeta]_t\}$$

The length n of the witness is assumed to be a power of 2.

The language $\mathcal{L}_n^*$ is natural in many applications. For example, it captures the knowledge of a valid opening of a commitment in a commitment scheme of [3]. It also captures the knowledge of a decryption key of the KSW predicate encryption scheme [38, Appendix B], with respect to which the decryption of a given ciphertext is correct. We elaborate more on the latter example.

In the KSW scheme, an attribute-based secret key consists of a set of group elements

$$\left\{[k]_2, [k_{1,i}]_2, [k_{2,i}]_2\right\}_{i=1}^{\ell}.$$

Given a ciphertext

$$\left([c']_t, [c]_1, \left\{[c_{1,i}]_1, [c_{2,i}]_1\right\}_{i=1}^{\ell}\right)$$

[8]https://github.com/gijsvl/groth-sahai

[9]http://gas.dia.unisa.it/projects/jpbc/

[10]This choice of the bit-length of the primes appears in the documentation of the JPBC, without mentioning the expected security level. According to [34], 882-bit primes corresponds to an expected 128-bit security. An educated guess is that 517-bit primes offer 80-bit security. The reason of choosing a low bit-length is such that we can produce experimental results in a reasonable time, since the operations in composite-order groups are much more expensive than their prime-order counterparts. Nevertheless, the concrete choice does not affect the relative efficiency between our implementation and that of GS proofs. We stress that however our choice does not serve as a recommendation for parameters for deployed systems.

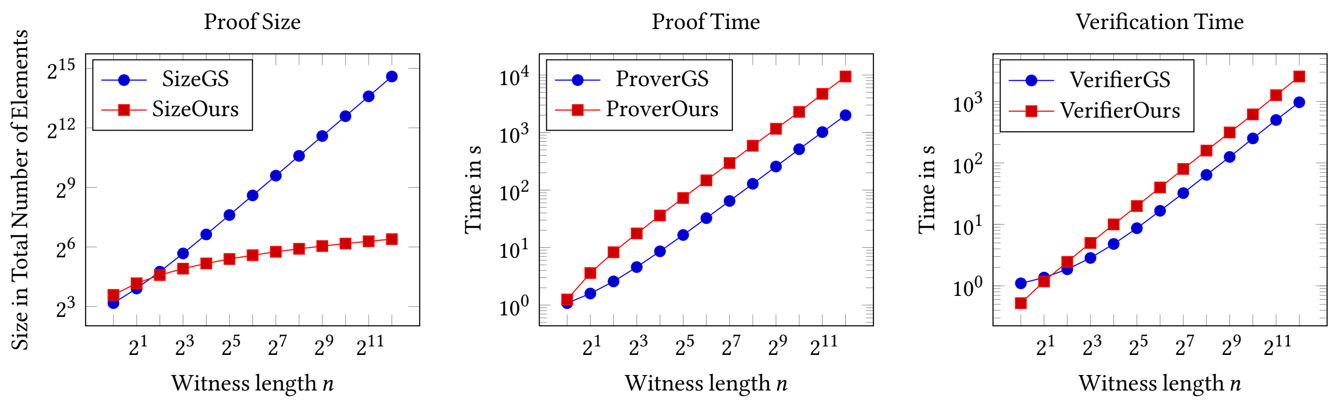

Figure 1: Comparison with Type A1 Curve with 3 517-bit primes

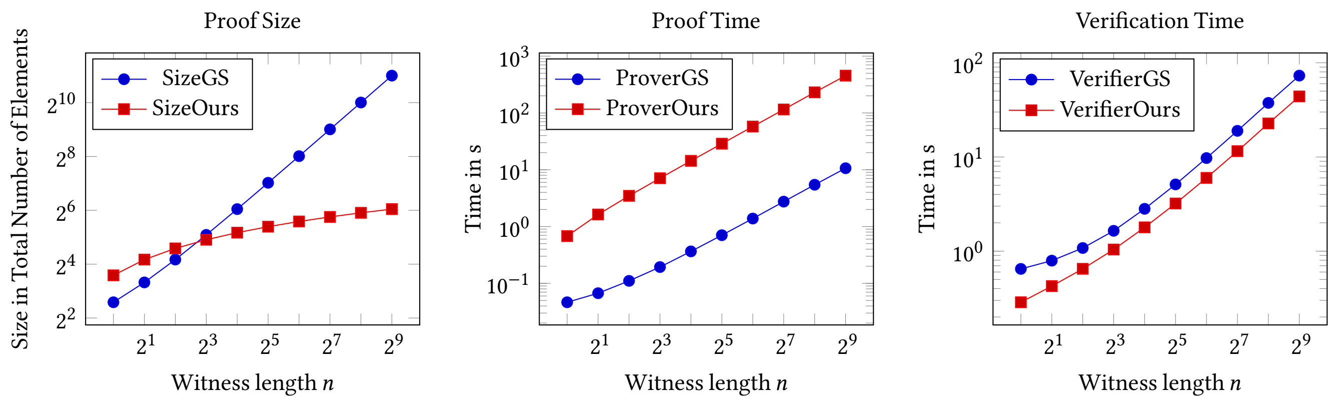

Figure 2: Comparison with BN Curve of 256-bit order

and a message $[m]_t$, proving correct decryption is to prove the knowledge of a secret key which satisfies

$$[c]_1[k]_2+\sum_{i=1}^{\ell}[c_{1,i}]_1[k_{1,i}]_2+\sum_{i=1}^{\ell}[c_{2,i}]_1[k_{2,i}]_2=[m]_t-[c']_t.$$

The KSW predicate encryption scheme is originally described over composite-order symmetric pairing groups, where $\mathbb{G}_1=\mathbb{G}_2$. Subsequent works transformed it into schemes in the prime-order asymmetric pairing group setting (*e.g.*, [5]). Note that operations in composite-order groups are generally much slower than their counterparts in prime-order groups.

We consider the settings where the length of the attribute ℓ of of the form $\ell=(n-1)/2$ for some $n=2^k$. By padding an identity element $[0]_1$ to the statement, we obtain a language in the form of $\mathcal{L}_n^*$.

5.2 Efficiency Comparison

In Figures 1 and 2 we illustrate the prover and verifier time for the two different settings respectively. In all graphs, the x-axes describe the witness length n in $\mathbb{G}_2$ elements, while the y-axes describe the proof size in total number of combined elements or the time in seconds to generate / verify a proof.

Our (zero-knowledge) proof consists of 2 $\mathbb{Z}_q$, 1 $\mathbb{G}_2$, and $(6\log n+3)$ $\mathbb{G}_t$ elements. A (witness-indistinguishable) GS proof consists of 2 $\mathbb{G}_1$ and $2n$ $\mathbb{G}_2$ elements in the BN setting, and $3n+3$ $\mathbb{G}_1/\mathbb{G}_2$ elements in the type A1 setting.

For both schemes and in both settings, it is clear that the prover (resp. verifier) time is linear in the witness length, and is increasing with the same rate.

Over the type A1 curve, our prover is between 3 to 4 times slower than that of GS proof, while the verificaiton time is only 2 to 3 times longer, for witness length $n\geq 2^5$.

Over the BN curve, our prover is around 40 times slower while for verification our scheme only takes about 60% of the time the GS verification needs. We suspect that the difference in prover time is due to specific optimizations for operations in BN curves.

In our implementation we make no use of optimization tricks such as multi-exponentition nor batch verification suggested in the literature (*e.g.*, [14]). Our evaluation shows clearly that even with a very naive implementation our scheme achieves comparable time efficiency with existing systems while the proof length is significantly shorter.

ACKNOWLEDGMENTS

We thank the anonymous referees for their valuable comments and helpful suggestions. This work is supported in part by the German

Research Foundation under Grant No.: 272573906 and 393541319. This work is also supported in part by the state of Bavaria at the Nuremberg Campus of Technology (NCT), and by a gift from DOS Networks and Ripple. NCT is a research cooperation between the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Technische Hochschule Nürnberg Georg Simon Ohm (THN).

REFERENCES

[1] Masayuki Abe. 2015. Structure-Preserving Cryptography. In *Advances in Cryptology - Asiacrypt 2015 (Lecture Notes in Computer Science)*, Vol. 9452. 1. https://www.iacr.org/archive/asiacrypt2015/94520356/94520356.pdf Abstract of invited talk.

[2] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. 2010. Structure-Preserving Signatures and Commitments to Group Elements. In *CRYPTO 2010 (LNCS)*, Tal Rabin (Ed.), Vol. 6223. Springer, Heidelberg, 209–236. https://doi.org/10.1007/978-3-642-14623-7_12

[3] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. 2016. Structure-Preserving Signatures and Commitments to Group Elements. *Journal of Cryptology* 29, 2 (April 2016), 363–421. https://doi.org/10.1007/s00145-014-9196-7

[4] Masayuki Abe, Kristiyan Haralambiev, and Miyako Ohkubo. 2012. Group to Group Commitments Do Not Shrink. In *EUROCRYPT 2012 (LNCS)*, David Pointcheval and Thomas Johansson (Eds.), Vol. 7237. Springer, Heidelberg, 301–317. https://doi.org/10.1007/978-3-642-29011-4_19

[5] Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit Sahai. 2015. On the Practical Security of Inner Product Functional Encryption. In *PKC 2015 (LNCS)*, Jonathan Katz (Ed.), Vol. 9020. Springer, Heidelberg, 777–798. https://doi.org/10.1007/978-3-662-46447-2_35

[6] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. 2017. Ligero: Lightweight Sublinear Arguments Without a Trusted Setup. In *ACM CCS 2017*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM Press, 2087–2104. https://doi.org/10.1145/3133956.3134104

[7] Paulo S. L. M. Barreto and Michael Naehrig. 2006. Pairing-Friendly Elliptic Curves of Prime Order. In *SAC 2005 (LNCS)*, Bart Preneel and Stafford Tavares (Eds.), Vol. 3897. Springer, Heidelberg, 319–331. https://doi.org/10.1007/11693383_22

[8] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. 2018. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046. https://eprint.iacr.org/2018/046.

[9] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. 2019. Aurora: Transparent Succinct Arguments for R1CS. In *EUROCRYPT 2019, Part I (LNCS)*, Yuval Ishai and Vincent Rijmen (Eds.), Vol. 11476. Springer, Heidelberg, 103–128. https://doi.org/10.1007/978-3-030-17653-2_4

[10] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. 2016. Interactive Oracle Proofs. In *TCC 2016-B, Part II (LNCS)*, Martin Hirt and Adam D. Smith (Eds.), Vol. 9986. Springer, Heidelberg, 31–60. https://doi.org/10.1007/978-3-662-53644-5_2

[11] Manuel Blum, Paul Feldman, and Silvio Micali. 1988. Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract). In *20th ACM STOC*. ACM Press, 103–112. https://doi.org/10.1145/62212.62222

[12] Dan Boneh and Matthew K. Franklin. 2001. Identity-Based Encryption from the Weil Pairing. In *CRYPTO 2001 (LNCS)*, Joe Kilian (Ed.), Vol. 2139. Springer, Heidelberg, 213–229. https://doi.org/10.1007/3-540-44647-8_13

[13] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. 2016. Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting, See [21], 327–357. https://doi.org/10.1007/978-3-662-49896-5_12

[14] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short Proofs for Confidential Transactions and More, See [36], 315–334. https://doi.org/10.1109/SP.2018.00020

[15] Jan Camenisch, Kristiyan Haralambiev, Markulf Kohlweiss, Jorn Lapon, and Vincent Naessens. 2011. Structure Preserving CCA Secure Encryption and Applications, See [41], 89–106. https://doi.org/10.1007/978-3-642-25385-0_5

[16] Jan Camenisch and Victor Shoup. 2003. Practical Verifiable Encryption and Decryption of Discrete Logarithms. In *CRYPTO 2003 (LNCS)*, Dan Boneh (Ed.), Vol. 2729. Springer, Heidelberg, 126–144. https://doi.org/10.1007/978-3-540-45146-4_8

[17] Julien Cathalo, Benoît Libert, and Moti Yung. 2009. Group Encryption: Non-interactive Realization in the Standard Model. In *ASIACRYPT 2009 (LNCS)*, Mitsuru Matsui (Ed.), Vol. 5912. Springer, Heidelberg, 179–196. https://doi.org/10.1007/978-3-642-10366-7_11

[18] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. 2012. Malleable Proof Systems and Applications. Cryptology ePrint Archive, Report 2012/012. http://eprint.iacr.org/2012/012.

[19] Taher ElGamal. 1985. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory* 31 (1985), 469–472.

[20] Amos Fiat and Adi Shamir. 1987. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO'86 (LNCS)*, Andrew M. Odlyzko (Ed.), Vol. 263. Springer, Heidelberg, 186–194. https://doi.org/10.1007/3-540-47721-7_12

[21] Marc Fischlin and Jean-Sébastien Coron (Eds.). 2016. *EUROCRYPT 2016, Part II*. LNCS, Vol. 9666. Springer, Heidelberg.

[22] Georg Fuchsbauer. 2011. Commuting Signatures and Verifiable Encryption. In *EUROCRYPT 2011 (LNCS)*, Kenneth G. Paterson (Ed.), Vol. 6632. Springer, Heidelberg, 224–245. https://doi.org/10.1007/978-3-642-20465-4_14

[23] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. 2013. Quadratic Span Programs and Succinct NIZKs without PCPs. In *EUROCRYPT 2013 (LNCS)*, Thomas Johansson and Phong Q. Nguyen (Eds.), Vol. 7881. Springer, Heidelberg, 626–645. https://doi.org/10.1007/978-3-642-38348-9_37

[24] Craig Gentry and Daniel Wichs. 2011. Separating succinct non-interactive arguments from all falsifiable assumptions. In *43rd ACM STOC*, Lance Fortnow and Salil P. Vadhan (Eds.). ACM Press, 99–108. https://doi.org/10.1145/1993636.1993651

[25] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. 2010. Groth-Sahai Proofs Revisited. In *PKC 2010 (LNCS)*, Phong Q. Nguyen and David Pointcheval (Eds.), Vol. 6056. Springer, Heidelberg, 177–192. https://doi.org/10.1007/978-3-642-13013-7_11

[26] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. 2006. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In *ACM CCS 2006*, Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM Press, 89–98. https://doi.org/10.1145/1180405.1180418 Available as Cryptology ePrint Archive Report 2006/309.

[27] Jens Groth. 2006. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In *ASIACRYPT 2006 (LNCS)*, Xuejia Lai and Kefei Chen (Eds.), Vol. 4284. Springer, Heidelberg, 444–459. https://doi.org/10.1007/11935230_29

[28] Jens Groth. 2011. Efficient Zero-Knowledge Arguments from Two-Tiered Homomorphic Commitments, See [41], 431–448. https://doi.org/10.1007/978-3-642-25385-0_23

[29] Jens Groth. 2016. On the Size of Pairing-Based Non-interactive Arguments, See [21], 305–326. https://doi.org/10.1007/978-3-662-49896-5_11

[30] Jens Groth and Yuval Ishai. 2008. Sub-linear Zero-Knowledge Argument for Correctness of a Shuffle, See [54], 379–396. https://doi.org/10.1007/978-3-540-78967-3_22

[31] Jens Groth, Rafail Ostrovsky, and Amit Sahai. 2006. Perfect Non-interactive Zero Knowledge for NP. In *EUROCRYPT 2006 (LNCS)*, Serge Vaudenay (Ed.), Vol. 4004. Springer, Heidelberg, 339–358. https://doi.org/10.1007/11761679_21

[32] Jens Groth and Amit Sahai. 2008. Efficient Non-interactive Proof Systems for Bilinear Groups, See [54], 415–432. https://doi.org/10.1007/978-3-540-78967-3_24

[33] J. Groth and A. Sahai. 2012. Efficient Noninteractive Proof Systems for Bilinear Groups. *SIAM J. Comput.* 41, 5 (2012), 1193–1232. https://doi.org/10.1137/080725386 arXiv:https://doi.org/10.1137/080725386

[34] Aurore Guillevic. 2013. Comparing the Pairing Efficiency over Composite-Order and Prime-Order Elliptic Curves. In *ACNS 13 (LNCS)*, Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini (Eds.), Vol. 7954. Springer, Heidelberg, 357–372. https://doi.org/10.1007/978-3-642-38980-1_22

[35] Christian Hanser and Daniel Slamanig. 2014. Structure-Preserving Signatures on Equivalence Classes and Their Application to Anonymous Credentials. In *ASIACRYPT 2014, Part I (LNCS)*, Palash Sarkar and Tetsu Iwata (Eds.), Vol. 8873. Springer, Heidelberg, 491–511. https://doi.org/10.1007/978-3-662-45611-8_26

[36] IEEE S&P 2018 2018. *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press.

[37] Jonathan Katz, Amit Sahai, and Brent Waters. 2008. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products, See [54], 146–162. https://doi.org/10.1007/978-3-540-78967-3_9

[38] Jonathan Katz, Amit Sahai, and Brent Waters. 2013. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. *Journal of Cryptology* 26, 2 (April 2013), 191–224. https://doi.org/10.1007/s00145-012-9119-4

[39] Joe Kilian. 1992. A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract). In *24th ACM STOC*. ACM Press, 723–732. https://doi.org/10.1145/129712.129782

[40] Russell W. F. Lai and Giulio Malavolta. 2019. Subvector Commitments with Application to Succinct Arguments. In *CRYPTO 2019, Part I (LNCS)*, Alexandra Boldyreva and Daniele Micciancio (Eds.), Vol. 11692. Springer, 530–560.

[41] Dong Hoon Lee and Xiaoyun Wang (Eds.). 2011. *ASIACRYPT 2011*. LNCS, Vol. 7073. Springer, Heidelberg.

[42] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. 2013. Linearly Homomorphic Structure-Preserving Signatures and Their Applications. In *CRYPTO 2013, Part II (LNCS)*, Ran Canetti and Juan A. Garay (Eds.), Vol. 8043. Springer, Heidelberg, 289–307. https://doi.org/10.1007/978-3-642-40084-1_17

[43] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. 2015. Linearly homomorphic structure-preserving signatures and their applications. *Designs, Codes and Cryptography* 77, 2-3 (2015), 441–477.

[44] Benoît Libert, Thomas Peters, and Moti Yung. 2015. Short Group Signatures via Structure-Preserving Signatures: Standard Model Security from Simple Assumptions. In *CRYPTO 2015, Part II (LNCS)*, Rosario Gennaro

and Matthew J. B. Robshaw (Eds.), Vol. 9216. Springer, Heidelberg, 296–316. https://doi.org/10.1007/978-3-662-48000-7_15

[45] Yehuda Lindell. 2003. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. *Journal of Cryptology* 16, 3 (June 2003), 143–184. https://doi.org/10.1007/s00145-002-0143-7

[46] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. 2011. Attribute-Based Signatures. In *CT-RSA 2011 (LNCS)*, Aggelos Kiayias (Ed.), Vol. 6558. Springer, Heidelberg, 376–392. https://doi.org/10.1007/978-3-642-19074-2_24

[47] Ralph C. Merkle. 1988. A Digital Signature Based on a Conventional Encryption Function. In *CRYPTO'87 (LNCS)*, Carl Pomerance (Ed.), Vol. 293. Springer, Heidelberg, 369–378. https://doi.org/10.1007/3-540-48184-2_32

[48] Silvio Micali. 1994. CS Proofs (Extended Abstracts). In *35th FOCS*. IEEE Computer Society Press, 436–453. https://doi.org/10.1109/SFCS.1994.365746

[49] Silvio Micali. 2000. Computationally Sound Proofs. *SIAM J. Comput.* 30, 4 (Oct. 2000), 1253–1298. https://doi.org/10.1137/S0097539795284959

[50] Torben P. Pedersen. 1992. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO'91 (LNCS)*, Joan Feigenbaum (Ed.), Vol. 576. Springer, Heidelberg, 129–140. https://doi.org/10.1007/3-540-46766-1_9

[51] Amit Sahai. 1999. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In *40th FOCS*. IEEE Computer Society Press, 543–553. https://doi.org/10.1109/SFFCS.1999.814628

[52] Amit Sahai and Brent R. Waters. 2005. Fuzzy Identity-Based Encryption. In *EUROCRYPT 2005 (LNCS)*, Ronald Cramer (Ed.), Vol. 3494. Springer, Heidelberg, 457–473. https://doi.org/10.1007/11426639_27

[53] Claus-Peter Schnorr. 1990. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89 (LNCS)*, Gilles Brassard (Ed.), Vol. 435. Springer, Heidelberg, 239–252. https://doi.org/10.1007/0-387-34805-0_22

[54] Nigel P. Smart (Ed.). 2008. *EUROCRYPT 2008*. LNCS, Vol. 4965. Springer, Heidelberg.

[55] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. 2018. Doubly-Efficient zkSNARKs Without Trusted Setup, See [36], 926–943. https://doi.org/10.1109/SP.2018.00060

A PROOFS FOR HARDNESS ASSUMPTION

A.1 Proof of Lemma 2.2

Proof. Let $\mathcal{A}$ be a PPT adversary against the (m, n_0, n_1, n_2)-GDLR assumption. We construct a PPT adversary $\mathcal{B}$ against the $(m,0,n_0+n_1,n_2)$-GDLR assumption.

$\mathcal{B}$ receives an $(m,\ 0,\ n_0\ +\ n_1,\ n_2)$-GDLR instance defined by $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, q, e, [B'_1]_2, [B'_2]_1)$ where $B'_1 \in \mathbb{Z}_q^{m\times(n_0+n_1)}$ and $B'_2 \in \mathbb{Z}_q^{m\times n_2}$. It parses $[B'_1]_2$ as $[B'_0\|B_1]_2$, samples $R_0 \in \mathbb{Z}_q^{n_0\times n_0}$, and computes $[B_0]_t := [B'_0]_2[R_0]_1$. It then runs $\mathcal{A}$ on $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_t,q,e,[B_0]_t,[B_1]_2,[B'_2]_1)$.

Clearly, $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_t,q,e,[B_0]_t,[B_1]_2,[B_2]_1)$ is distributed identically as a random (m,n_0,n_1,n_2)-GDLR instance. Therefore, by assumption, with non-negligible probability, $\mathcal{A}$ outputs $(\mathbf{a}_0,[\mathbf{a}_1]_1,[\mathbf{a}_2]_2)$ not all zero such that

$$[B_0]_t\mathbf{a}_0+[B_1]_2[\mathbf{a}_1]_1+[B_2]_1[\mathbf{a}_2]_2=[\mathbf{0}]_t.$$

Suppose this is the case, by construction we have

$$[B'_1]_2\begin{bmatrix}R_0\mathbf{a}_0\\ \mathbf{a}_1\end{bmatrix}_1+[B'_2]_1[\mathbf{a}_2]_2=[\mathbf{0}]_t.$$

Since $(\mathbf{a}_0,\ [\mathbf{a}_1]_1,\ [\mathbf{a}_2]_2)$ is not all zero, the probability that $\left(\begin{bmatrix}R_0\mathbf{a}_0\\ \mathbf{a}_1\end{bmatrix}_1,[\mathbf{a}_2]_2\right)$ is zero is at most $1/q$ which is negligible. □

A.2 Proof of Theorem 2.3

Proof. By Lemma 2.2, it suffices to prove that the $(m,0,n_1,n_2)$-GDLR assumption holds. We focus on the case $n_1,n_2 \geq 1$. The other cases can be proven analogously.

By the SXDH assumption, the following distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable:

$$\mathcal{D}_1 := \left\{ \begin{pmatrix}\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_t,q,e,\\ [B_1]_2,[B_2]_1\end{pmatrix} : B_i \leftarrow_\$ \mathbb{Z}_q^{m\times n_i},\ i\in\{1,2\} \right\}$$

$$\mathcal{D}_2 := \left\{ \begin{pmatrix}\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_t,q,e,\\ [B_1]_2,[B_2]_1\end{pmatrix} : \begin{matrix}\mathbf{b}_i \leftarrow_\$ \mathbb{Z}_q^{m},\ i\in\{1,2\}\\ \mathbf{r}_i \leftarrow_\$ \mathbb{Z}_q^{n_i},\ i\in\{1,2\}\\ B_i := \mathbf{b}_i\mathbf{r}_i^T,\ i\in\{1,2\}\end{matrix} \right\}$$

From this observation, we can construct a PPT adversary $\mathcal{B}$ against the $(m,0,1,1)$-GDLR assumption given a PPT adversary $\mathcal{A}$ against the $(m,0,n_1,n_2)$-GDLR assumption.

$\mathcal{B}$ receives the $(m,0,1,1)$-GDLR instance defined by $([\mathbf{b}_1]_2,[\mathbf{b}_2]_1)$ where $\mathbf{b}_i \leftarrow_\$ \mathbb{Z}_q^m$ for $i\in\{1,2\}$. It samples $\mathbf{r}_i \leftarrow_\$ \mathbb{Z}_q^{n_i}$ for $i\in\{1,2\}$, and computes $([B_1]_2,[B_2]_1) = ([\mathbf{b}_1]_2\mathbf{r}_1^T,[\mathbf{b}_2]_1\mathbf{r}_2^T)$. It then runs $\mathcal{A}$ to solve the $(m,0,n_1,n_2)$-GDLR instance defined by $([B_1]_2,[B_2]_1)$. From the above observation, this instance is indistinguishable to a random $(m,0,n_1,n_2)$-GDLR instance.

Suppose $\mathcal{A}$ outputs $([\mathbf{a}_1]_1,\ [\mathbf{a}_2]_2)$ not all zero such that $[B_1]_2[\mathbf{a}_1]_1+[B_2]_1[\mathbf{a}_2]_2=[\mathbf{0}]_t$. Let $([a_1]_1,[a_2]_2)=(\mathbf{r}_1^T[\mathbf{a}_1]_1,\mathbf{r}_2^T[\mathbf{a}_2]_2)$. By construction, we have $[\mathbf{b}_1]_2[a_1]_1+[\mathbf{b}_2]_1[a_2]_2=[\mathbf{0}]_t$. Furthermore, since $([\mathbf{a}_1]_1,[\mathbf{a}_2]_2)$ is not all zero, the probability that $([a_1]_1,[a_2]_2)$ is all zero is at most $1/q$, which is negligible. Therefore $\mathcal{B}$ successfully solves the $(m,0,1,1)$-GDLR instance.

However, note that for a random $(m,0,1,1)$-GDLR instance, the matrix $\begin{pmatrix}\mathbf{b}_1 & \mathbf{b}_2\end{pmatrix}$ has full rank $(=2)$ with overwhelming probability. Therefore the only solution to $[\mathbf{b}_1]_2[a_1]_1 + [\mathbf{b}_2]_1[a_2]_2 = [\mathbf{0}]_t$ is $([0]_1,[0]_2)$, which is not a valid solution to the $(m,0,1,1)$-GDLR instance. We can thus conclude that the adversary $\mathcal{B}$, and hence $\mathcal{A}$, cannot exist. □

B PROOFS FOR OUTER PROTOCOL

For a clearer presentation of the security proofs, we recall the message flow of the above protocol:

(1) $\mathcal{P}\rightarrow\mathcal{V}$: $\begin{pmatrix}[\hat{\mathbf{a}}_L]_t,[\hat{\mathbf{a}}_R]_t,[\hat{\mathbf{a}}_1]_t,[\hat{\mathbf{a}}_2]_t,[\hat{\mathbf{a}}_t]_t,\\ [\hat{\mathbf{s}}_0]_t,[\hat{\mathbf{s}}_1]_t,[\hat{\mathbf{s}}_2]_t,[\hat{\mathbf{s}}_t^{(\mathrm{mix})}]_t,[\hat{\mathbf{s}}_t^{(t)}]_t\end{pmatrix}$

(2) $\mathcal{P}\leftarrow\mathcal{V}$: θ

(3) $\mathcal{P}\rightarrow\mathcal{V}$: $\left(\left\{[\hat{p}_{0,i}]_1\right\}_{i=0}^2, \left\{[\hat{p}_{1,i}]_t\right\}_{i=0}^2, \left\{[\hat{p}_{2,i}]_t\right\}_{i=0}^2, \left\{[\hat{\mathbf{p}}_{t,i}]_t\right\}_{i=0}^4\right)$

(4) $\mathcal{P}\leftarrow\mathcal{V}$: (x_0,x_1,x_2,x_t)

(5) $\mathcal{P}\rightarrow\mathcal{V}$:

$$\begin{pmatrix}e_0,f_0,\tilde{\mathbf{l}}_0,\tilde{\mathbf{r}}_0,\tilde{p}_0,\\ e_1,f_1,[\tilde{\mathbf{l}}_1]_1,\tilde{\mathbf{r}}_1,[\tilde{p}_1]_1,\\ e_2,f_2,\tilde{\mathbf{l}}_2,[\tilde{\mathbf{r}}_2]_2,[\tilde{p}_2]_2,\\ e_t^{(\mathrm{mix})},e_t^{(t)},f_t,\tilde{\mathbf{l}}_{t,0},[\tilde{\mathbf{r}}_{t,t,1}]_t,[\tilde{\mathbf{l}}_{t,1}]_1,[\tilde{\mathbf{r}}_{t,2}]_2,[\tilde{p}_t]_t\end{pmatrix}$$

B.1 Proof of Theorem 3.1

Proof. Perfect completeness is straightforward. To prove computational special honest-verifier zero-knowledge, we construct a simulator $\mathcal{S}$ which, on input $(C,\mathbf{y}_0,[\mathbf{y}_1]_1,[\mathbf{y}_2]_2,[\mathbf{y}_t]_t)$ does the following:

- Let θ,x_0,x_1,x_2,x_t be given by the adversary.
- Compute

$$\begin{pmatrix}\boldsymbol{\alpha}_0,\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2,\boldsymbol{\alpha}_t,\\ \boldsymbol{\beta}_L,\boldsymbol{\beta}_R,\boldsymbol{\beta}_1,\boldsymbol{\beta}_2,\\ [\boldsymbol{\beta}_t]_t,\zeta_0,[\zeta_1]_1,[\zeta_2]_2,[\zeta_t]_t\end{pmatrix}$$

$\leftarrow \mathsf{CompressStatement}(C,\mathbf{y}_0,[\mathbf{y}_1]_1,[\mathbf{y}_2]_2,[\mathbf{y}_t]_t;\theta)$.

- Sample the commitments $([\hat{\mathbf{a}}_L]_t,[\hat{\mathbf{a}}_R]_t,[\hat{\mathbf{a}}_1]_t,[\hat{\mathbf{a}}_2]_t,[\hat{\mathbf{a}}_t]_t)$ by committing to random elements of the appropriate length and type using uniform randomness.

- Sample the commitments

$$\left(\{[\hat{p}_{0,i}]_1\}_{i=0}^2, \{[\hat{p}_{1,i}]_t\}_{i=0}^2, \{[\hat{p}_{2,i}]_t\}_{i=0}^2, \{[\hat{\mathbf{p}}_{t,i}]_t\}_{i=0}^4 \right)$$

by committing to random elements of the appropriate length and type using uniform randomness. Specifically, sample uniformly random

$$\begin{pmatrix} \{p_{0,i},z_{0,i}\}_{i=0}^2, \{[p_{1,i}]_1,z_{1,i}\}_{i=0}^2, \\ \{[p_{2,i}]_2,z_{2,i}\}_{i=0}^2, \{[\mathbf{p}_{t,i}]_t,z_{t,i}\}_{i=0}^4 \end{pmatrix}$$

and compute
  - $[\hat{p}_{0,i}]_1 = \mathsf{Com}^{(0)}_{\mathsf{pp}_0}(p_{0,i};z_{0,i})$ for $i \in \{0,1,2\}$
  - $[\hat{p}_{1,i}]_t = \mathsf{Com}^{(1)}_{\mathsf{pp}_1}([p_{1,i}]_1;z_{1,i})$ for $i \in \{0,1,2\}$
  - $[\hat{p}_{2,i}]_t = \mathsf{Com}^{(2)}_{\mathsf{pp}_2}([p_{2,i}]_2;z_{2,i})$ for $i \in \{0,1,2\}$
  - $[\hat{\mathbf{p}}_{t,i}]_t = \mathsf{Com}^{(t)}_{\mathsf{pp}_t}([p_{t,i}]_t;z_{t,i})$ for $i \in \{0,1,\dots,4\}$
- Compute $(\tilde{p}_0,[\tilde{p}_1]_1,[\tilde{p}_2]_2,[\tilde{p}_t]_t,f_0,f_1,f_2,f_t)$ as follows:

$$\tilde{p}_0 = \sum_{i=0}^2 p_{0,i}\cdot x_0^i + \zeta_0 \cdot x_0^3$$

$$[\tilde{p}_1]_1 = \sum_{i=0}^2 [p_{1,i}]_1 \cdot x_1^i + [\zeta_1]_1 \cdot x_1^3$$

$$[\tilde{p}_2]_2 = \sum_{i=0}^2 [p_{2,i}]_2 \cdot x_2^i + [\zeta_2]_2 \cdot x_2^3$$

$$[\tilde{p}_t]_t = \sum_{i=0}^4 [\mathbf{p}_{t,i}]_t \cdot x_t^i + [\zeta_t]_t \cdot x_t^5$$

$$f_i = \sum_{j=0}^2 z_{i,j}\cdot x_i^j, \forall i \in \{0,1,2\}$$

$$f_t = \sum_{i=0}^4 z_{t,i}\cdot x_t^i$$

- Sample $e_0,e_1,e_2,e_t^{(\mathrm{mix})},e_t^{(t)} \leftarrow_\$ \mathbb{Z}_q$.
- Compute $[\tilde{\mathbf{r}}_{t,t,0}]_t = [\boldsymbol{\beta}_t]_t \cdot x_t^4$.
- Sample $\tilde{\mathbf{l}}_0$, $\tilde{\mathbf{r}}_0$, $[\tilde{\mathbf{l}}_1]_1$, $\tilde{\mathbf{r}}_1$, $\tilde{\mathbf{l}}_2$, $[\tilde{\mathbf{r}}_2]_2$, $\tilde{\mathbf{l}}_{t,0}$, $[\tilde{\mathbf{r}}_{t,t,1}]_t$, $[\tilde{\mathbf{l}}_{t,1}]_1$, $[\tilde{\mathbf{r}}_{t,2}]_2$ uniformly at random from the appropriate domain subject to the following linear constraints:

$$\tilde{p}_0 = \left\langle \tilde{\mathbf{l}}_0, \tilde{\mathbf{r}}_0 \right\rangle$$

$$[\tilde{p}_1]_1 = \left\langle [\tilde{\mathbf{l}}_1]_1, \tilde{\mathbf{r}}_1 \right\rangle$$

$$[\tilde{p}_2]_2 = \left\langle \tilde{\mathbf{l}}_2, [\tilde{\mathbf{r}}_2]_2 \right\rangle$$

$$[\tilde{p}_t]_t = \left\langle \tilde{\mathbf{l}}_{t,0}, \begin{pmatrix} [\tilde{\mathbf{r}}_{t,t,0}]_t \\ [\tilde{\mathbf{r}}_{t,t,1}]_t \end{pmatrix} \right\rangle + \left\langle [\tilde{\mathbf{l}}_{t,1}]_1, [\tilde{\mathbf{r}}_{t,2}]_2 \right\rangle$$

- Compute the following:

$$(B'_{L,0} \| B'_{R,0}) = (B_L \| B_R \circ (\mathbf{1}^2(\boldsymbol{\alpha}_0^{\circ -1})^T))$$

$$[\mathbf{p}_0]_t = [B'_{L,0}]_t \tilde{\mathbf{l}}_0 + [B'_{R,0}]_t \tilde{\mathbf{r}}_0$$

$$(B'_{R,1}, B'_{1,1}) = (B_R \circ (\mathbf{1}^2(\boldsymbol{\alpha}_1^{\circ -1})^T), B_1)$$

$$[\mathbf{p}_1]_t = [B'_{R,1}]_t \tilde{\mathbf{r}}_1 + [B'_{1,1}]_2 [\tilde{\mathbf{l}}_1]_1$$

$$(B'_{L,2}, B'_{2,2}) = (B_L \circ (\mathbf{1}^2(\boldsymbol{\alpha}_2^{\circ -1})^T), B_2)$$

$$[\mathbf{p}_2]_t = [B'_{L,2}]_t \tilde{\mathbf{l}}_2 + [B'_{2,2}]_1 [\tilde{\mathbf{r}}_2]_2$$

$$(B'_{L,t}, B'_{1,t}, B'_{2,t}) = (B_L, B_1 \circ (\mathbf{1}^2(\boldsymbol{\alpha}_t^{\circ -1})^T), B_2)$$

$$[\mathbf{p}_t]_t = [B'_{L,t}]_t \tilde{\mathbf{l}}_{t,0} + [B'_{1,t}]_2 [\tilde{\mathbf{l}}_{t,1}]_1 + [B'_{2,t}]_1 [\tilde{\mathbf{r}}_{t,2}]_2$$

- Compute $\left([\hat{\mathbf{s}}_0]_t,[\hat{\mathbf{s}}_1]_t,[\hat{\mathbf{s}}_2]_t,[\hat{\mathbf{s}}_t^{(\mathrm{mix})}]_t,[\hat{\mathbf{s}}_t^{(t)}]_t\right)$ as:

$$\begin{aligned}
[\hat{\mathbf{s}}_0]_t &= [\mathbf{p}_0]_t + [\mathbf{b}^{(\mathrm{mix})}]_t \cdot e_0 - \left([\hat{\mathbf{a}}_L]_t + [B'_{L,0}]_t \boldsymbol{\beta}_L\right) \cdot x_0 \\
&\quad - \left([\hat{\mathbf{a}}_R]_t + [B'_{R,0}]_t \boldsymbol{\beta}_R\right) \cdot x_0^2 \\
[\hat{\mathbf{s}}_1]_t &= [\mathbf{p}_1]_t + [\mathbf{b}^{(\mathrm{mix})}]_t \cdot e_1 - [\hat{\mathbf{a}}_1]_t \cdot x_1 \\
&\quad - ([\hat{\mathbf{a}}_R]_t + [B'_{R,1}]_t \boldsymbol{\beta}_1) \cdot x_1^2 \\
[\hat{\mathbf{s}}_2]_t &= [\mathbf{p}_2]_t + [\mathbf{b}^{(\mathrm{mix})}]_t \cdot e_2 - [\hat{\mathbf{a}}_2]_t \cdot x_2 \\
&\quad - ([\hat{\mathbf{a}}_L]_t + [B'_{L,2}]_t \boldsymbol{\beta}_2) \cdot x_2^2 \\
[\hat{\mathbf{s}}_t^{(\mathrm{mix})}]_t &= [\mathbf{p}_t]_t + [\mathbf{b}^{(\mathrm{mix})}]_t \cdot e_t^{(\mathrm{mix})} - [\hat{\mathbf{a}}_L]_t \cdot x_t \\
&\quad - [\hat{\mathbf{a}}_1]_t \cdot x_t^2 - [\hat{\mathbf{a}}_2]_t \cdot x_t^3 \\
[\hat{\mathbf{s}}_t^{(t)}]_t &= \begin{bmatrix} 0 \\ \tilde{\mathbf{r}}_{t,t,1} \end{bmatrix}_t + \begin{bmatrix} b \\ \mathbf{b}_t \end{bmatrix}_t e_t^{(t)} - [\hat{\mathbf{a}}_t]_t \cdot x_t^4
\end{aligned}$$

  Note that $\left([\hat{\mathbf{s}}_0]_t,[\hat{\mathbf{s}}_1]_t,[\hat{\mathbf{s}}_2]_t,[\hat{\mathbf{s}}_t^{(\mathrm{mix})}]_t,[\hat{\mathbf{s}}_t^{(t)}]_t\right)$ are appropriately structured as commitments.
- Output the simulated transcript.

We analyze the distribution of the transcripts output by the simulator. First note that the commitments $([\hat{\mathbf{a}}_L]_t,[\hat{\mathbf{a}}_R]_t,[\hat{\mathbf{a}}_1]_t,[\hat{\mathbf{a}}_2]_t,[\hat{\mathbf{a}}_t]_t)$ and

$$\left(\{[\hat{p}_{0,i}]_1\}_{i=0}^2, \{[\hat{p}_{1,i}]_t\}_{i=0}^2, \{[\hat{p}_{2,i}]_t\}_{i=0}^2, \{[\hat{\mathbf{p}}_{t,i}]_t\}_{i=0}^4 \right)$$

are computationally indistinguishable from their real counterpart since $\mathsf{Com}^{(0)}$, $\mathsf{Com}^{(1)}$, $\mathsf{Com}^{(2)}$, $\mathsf{Com}^{(t)}$, and $\mathsf{Com}^{(\mathrm{mix})}$ are computationally hiding. The remaining parts of the simulated transcripts distribute identically as their counterparts in real accepting transcripts. □

B.2 Proof of Theorem 3.2

Proof. We would like to show that for any adversary $\mathcal{A}$ producing transcripts with an honest verifier, there exists an extractor $\mathcal{E}$ which produces 1) transcripts which are indistinguishable to those produced by $\mathcal{A}$, and 2) witnesses if the transcripts are accepting. Part 1 is trivial since $\mathcal{E}$ is given access to an oracle which outputs transcripts produced by $\mathcal{A}$. We focus on part 2 and construct an extractor $\mathcal{E}$ as follows.

$\mathcal{E}$ runs $\mathcal{A}$ on sufficiently many uniformly chosen θ (sufficient in the context of Section 2.6.3), and 6 uniformly chosen (x_0,x_1,x_2,x_t). This produces polynomially many transcripts. By assumption, we have that with non-negligible probability, all polynomially many transcripts are accepting. Furthermore, for any value of θ, with overwhelming probability, we have that all 6 values of x_i are distinct, for all $i \in \{0,1,2,t\}$. Suppose that both events happen.

In the following, we first analyze the 6 transcripts for one fixed θ. Let

$$\begin{pmatrix} \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_t, \\ \boldsymbol{\beta}_L, \boldsymbol{\beta}_R, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \\ [\boldsymbol{\beta}_t]_t, \zeta_0, [\zeta_1]_1, [\zeta_2]_2, [\zeta_t]_t \end{pmatrix}$$
$$\leftarrow \mathsf{CompressStatement}(C, \mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t; \theta).$$

Since all 6 transcripts are accepting, we have

$$\begin{aligned} [B'_{L,0}]_t \tilde{\mathbf{l}}_0 + [B'_{R,0}]_t \tilde{\mathbf{r}}_0 = & -[\mathbf{b}^{(\mathrm{mix})}]_t \cdot e_0 + [\hat{\mathbf{s}}_0]_t \\ & + \left([\hat{\mathbf{a}}_L]_t + [B'_{L,0}]_t \boldsymbol{\beta}_L\right) \cdot x_0 \\ & + \left([\hat{\mathbf{a}}_R]_t + [B'_{R,0}]_t \boldsymbol{\beta}_R\right) \cdot x_0^2 \\ [B'_{R,1}]_t \tilde{\mathbf{r}}_1 + [B'_{1,1}]_2 [\tilde{\mathbf{l}}_1]_1 = & -[\mathbf{b}^{(\mathrm{mix})}]_t \cdot e_1 + [\hat{\mathbf{s}}_1]_t \\ & + [\hat{\mathbf{a}}_1]_t \cdot x_1 + ([\hat{\mathbf{a}}_R]_t + [B'_{R,1}]_t \boldsymbol{\beta}_1) \cdot x_1^2 \\ [B'_{L,2}]_t \tilde{\mathbf{l}}_2 + [B'_{2,2}]_1 [\tilde{\mathbf{r}}_2]_2 = & -[\mathbf{b}^{(\mathrm{mix})}]_t \cdot e_2 + [\hat{\mathbf{s}}_2]_t \\ & + [\hat{\mathbf{a}}_2]_t \cdot x_2 + ([\hat{\mathbf{a}}_L]_t + [B'_{L,2}]_t \boldsymbol{\beta}_2) \cdot x_2^2 \\ [B'_{L,t}]_t \tilde{\mathbf{l}}_{t,0} + [B'_{1,t}]_2 [\tilde{\mathbf{l}}_{t,1}]_1 + & [B'_{2,t}]_1 [\tilde{\mathbf{r}}_{t,2}]_2 \\ = & -[\mathbf{b}^{(\mathrm{mix})}]_t \cdot e_t^{(\mathrm{mix})} + [\hat{\mathbf{s}}_t^{(\mathrm{mix})}]_t \\ & + [\hat{\mathbf{a}}_L]_t \cdot x_t + [\hat{\mathbf{a}}_1]_t \cdot x_t^2 + [\hat{\mathbf{a}}_2]_t \cdot x_t^3 \end{aligned}$$

$$\begin{aligned} \begin{bmatrix} 0 \\ \tilde{\mathbf{r}}_{t,t,1} \end{bmatrix}_t = & -\begin{bmatrix} b \\ \mathbf{b}_t \end{bmatrix}_t e_t^{(t)} + [\hat{\mathbf{s}}_t^{(t)}]_t + [\hat{\mathbf{a}}_t]_t \cdot x_t^4 \\ \mathsf{Com}^{(0)}_{\mathsf{pp}_0}(\tilde{p}_0; f_0) = & \sum_{i=0}^{2} [\hat{p}_{0,i}]_1 \cdot x_0^i + \mathsf{Com}^{(0)}_{\mathsf{pp}_0}(\zeta_0; 0) \cdot x_0^3 \\ \mathsf{Com}^{(1)}_{\mathsf{pp}_1}([\tilde{p}_1]_1; f_1) = & \sum_{i=0}^{2} [\hat{p}_{1,i}]_t \cdot x_1^i + \mathsf{Com}^{(1)}_{\mathsf{pp}_1}([\zeta_1]_1; 0) \cdot x_1^3 \\ \mathsf{Com}^{(2)}_{\mathsf{pp}_2}([\tilde{p}_2]_2; f_2) = & \sum_{i=0}^{2} [\hat{p}_{2,i}]_t \cdot x_2^i + \mathsf{Com}^{(2)}_{\mathsf{pp}_2}([\zeta_2]_2; 0) \cdot x_2^3 \\ \mathsf{Com}^{(t)}_{\mathsf{pp}_t}([\tilde{p}_t]_t; f_t) = & \sum_{i=0}^{4} [\hat{\mathbf{p}}_{t,i}]_t \cdot x_t^i + \mathsf{Com}^{(t)}_{\mathsf{pp}_t}([\zeta_t]_t; 0) \cdot x_t^5 \end{aligned}$$

for all 6 distinct values of x_0, x_1, x_2, and x_t respectively. Through Gaussian elimination, we can obtain the following:

- openings of $[\hat{\mathbf{a}}_L]_t$, $[\hat{\mathbf{a}}_R]_t$, $[\hat{\mathbf{a}}_t]_t$, and $[\hat{\mathbf{s}}_0]_t$ with respect to $\mathsf{Com}^{(\mathrm{mix})}_{B_L \| B_R}$
- openings of $[\hat{\mathbf{a}}_R]_t$, $[\hat{\mathbf{a}}_1]_t$, and $[\hat{\mathbf{s}}_1]_t$ with respect to $\mathsf{Com}^{(\mathrm{mix})}_{(B_R, B_1)}$
- openings of $[\hat{\mathbf{a}}_L]_t$, $[\hat{\mathbf{a}}_2]_t$, and $[\hat{\mathbf{s}}_2]_t$ with respect to $\mathsf{Com}^{(\mathrm{mix})}_{(B_L, B_2)}$
- openings of $[\hat{\mathbf{a}}_L]_t$, $[\hat{\mathbf{a}}_1]_t$, $[\hat{\mathbf{a}}_2]_t$, and $[\hat{\mathbf{s}}_t^{(\mathrm{mix})}]_t$ with respect to $\mathsf{Com}^{(\mathrm{mix})}_{(B_L, B_1, B_2)}$
- openings of $[\hat{\mathbf{a}}_t]_t$ and $[\hat{\mathbf{s}}_t^{(t)}]_t$ with respect to $\mathsf{Com}^{(t)}_{\mathbf{b}_t}$
- openings of $[\hat{p}_{0,i}]_1$ with respect to $\mathsf{Com}^{(0)}_{\mathsf{pp}_0}$ for $i \in \{0,1,2\}$
- openings of $[\hat{p}_{1,i}]_t$ with respect to $\mathsf{Com}^{(0)}_{\mathsf{pp}_1}$ for $i \in \{0,1,2\}$
- openings of $[\hat{p}_{2,i}]_t$ with respect to $\mathsf{Com}^{(0)}_{\mathsf{pp}_2}$ for $i \in \{0,1,2\}$
- openings of $[\hat{\mathbf{p}}_{t,i}]_t$ with respect to $\mathsf{Com}^{(0)}_{\mathsf{pp}_t}$ for $i \in \{0,1,\ldots,4\}$

We examine the openings of $[\hat{\mathbf{a}}_L]_t$ with respect to $\mathsf{Com}^{(\mathrm{mix})}_{B_L \| B_R}$, $\mathsf{Com}^{(\mathrm{mix})}_{(B_L, B_2)}$, and $\mathsf{Com}^{(\mathrm{mix})}_{(B_L, B_1, B_2)}$ respectively. Note that only B_L is common in all three bases. Under the GDLR assumption, the only non-zero component in the openings of $[\hat{\mathbf{a}}_L]_t$ corresponds to the basis B_L. We therefore obtain an opening of $[\hat{\mathbf{a}}_L]_t$ with respect to $\mathsf{Com}^{(\mathrm{mix})}_{B_L}$. Similarly, we can obtain the following:

- an opening of $[\hat{\mathbf{a}}_R]_t$ with respect to $\mathsf{Com}^{(\mathrm{mix})}_{B_R}$
- an opening of $[\hat{\mathbf{a}}_1]_t$ with respect to $\mathsf{Com}^{(\mathrm{mix})}_{B_1}$
- an opening of $[\hat{\mathbf{a}}_2]_t$ with respect to $\mathsf{Com}^{(\mathrm{mix})}_{B_2}$

Summarizing the above, we obtain the following

- $[\hat{\mathbf{a}}_L]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_L}(\mathbf{a}_L^\dagger; r_L^\dagger)$
- $[\hat{\mathbf{a}}_R]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_R}(\mathbf{a}_R^\dagger; r_R^\dagger)$
- $[\hat{\mathbf{a}}_1]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_1}([\mathbf{a}_1^\dagger]_1; r_1^\dagger)$
- $[\hat{\mathbf{a}}_2]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_2}([\mathbf{a}_2^\dagger]_2; r_2^\dagger)$
- $[\hat{\mathbf{a}}_t]_t = \mathsf{Com}^{(t)}_{\mathbf{b}_t}\left([\mathbf{a}_t^\dagger]_t; r_t^\dagger\right)$
- $[\hat{\mathbf{s}}_0]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_L \| B_R}\left(\begin{pmatrix} \mathbf{s}_{L,0}^\dagger \\ \mathbf{s}_{R,0}^\dagger \end{pmatrix}; s_0^\dagger\right)$
- $[\hat{\mathbf{s}}_1]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_R, B_1}(\mathbf{s}_{R,1}^\dagger, [\mathbf{s}_{1,1}^\dagger]_1; s_1^\dagger)$
- $[\hat{\mathbf{s}}_2]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_L, B_2}(\mathbf{s}_{L,2}^\dagger, [\mathbf{s}_{2,2}^\dagger]_2; s_2^\dagger)$
- $[\hat{\mathbf{s}}_t^{(\mathrm{mix})}]_t = \mathsf{Com}^{(\mathrm{mix})}_{B_L, B_1, B_2}\left(\mathbf{s}_{L,t}^\dagger, [\mathbf{s}_{1,t}^\dagger]_1, [\mathbf{s}_{2,t}^\dagger]_2; s_t^{\dagger(\mathrm{mix})}\right)$
- $[\hat{\mathbf{s}}_t^{(t)}]_t = \mathsf{Com}^{(t)}_{\mathbf{b}_t}\left([\mathbf{s}_t^\dagger]_t; s_t^{\dagger(t)}\right)$
- $[\hat{p}_{0,i}]_1 = \mathsf{Com}^{(0)}_{\mathsf{pp}_0}(p_{0,i}^\dagger; z_{0,i}^\dagger)$ for $i \in \{0,1,2\}$
- $[\hat{p}_{1,i}]_t = \mathsf{Com}^{(1)}_{\mathsf{pp}_1}([p_{1,i}^\dagger]_1; z_{1,i}^\dagger)$ for $i \in \{0,1,2\}$
- $[\hat{p}_{2,i}]_t = \mathsf{Com}^{(2)}_{\mathsf{pp}_2}([p_{2,i}^\dagger]_2; z_{2,i}^\dagger)$ for $i \in \{0,1,2\}$
- $[\hat{\mathbf{p}}_{t,i}]_t = \mathsf{Com}^{(t)}_{\mathsf{pp}_t}([p_{t,i}^\dagger]_t; z_{t,i}^\dagger)$ for $i \in \{0,1,\ldots,4\}$

where the values with superscript † are known by our extractor. Note that the openings of

$$\begin{pmatrix} [\hat{\mathbf{a}}_L]_t, [\hat{\mathbf{a}}_R]_t, [\hat{\mathbf{a}}_1]_t, [\hat{\mathbf{a}}_2]_t, [\hat{\mathbf{a}}_t]_t, \\ [\hat{\mathbf{s}}_0]_t, [\hat{\mathbf{s}}_1]_t, [\hat{\mathbf{s}}_2]_t, [\hat{\mathbf{s}}_t^{(\mathrm{mix})}]_t, [\hat{\mathbf{s}}_t^{(t)}]_t \end{pmatrix}$$

for any $(\theta, x_0, x_1, x_2, x_t)$ should be identical, since otherwise one could break the binding properties of the commitments. Similarly, for each θ, the openings of

$$\left(\{[\hat{p}_{0,i}]_1\}_{i=0}^2, \{[\hat{p}_{1,i}]_t\}_{i=0}^2, \{[\hat{p}_{2,i}]_t\}_{i=0}^2, \{[\hat{\mathbf{p}}_{t,i}]_t\}_{i=0}^4 \right)$$

(dependent on θ) should be identical for all values of (x_0, x_1, x_2, x_t).

In what follows, we argue that $\mathsf{wit}^\dagger := (\mathbf{a}_L^\dagger, \mathbf{a}_R^\dagger, [\mathbf{a}_1^\dagger]_1, [\mathbf{a}_2^\dagger]_2, [\mathbf{a}_t^\dagger]_t)$ is a valid witness to the system of linear equations specified by $\begin{pmatrix} \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_t, \\ \boldsymbol{\beta}_L, \boldsymbol{\beta}_R, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \\ [\boldsymbol{\beta}_t]_t, \zeta_0, [\zeta_1]_1, [\zeta_2]_2, [\zeta_t]_t \end{pmatrix}$ for all values of θ. Suppose that is the case, then we can extract a witness $(\mathbf{x}_0, [\mathbf{x}_1]_1, [\mathbf{x}_2]_2, [\mathbf{x}_t]_t)$ of $(C, \mathbf{y}_0, [\mathbf{y}_1]_1, [\mathbf{y}_2]_2, [\mathbf{y}_t]_t)$ according to the discussion in Section 2.6.3.

We now put back the above expressions of

$$\left(\{[\hat{p}_{0,i}]_1\}_{i=0}^2, \{[\hat{p}_{1,i}]_t\}_{i=0}^2, \{[\hat{p}_{2,i}]_t\}_{i=0}^2, \{[\hat{\mathbf{p}}_{t,i}]_t\}_{i=0}^4 \right)$$

to the verification equations

$$\begin{aligned}
[B'_{L,0}]_t\tilde{\mathbf{l}}_0+[B'_{R,0}]_t\tilde{\mathbf{r}}_0 &= -[\mathbf{b}^{(\text{mix})}]_t\cdot e_0+[\hat{\mathbf{s}}_0]_t\\
&\quad+\left([\hat{\mathbf{a}}_L]_t+[B'_{L,0}]_t\boldsymbol{\beta}_L\right)\cdot x_0\\
&\quad+\left([\hat{\mathbf{a}}_R]_t+[B'_{R,0}]_t\boldsymbol{\beta}_R\right)\cdot x_0^2\\
[B'_{R,1}]_t\tilde{\mathbf{r}}_1+[B'_{1,1}]_2[\tilde{\mathbf{l}}_1]_1 &= -[\mathbf{b}^{(\text{mix})}]_t\cdot e_1+[\hat{\mathbf{s}}_1]_t\\
&\quad+[\hat{\mathbf{a}}_1]_t\cdot x_1+([\hat{\mathbf{a}}_R]_t+[B'_{R,1}]_t\boldsymbol{\beta}_1)\cdot x_1^2\\
[B'_{L,2}]_t\tilde{\mathbf{l}}_2+[B'_{2,2}]_1[\tilde{\mathbf{r}}_2]_2 &= -[\mathbf{b}^{(\text{mix})}]_t\cdot e_2+[\hat{\mathbf{s}}_2]_t\\
&\quad+[\hat{\mathbf{a}}_2]_t\cdot x_2+([\hat{\mathbf{a}}_L]_t+[B'_{L,2}]_t\boldsymbol{\beta}_2)\cdot x_2^2\\
[B'_{L,t}]_t\tilde{\mathbf{l}}_{t,0}+[B'_{1,t}]_2[\tilde{\mathbf{l}}_{t,1}]_1&+[B'_{2,t}]_1[\tilde{\mathbf{r}}_{t,2}]_2\\
&= -[\mathbf{b}^{(\text{mix})}]_t\cdot e_t^{(\text{mix})}+[\hat{\mathbf{s}}_t^{(\text{mix})}]_t\\
&\quad+[\hat{\mathbf{a}}_L]_t\cdot x_t+[\hat{\mathbf{a}}_1]_t\cdot x_t^2+[\hat{\mathbf{a}}_2]_t\cdot x_t^3\\
\begin{bmatrix}0\\ \tilde{\mathbf{r}}_{t,t,1}\end{bmatrix}_t &= -\begin{bmatrix}b\\ \mathbf{b}_t\end{bmatrix}_t e_t^{(t)}+[\hat{\mathbf{s}}_t^{(t)}]_t+[\hat{\mathbf{a}}_t]_t\cdot x_t^4
\end{aligned}$$

We examine the first verification equation. By rearranging the terms, we obtain a discrete logarithm representation of the identity element with basis $[\mathbf{b}\|B_L\|B_R]$. By the GDLR assumption, this representation must be trivial (all zero). Applying the same argument to the other verification equations, we conclude that the following relations hold for all 5 values of (x_0,x_1,x_2,x_t):

$$\begin{aligned}
\tilde{\mathbf{l}}_0 &= (\mathbf{a}_L^\dagger+\boldsymbol{\beta}_L)\cdot x_0+\mathbf{s}_{L,0}^\dagger\\
\tilde{\mathbf{r}}_0 &= (\boldsymbol{\alpha}_0\circ\mathbf{a}_R^\dagger+\boldsymbol{\beta}_R)\cdot x_0^2+\boldsymbol{\alpha}_0\circ\mathbf{s}_{R,0}^\dagger\\
[\tilde{\mathbf{l}}_1]_1 &= [\mathbf{a}_1^\dagger]_1\cdot x_1+[\mathbf{s}_{1,1}^\dagger]_1\\
\tilde{\mathbf{r}}_1 &= (\boldsymbol{\alpha}_1\circ\mathbf{a}_R^\dagger+\boldsymbol{\beta}_1)\cdot x_1^2+\boldsymbol{\alpha}_1\circ\mathbf{s}_{R,1}^\dagger\\
\tilde{\mathbf{l}}_2 &= (\boldsymbol{\alpha}_2\circ\mathbf{a}_L^\dagger+\boldsymbol{\beta}_2)\cdot x_2^2+\boldsymbol{\alpha}_2\circ\mathbf{s}_{L,2}^\dagger\\
[\tilde{\mathbf{r}}_2]_2 &= [\mathbf{a}_2^\dagger]_2\cdot x_2+[\mathbf{s}_{2,2}^\dagger]_2\\
\tilde{\mathbf{l}}_{t,0} &= \mathbf{a}_L^\dagger\cdot x_t+\mathbf{s}_{L,t}^\dagger\\
[\tilde{\mathbf{r}}_{t,t,1}]_t &= [\mathbf{a}_t^\dagger]_t\cdot x_t^4+[\mathbf{s}_t^\dagger]_t\\
[\tilde{\mathbf{l}}_{t,1}]_1 &= \boldsymbol{\alpha}_t\circ[\mathbf{a}_1^\dagger]_1\cdot x_t^2+\boldsymbol{\alpha}_t\circ[\mathbf{s}_{1,t}^\dagger]_1\\
[\tilde{\mathbf{r}}_{t,2}]_2 &= [\mathbf{a}_2^\dagger]_2\cdot x_t^3+[\mathbf{s}_{2,t}^\dagger]_2
\end{aligned}$$

Note that the left-hand-side of the above system depend on the value of (x_0,x_1,x_2,x_t). For example, for the 5 distinct values of x_0, we have 5 (possibly) distinct values of $\tilde{\mathbf{l}}_0$.

Similarly, we put back the expressions of

$$\left(\ \{[\hat{p}_{0,i}]_1\}_{i=0}^2,\ \{[\hat{p}_{1,i}]_t\}_{i=0}^2,\ \{[\hat{p}_{2,i}]_t\}_{i=0}^2,\ \{[\hat{\mathbf{p}}_{t,i}]_t\}_{i=0}^4\right)$$

to the verification equations

$$\begin{aligned}
\mathsf{Com}^{(0)}_{\mathsf{pp}_0}(\tilde{p}_0;f_0) &= \sum_{i=0}^{2}[\hat{p}_{0,i}]_1\cdot x_0^i+\mathsf{Com}^{(0)}_{\mathsf{pp}_0}(\zeta_0;0)\cdot x_0^3\\
\mathsf{Com}^{(1)}_{\mathsf{pp}_1}([\tilde{p}_1]_1;f_1) &= \sum_{i=0}^{2}[\hat{p}_{1,i}]_t\cdot x_1^i+\mathsf{Com}^{(1)}_{\mathsf{pp}_1}([\zeta_1]_1;0)\cdot x_1^3\\
\mathsf{Com}^{(2)}_{\mathsf{pp}_2}([\tilde{p}_2]_2;f_2) &= \sum_{i=0}^{2}[\hat{p}_{2,i}]_t\cdot x_2^i+\mathsf{Com}^{(2)}_{\mathsf{pp}_2}([\zeta_2]_2;0)\cdot x_2^3\\
\mathsf{Com}^{(t)}_{\mathsf{pp}_t}([\tilde{p}_t]_t;f_t) &= \sum_{i=0}^{4}[\hat{\mathbf{p}}_{t,i}]_t\cdot x_t^i+\mathsf{Com}^{(t)}_{\mathsf{pp}_t}([\zeta_t]_t;0)\cdot x_t^5
\end{aligned}$$

and can conclude that

$$\tilde{p}_0=\sum_{i=0}^{2}p_{0,i}^\dagger x_0^i+\zeta_0x_0^3 \qquad [\tilde{p}_1]_1=\sum_{i=0}^{2}p_{1,i}^\dagger x_1^i+[\zeta_1]_1x_1^3$$

$$[\tilde{p}_2]_2=\sum_{i=0}^{2}p_{2,i}^\dagger x_2^i+[\zeta_2]_2x_2^3 \qquad [\tilde{p}_t]_t=\sum_{i=0}^{5}p_{t,i}^\dagger x_t^i+[\zeta_t]_tx_t^5$$

Combining the above, we obtain the following system of relations

$$\begin{aligned}
&\left\langle(\mathbf{a}_L^\dagger+\boldsymbol{\beta}_L)\cdot x_0+\mathbf{s}_{L,0}^\dagger,(\boldsymbol{\alpha}_0\circ\mathbf{a}_R^\dagger+\boldsymbol{\beta}_R)\cdot x_0^2+\boldsymbol{\alpha}_0\circ\mathbf{s}_{R,0}^\dagger\right\rangle\\
&=\sum_{i=0}^{2}p_{0,i}^\dagger x_0^i+\zeta_0x_0^3\\
&\left\langle[\mathbf{a}_1^\dagger]_1\cdot x_1+[\mathbf{s}_{1,1}^\dagger]_1,(\boldsymbol{\alpha}_1\circ\mathbf{a}_R^\dagger+\boldsymbol{\beta}_1)\cdot x_1^2+\boldsymbol{\alpha}_1\circ\mathbf{s}_{R,1}^\dagger\right\rangle\\
&=\sum_{i=0}^{2}p_{1,i}^\dagger x_1^i+[\zeta_1]_1x_1^3\\
&\left\langle(\boldsymbol{\alpha}_2\circ\mathbf{a}_L^\dagger+\boldsymbol{\beta}_2)\cdot x_2^2+\boldsymbol{\alpha}_2\circ\mathbf{s}_{L,2}^\dagger,[\mathbf{a}_2^\dagger]_2\cdot x_2+[\mathbf{s}_{2,2}^\dagger]_2\right\rangle\\
&=\sum_{i=0}^{2}p_{2,i}^\dagger x_2^i+[\zeta_2]_2x_2^3\\
&\left\langle\mathbf{a}_L^\dagger\cdot x_t+\mathbf{s}_{L,t}^\dagger,\begin{bmatrix}\boldsymbol{\beta}_t\\ \mathbf{a}_t^\dagger\end{bmatrix}_t\cdot x_t^4+\begin{bmatrix}\mathbf{0}\\ \mathbf{s}_t^\dagger\end{bmatrix}_t\right\rangle\\
&+\left\langle\boldsymbol{\alpha}_t\circ[\mathbf{a}_1^\dagger]_1\cdot x_t^2+\boldsymbol{\alpha}_t\circ[\mathbf{s}_{1,t}^\dagger]_1,[\mathbf{a}_2^\dagger]_2\cdot x_t^3+[\mathbf{s}_{2,t}^\dagger]_2\right\rangle\\
&=\sum_{i=0}^{5}p_{t,i}^\dagger x_t^i+[\zeta_t]_tx_t^5
\end{aligned}$$

Note that each of the above relations can be interpreted as a degree-5 polynomial (in some x_i) which has 6 distinct roots. Each polynomial therefore must be a zero polynomial. We can therefore conclude that

$$\begin{aligned}
\left\langle\mathbf{a}_L^\dagger+\boldsymbol{\beta}_L,\boldsymbol{\alpha}_0\circ\mathbf{a}_R^\dagger+\boldsymbol{\beta}_R\right\rangle&=\zeta_0\\
\left\langle[\mathbf{a}_1^\dagger]_1,\boldsymbol{\alpha}_1\circ\mathbf{a}_R^\dagger+\boldsymbol{\beta}_1\right\rangle&=[\zeta_1]_1\\
\left\langle\boldsymbol{\alpha}_2\circ\mathbf{a}_L^\dagger+\boldsymbol{\beta}_2,[\mathbf{a}_2^\dagger]_2\right\rangle&=[\zeta_2]_2\\
\left\langle\mathbf{a}_L^\dagger,\begin{bmatrix}\boldsymbol{\beta}_t\\ \mathbf{a}_t^\dagger\end{bmatrix}_t\right\rangle+\left\langle\boldsymbol{\alpha}_t\circ[\mathbf{a}_1^\dagger]_1,[\mathbf{a}_2^\dagger]_2\right\rangle&=[\zeta_t]_t
\end{aligned}$$

exactly as we desired. □

C PROOFS FOR INNER PROTOCOLS

C.1 Proof of Theorem 4.1

Proof. Let $n=2^i$ be the length of the vectors. We show this via an induction over i. For the base case ($n=1$), the prover simply sends $\mathbf{a}_0$ and $\mathbf{a}_1$ to the verifier. This is trivially witness-extended emulatable since $(\mathbf{a}_0,\mathbf{a}_1)$ is given in clear. We now describe an algorithm that given black-box access to the prover extracts a valid witness for a given statement $([B_0\|B_1\|\mathbf{b}]_t,[\mathbf{p}]_t)$, with all but negligible probability.

The algorithm obtains $([\mathbf{l}]_t,[\mathbf{r}]_t)$ by the prover and rewinds it with four uniformly sampled $(x_1,x_2,x_t,x_4)\leftarrow_{\$}\mathbb{Z}_q^4$. By induction hypothesis, there exists an efficient extractor that outputs $(\mathbf{a}_0^{(i)},\mathbf{a}_1^{(i)})$ such that

$$\begin{aligned}[\tilde{\mathbf{p}}]_t &=[\mathbf{l}]_t\cdot x_i^2+[\mathbf{p}]_t+[\mathbf{r}]_t\cdot x_i^{-2}\\ &=[\tilde{B}_0]_t\mathbf{a}_0^{(i)}+[\tilde{B}_1]_t\mathbf{a}_1^{(i)}+[\mathbf{b}]_t\left\langle \mathbf{a}_0^{(i)},\mathbf{a}_1^{(i)}\right\rangle\\ &=\left([B_{00}]_t\cdot x_i^{-1}+[B_{01}]_t\cdot x_i\right)\cdot\mathbf{a}_0^{(i)}\\ &\quad+\left([B_{10}]_t\cdot x_i+[B_{11}]_t\cdot x_i^{-1}\right)\cdot\mathbf{a}_1^{(i)}+[\mathbf{b}]_t\left\langle \mathbf{a}_0^{(i)},\mathbf{a}_1^{(i)}\right\rangle\end{aligned}$$

for all $i\in\{1,2,3,4\}$. Define (v_1,v_2,v_t) such that

$$\sum_{i=1}^{3}v_ix_i^2=1\qquad \sum_{i=1}^{3}v_i=0\qquad \sum_{i=1}^{3}v_ix_i^{-2}=0$$

and use them as the coefficients to compute, via a linear combination, $(\mathbf{a}_0^{(l)},\mathbf{a}_1^{(l)},c^{(l)})$ such that

$$[\mathbf{l}]_t=[B_0]_t\cdot\mathbf{a}_0^{(l)}+[B_1]_t\cdot\mathbf{a}_1^{(l)}+[\mathbf{b}]_t\cdot c^{(l)}.$$

Use the same procedure to compute

$$\begin{aligned}[\mathbf{r}]_t &=[B_0]_t\cdot\mathbf{a}_0^{(r)}+[B_1]_t\cdot\mathbf{a}_1^{(r)}+[\mathbf{b}]_t\cdot c^{(r)}\ \text{ and}\\ [\mathbf{p}]_t &=[B_0]_t\cdot\mathbf{a}_0^{(p)}+[B_1]_t\cdot\mathbf{a}_1^{(p)}+[\mathbf{b}]_t\cdot c^{(p)}.\end{aligned}$$

Substituting with the equation above, we obtain for all $i\in\{1,2,3,4\}$ the following relation

$$\begin{aligned}&[\mathbf{l}]_t\cdot x_i^2+[\mathbf{p}]_t+[\mathbf{r}]_t\cdot x_i^{-2}\\ &=[B_0]_t\cdot(\mathbf{a}_0^{(l)}x_i^2+\mathbf{a}_0^{(r)}x_i^{-2}+\mathbf{a}_0^{(p)})\\ &\quad+[B_1]_t\cdot(\mathbf{a}_1^{(l)}x_i^2+\mathbf{a}_1^{(r)}x_i^{-2}+\mathbf{a}_1^{(p)})\\ &\quad+[\mathbf{b}]_t\cdot(c^{(l)}x_i^2+c^{(r)}x_i^{-2}+c^{(p)})\\ &=\left([B_{00}]_t\cdot x_i^{-1}+[B_{01}]_t\cdot x_i\right)\cdot\mathbf{a}_0^{(i)}\\ &\quad+\left([B_{10}]_t\cdot x_i+[B_{11}]_t\cdot x_i^{-1}\right)\cdot\mathbf{a}_1^{(i)}\\ &\quad+[\mathbf{b}]_t\left\langle \mathbf{a}_0^{(i)},\mathbf{a}_1^{(i)}\right\rangle.\end{aligned}$$

Then the following equations

$$\begin{aligned}\mathbf{a}_0^{(i)}\cdot x_i^{-1}&=\mathbf{a}_{00}^{(l)}x_i^2+\mathbf{a}_{00}^{(r)}x_i^{-2}+\mathbf{a}_{00}^{(p)}\\ \mathbf{a}_0^{(i)}\cdot x_i&=\mathbf{a}_{01}^{(l)}x_i^2+\mathbf{a}_{01}^{(r)}x_i^{-2}+\mathbf{a}_{01}^{(p)}\\ \mathbf{a}_1^{(i)}\cdot x_i&=\mathbf{a}_{10}^{(l)}x_i^2+\mathbf{a}_{10}^{(r)}x_i^{-2}+\mathbf{a}_{10}^{(p)}\\ \mathbf{a}_1^{(i)}\cdot x_i^{-1}&=\mathbf{a}_{11}^{(l)}x_i^2+\mathbf{a}_{11}^{(r)}x_i^{-2}+\mathbf{a}_{11}^{(p)}\\ \left\langle \mathbf{a}_0^{(i)},\mathbf{a}_1^{(i)}\right\rangle&=c^{(l)}x_i^2+c^{(r)}x_i^{-2}+c^{(p)}\end{aligned}$$

must hold unless a non-trivial discrete logarithm relation among the basis $[B_0]_t,[B_1]_t,[\mathbf{b}]_t$ is revealed. Therefore, with all but negligible probability, we have that

$$\mathbf{a}_{00}^{(l)}x_i^3+(\mathbf{a}_{00}^{(p)}-\mathbf{a}_{01}^{(l)})x_i+(\mathbf{a}_{00}^{(r)}-\mathbf{a}_{01}^{(p)})x_i^{-1}-\mathbf{a}_{01}^{(r)}x_i^{-3}=0 \tag{39}$$

and

$$\mathbf{a}_{11}^{(l)}x_i^3+(\mathbf{a}_{11}^{(p)}-\mathbf{a}_{10}^{(l)})x_i+(\mathbf{a}_{11}^{(r)}-\mathbf{a}_{10}^{(p)})x_i^{-1}-\mathbf{a}_{10}^{(r)}x_i^{-3}=0 \tag{40}$$

Since (x_1,x_2,x_t,x_4) are distinct with all but negligible probability Equation 39 and 40 are satisfied only if all the coefficients are 0. This allows us to simplify the above system to

$$\begin{aligned}\mathbf{a}_0^{(i)}&=\mathbf{a}_{00}^{(p)}\cdot x_i+\mathbf{a}_{01}^{(p)}\cdot x_i^{-1}\\ \mathbf{a}_1^{(i)}&=\mathbf{a}_{11}^{(p)}\cdot x_i+\mathbf{a}_{10}^{(p)}\cdot x_i^{-1}\\ \left\langle \mathbf{a}_0^{(i)},\mathbf{a}_1^{(i)}\right\rangle&=c^{(l)}x_i^2+c^{(r)}x_i^{-2}+c^{(p)}.\end{aligned}$$

Substituting we have

$$c^{(l)}x_i^2+c^{(r)}x_i^{-2}+c^{(p)}=\left\langle \mathbf{a}_{00}^{(p)},\mathbf{a}_{11}^{(p)}\right\rangle x_i^2+\left\langle \mathbf{a}_0^{(p)},\mathbf{a}_1^{(p)}\right\rangle+\left\langle \mathbf{a}_{01}^{(p)},\mathbf{a}_{10}^{(p)}\right\rangle x_i^{-2}$$

which implies, since (x_1,x_2,x_t,x_4) are distinct with all but negligible probability, that

$$c^{(p)}=\left\langle \mathbf{a}_0^{(p)},\mathbf{a}_1^{(p)}\right\rangle$$

as desired. Note that the extractor runs in time $4^{\log(n)}=n^2$ and it is therefore efficient. □

C.2 Proof of Theorem 4.2

Proof. Let $n=2^i$ be the length of the vectors. We show this via an induction over i. For the base case ($n=1$), the prover simply sends $\mathbf{a}_0$ and $[\mathbf{a}_1]_1$ to the verifier. This is trivially witness-extended emulatable since $([\mathbf{a}_1]_1,\mathbf{a}_0)$ is given in clear. We now describe an algorithm that given black-box access to the prover extracts a valid witness for a given statement $([B_1\|\mathbf{b}]_2,[B_0]_t,[\mathbf{p}]_t)$, with all but negligible probability.

The algorithm obtains $([\mathbf{l}]_t,[\mathbf{r}]_t)$ by the prover and rewinds it with four uniformly sampled $(x_1,x_2,x_t,x_4)\leftarrow_{\$}\mathbb{Z}_q^4$. By induction hypothesis, there exists an efficient extractor that outputs $([\mathbf{a}_1^{(i)}]_1,\mathbf{a}_0^{(i)})$ such that

$$\begin{aligned}[\tilde{\mathbf{p}}]_t &=[\mathbf{l}]_t\cdot x_i^2+[\mathbf{p}]_t+[\mathbf{r}]_t\cdot x_i^{-2}\\ &=[\tilde{B}_0]_2[\mathbf{a}_1^{(i)}]_1+[\tilde{B}_1]_t\mathbf{a}_0^{(i)}+[\mathbf{b}]_2\left\langle [\mathbf{a}_1^{(i)}]_1,\mathbf{a}_0^{(i)}\right\rangle\\ &=\left([B_{10}]_2\cdot x_i^{-1}+[B_{11}]_2\cdot x_i\right)\cdot[\mathbf{a}_1^{(i)}]_1\\ &\quad+\left([B_{00}]_t\cdot x_i+[B_{01}]_t\cdot x_i^{-1}\right)\cdot\mathbf{a}_0^{(i)}+[\mathbf{b}]_2\left\langle [\mathbf{a}_1^{(i)}]_1,\mathbf{a}_0^{(i)}\right\rangle\end{aligned}$$

for all $i\in\{1,2,3,4\}$. Define (v_1,v_2,v_t) such that

$$\sum_{i=1}^{3}v_ix_i^2=1\qquad \sum_{i=1}^{3}v_i=0\qquad \sum_{i=1}^{3}v_ix_i^{-2}=0$$

and use them as the coefficients to compute, via a linear combination, $[\mathbf{a}_1^{(l)}]_1,\mathbf{a}_0^{(l)},[c^{(l)}]_1)$ such that

$$[\mathbf{l}]_t=[B_1]_2\cdot[\mathbf{a}_1^{(l)}]_1+[B_0]_t\cdot\mathbf{a}_0^{(l)}+[\mathbf{b}]_2\cdot[c^{(l)}]_1.$$

Use the same procedure to compute

$$\begin{aligned}[\mathbf{r}]_t &=[B_1]_2\cdot[\mathbf{a}_1^{(r)}]_1+[B_0]_t\cdot\mathbf{a}_0^{(r)}+[\mathbf{b}]_2\cdot[c^{(r)}]_1\ \text{ and}\\ [\mathbf{p}]_t &=[B_1]_2\cdot[\mathbf{a}_1^{(p)}]_1+[B_0]_t\cdot\mathbf{a}_0^{(p)}+[\mathbf{b}]_2\cdot[c^{(p)}]_1.\end{aligned}$$

Substituting with the equation above, we obtain for all $i\in\{1,2,3,4\}$ the following relation

$$\begin{aligned}&[\mathbf{l}]_t\cdot x_i^2+[\mathbf{p}]_t+[\mathbf{r}]_t\cdot x_i^{-2}\\ =&[B_1]_2\cdot([\mathbf{a}_1^{(l)}]_1x_i^2+[\mathbf{a}_1^{(r)}]_1x_i^{-2}+[\mathbf{a}_1^{(p)}]_1)\\ &+[B_0]_t\cdot(\mathbf{a}_0^{(l)}x_i^2+\mathbf{a}_0^{(r)}x_i^{-2}+\mathbf{a}_0^{(p)})\\ &+[\mathbf{b}]_2\cdot([c^{(l)}]_1x_i^2+[c^{(r)}]_1x_i^{-2}+[c^{(p)}]_1)\\ =&\left([B_{10}]_2\cdot x_i^{-1}+[B_{11}]_2\cdot x_i\right)\cdot[\mathbf{a}_1^{(i)}]_1\\ &+\left([B_{00}]_t\cdot x_i+[B_{01}]_t\cdot x_i^{-1}\right)\cdot\mathbf{a}_0^{(i)}+[\mathbf{b}]_2\left\langle[\mathbf{a}_1^{(i)}]_1,\mathbf{a}_0^{(i)}\right\rangle.\end{aligned}$$

Then the following equations

$$\begin{aligned}\mathbf{a}_0^{(i)}\cdot x_i&=\mathbf{a}_{00}^{(l)}x_i^2+\mathbf{a}_{00}^{(r)}x_i^{-2}+\mathbf{a}_{00}^{(p)}\\ \mathbf{a}_0^{(i)}\cdot x_i^{-1}&=\mathbf{a}_{01}^{(l)}x_i^2+\mathbf{a}_{01}^{(r)}x_i^{-2}+\mathbf{a}_{01}^{(p)}\\ [\mathbf{a}_1^{(i)}]_1\cdot x_i^{-1}&=[\mathbf{a}_{10}^{(l)}]_1x_i^2+[\mathbf{a}_{10}^{(r)}]_1x_i^{-2}+[\mathbf{a}_{10}^{(p)}]_1\\ [\mathbf{a}_1^{(i)}]_1\cdot x_i&=[\mathbf{a}_{11}^{(l)}]_1x_i^2+[\mathbf{a}_{11}^{(r)}]_1x_i^{-2}+[\mathbf{a}_{11}^{(p)}]_1\\ \left\langle[\mathbf{a}_1^{(i)}]_1,\mathbf{a}_0^{(i)}\right\rangle&=[c^{(l)}]_1x_i^2+[c^{(r)}]_1x_i^{-2}+[c^{(p)}]_1\end{aligned}$$

must hold unless a non-trivial discrete logarithm relation among the basis $[B_1\|\mathbf{b}]_2,[B_0]_t$ is revealed. Therefore, with all but negligible probability, we have that

$$\begin{aligned}&[\mathbf{a}_{10}^{(l)}]_1x_i^3+([\mathbf{a}_{10}^{(p)}]_1-[\mathbf{a}_{11}^{(l)}]_1)x_i\\ &+([\mathbf{a}_{10}^{(r)}]_1-[\mathbf{a}_{11}^{(p)}]_1)x_i^{-1}-[\mathbf{a}_{11}^{(r)}]_1x_i^{-3}=0\end{aligned}\tag{41}$$

and

$$\mathbf{a}_{01}^{(l)}x_i^3+(\mathbf{a}_{01}^{(p)}-\mathbf{a}_{00}^{(l)})x_i+(\mathbf{a}_{01}^{(r)}-\mathbf{a}_{00}^{(p)})x_i^{-1}-\mathbf{a}_{00}^{(r)}x_i^{-3}=0\tag{42}$$

Since (x_1,x_2,x_t,x_4) are distinct with all but negligible probability Equation 41 and 42 are satisfied only if all the coefficients are 0. This allows us to simplify the above system to

$$\begin{aligned}\mathbf{a}_0^{(i)}&=\mathbf{a}_{01}^{(p)}\cdot x_i+\mathbf{a}_{00}^{(p)}\cdot x_i^{-1}\\ [\mathbf{a}_1^{(i)}]_1&=[\mathbf{a}_{10}^{(p)}]_1\cdot x_i+[\mathbf{a}_{11}^{(p)}]_1\cdot x_i^{-1}\\ \left\langle[\mathbf{a}_1^{(i)}]_1,\mathbf{a}_0^{(i)}\right\rangle&=[c^{(l)}]_1x_i^2+[c^{(r)}]_1x_i^{-2}+[c^{(p)}]_1.\end{aligned}$$

Substituting we have

$$\begin{aligned}&[c^{(l)}]_1x_i^2+[c^{(r)}]_1x_i^{-2}+[c^{(p)}]_1\\ =&\left\langle[\mathbf{a}_{10}^{(p)}]_1,\mathbf{a}_{01}^{(p)}\right\rangle x_i^2+\left\langle[\mathbf{a}_0^{(p)}]_1,\mathbf{a}_1^{(p)}\right\rangle+\left\langle[\mathbf{a}_{11}^{(p)}]_1,\mathbf{a}_{00}^{(p)}\right\rangle x_i^{-2}\end{aligned}$$

which implies, since (x_1,x_2,x_t,x_4) are distinct with all but negligible probability, that

$$[c^{(p)}]_1=\left\langle[\mathbf{a}_1^{(p)}]_1,\mathbf{a}_0^{(p)}\right\rangle$$

as desired. Note that the extractor runs in time $4^{\log(n)}=n^2$ and it is therefore efficient. □

C.3 Proof of Theorem 4.3

Proof. Let $n=2^i$ be the length of the vectors. We show this via an induction over i. For the base case ($n=1$), the prover simply sends $([\mathbf{a}_1]_1,[\mathbf{a}_2]_2,\mathbf{a}_0)$ to the verifier. This is trivially witness-extended emulatable since the witness is given in clear. We now describe an algorithm that given black-box access to the prover extracts a valid witness for a given statement $([B_1]_2,[B_2]_1,[B_0]_t,[\mathbf{p}]_t,[\mathbf{a}_t]_t,[c]_t)$, with all but negligible probability.

The algorithm obtains $([\mathbf{l}]_t,[\mathbf{r}]_t,[c_L]_t,[c_R]_t)$ by the prover and rewinds it with four uniformly sampled $(x_1,x_2,x_t,x_4)\leftarrow_{\$}\mathbb{Z}_q^4$. By induction hypothesis, there exists an efficient extractor that outputs $([\mathbf{a}_1^{(i)}]_1,[\mathbf{a}_2^{(i)}]_2,\mathbf{a}_0^{(i)})$ such that

$$\begin{aligned}[\tilde{\mathbf{p}}]_t=&[\mathbf{l}]_t\cdot x_i^2+[\mathbf{p}]_t+[\mathbf{r}]_t\cdot x_i^{-2}\\ =&[\tilde{B}_0]_2[\mathbf{a}_1^{(i)}]_1+[\tilde{B}_1]_1[\mathbf{a}_2^{(i)}]_2+[\tilde{B}_2]_t\mathbf{a}_0^{(i)}\\ =&\left([B_{00}]_2\cdot x_i^{-1}+[B_{01}]_2\cdot x_i\right)\cdot[\mathbf{a}_1^{(i)}]_1\\ &+\left([B_{10}]_1\cdot x_i+[B_{11}]_1\cdot x_i^{-1}\right)\cdot[\mathbf{a}_2^{(i)}]_2\\ &+\left([B_{20}]_1\cdot x_i^{-1}+[B_{21}]_1\cdot x_i\right)\cdot\mathbf{a}_0^{(i)}\end{aligned}$$

and, for all $i\in\{1,2,3,4\}$,

$$\begin{aligned}[\tilde{c}]_t=&[c_L]_t\cdot x_i^2+[c]_t+[c_R]_t\cdot x_i^{-2}\\ =&\left\langle[\mathbf{a}_0^{(i)}]_1,[\mathbf{a}_1^{(i)}]_2\right\rangle+\left\langle\mathbf{a}_2^{(i)},[\tilde{\mathbf{a}}_t]_t\right\rangle\end{aligned}$$

where $[\tilde{\mathbf{a}}_t]_t=[\mathbf{a}_{t0}]_t\cdot x^{-1}+[\mathbf{a}_{t1}]_t\cdot x_i$. Define (v_1,v_2,v_t) such that

$$\sum_{i=1}^{3}v_ix_i^2=1\qquad\sum_{i=1}^{3}v_i=0\qquad\sum_{i=1}^{3}v_ix_i^{-2}=0$$

and use them as the coefficients to compute, via a linear combination, $([\mathbf{a}_1^{(l)}]_1,[\mathbf{a}_2^{(l)}]_2,\mathbf{a}_0^{(l)})$ such that

$$[\mathbf{l}]_t=[B_1]_2\cdot[\mathbf{a}_1^{(l)}]_1+[B_2]_1\cdot[\mathbf{a}_2^{(l)}]_2+[B_0]_t\cdot\mathbf{a}_0^{(l)}.$$

Use the same algorithm as above to compute $([\mathbf{a}_1^{(r)}]_1,[\mathbf{a}_2^{(r)}]_2,\mathbf{a}_0^{(r)})$ and $([\mathbf{a}_1^{(p)}]_1,[\mathbf{a}_2^{(p)}]_2,\mathbf{a}_0^{(p)})$ that satisfy the analogous constraint. Substituting with the equation above, we obtain for all $i\in\{1,2,3,4\}$ the following relation

$$\begin{aligned}&[\mathbf{l}]_t\cdot x_i^2+[\mathbf{p}]_t+[\mathbf{r}]_t\cdot x_i^{-2}\\ =&[B_1]_2\cdot([\mathbf{a}_1^{(l)}]_1x_i^2+[\mathbf{a}_1^{(r)}]_1x_i^{-2}+[\mathbf{a}_1^{(p)}]_1)\\ &+[B_2]_1\cdot([\mathbf{a}_2^{(l)}]_2x_i^2+[\mathbf{a}_2^{(r)}]_2x_i^{-2}+[\mathbf{a}_2^{(p)}]_2)\\ &+[B_0]_t\cdot(\mathbf{a}_0^{(l)}x_i^2+\mathbf{a}_0^{(r)}x_i^{-2}+\mathbf{a}_0^{(p)})\\ =&\left([B_{00}]_2\cdot x_i^{-1}+[B_{01}]_2\cdot x_i\right)\cdot[\mathbf{a}_1^{(i)}]_1\\ &+\left([B_{10}]_1\cdot x_i+[B_{11}]_1\cdot x_i^{-1}\right)\cdot[\mathbf{a}_2^{(i)}]_2\\ &+\left([B_{20}]_1\cdot x_i^{-1}+[B_{21}]_1\cdot x_i\right)\cdot\mathbf{a}_0^{(i)}.\end{aligned}$$

Then the following equations

$$\begin{aligned}[\mathbf{a}_1^{(i)}]_1\cdot x_i^{-1}&=[\mathbf{a}_{10}^{(l)}]_1x_i^2+[\mathbf{a}_{10}^{(r)}]_1x_i^{-2}+[\mathbf{a}_{10}^{(p)}]_1\\ [\mathbf{a}_1^{(i)}]_1\cdot x_i&=[\mathbf{a}_{11}^{(l)}]_1x_i^2+[\mathbf{a}_{11}^{(r)}]_1x_i^{-2}+[\mathbf{a}_{11}^{(p)}]_1\\ [\mathbf{a}_2^{(i)}]_2\cdot x_i&=[\mathbf{a}_{20}^{(l)}]_2x_i^2+[\mathbf{a}_{20}^{(r)}]_2x_i^{-2}+[\mathbf{a}_{20}^{(p)}]_2\\ [\mathbf{a}_2^{(i)}]_2\cdot x_i^{-1}&=[\mathbf{a}_{21}^{(l)}]_2x_i^2+[\mathbf{a}_{21}^{(r)}]_2x_i^{-2}+[\mathbf{a}_{21}^{(p)}]_2\\ \mathbf{a}_0^{(i)}\cdot x_i^{-1}&=\mathbf{a}_{00}^{(l)}x_i^2+\mathbf{a}_{00}^{(r)}x_i^{-2}+\mathbf{a}_{00}^{(p)}\\ \mathbf{a}_0^{(i)}\cdot x_i&=\mathbf{a}_{01}^{(l)}x_i^2+\mathbf{a}_{01}^{(r)}x_i^{-2}+\mathbf{a}_{01}^{(p)}\end{aligned}$$

must hold unless a non-trivial discrete logarithm relation among the basis $[B_1]_2,[B_2]_1,[B_0]_t$ is revealed. Therefore, with all but negligible probability, we have that

$$\begin{aligned}&[\mathbf{a}_{10}^{(l)}]_1x_i^3+([\mathbf{a}_{10}^{(p)}]_1-[\mathbf{a}_{11}^{(l)}]_1)x_i\\ &+([\mathbf{a}_{10}^{(r)}]_1-[\mathbf{a}_{11}^{(p)}]_1)x_i^{-1}-[\mathbf{a}_{11}^{(r)}]_1x_i^{-3}=0\end{aligned}\tag{43}$$

and

$$[\mathbf{a}_{21}^{(l)}]_2 x_i^3 + ([\mathbf{a}_{21}^{(p)}]_2 - [\mathbf{a}_{20}^{(l)}]_2) x_i \qquad (44)$$

$$+([\mathbf{a}_{21}^{(r)}]_2 - [\mathbf{a}_{20}^{(p)}]_2) x_i^{-1} - [\mathbf{a}_{20}^{(r)}]_2 x_i^{-3} = 0 \qquad (45)$$

and

$$\mathbf{a}_{00}^{(l)} x_i^3 + (\mathbf{a}_{00}^{(p)} - \mathbf{a}_{01}^{(l)}) x_i$$

$$+(\mathbf{a}_{00}^{(r)} - \mathbf{a}_{01}^{(p)}) x_i^{-1} - \mathbf{a}_{01}^{(r)} x_i^{-3} = 0. \qquad (46)$$

Since (x_1, x_2, x_t, x_4) are distinct with all but negligible probability Equation 43, 45, and 46 are satisfied only if all the coefficients are 0. This allows us to simplify the above system to

$$[\mathbf{a}_1^{(i)}]_1 = [\mathbf{a}_{10}^{(p)}]_1 \cdot x_i + [\mathbf{a}_{11}^{(p)}]_1 \cdot x_i^{-1}$$

$$[\mathbf{a}_2^{(i)}]_2 = [\mathbf{a}_{21}^{(p)}]_2 \cdot x_i + [\mathbf{a}_{20}^{(p)}]_2 \cdot x_i^{-1}$$

$$\mathbf{a}_0^{(i)} = \mathbf{a}_{00}^{(p)} \cdot x_i + \mathbf{a}_{01}^{(p)} \cdot x_i^{-1}.$$

Substituting we have

$$\begin{aligned}
[\tilde{c}]_t =& [c_L]_t \cdot x_i^2 + [c]_t + [c_R]_t \cdot x_i^{-2} \\
=& \left\langle [\mathbf{a}_1^{(i)}]_1, [\mathbf{a}_2^{(i)}]_2 \right\rangle + \left\langle \mathbf{a}_0^{(i)}, [\tilde{\mathbf{a}}_t]_t \right\rangle \\
=& \left\langle [\mathbf{a}_{10}^{(p)}]_1 \cdot x_i + [\mathbf{a}_{11}^{(p)}]_1 \cdot x_i^{-1}, [\mathbf{a}_{21}^{(p)}]_2 \cdot x_i + [\mathbf{a}_{20}^{(p)}]_2 \cdot x_i^{-1} \right\rangle \\
&+ \left\langle \mathbf{a}_{00}^{(p)} \cdot x_i + \mathbf{a}_{01}^{(p)} \cdot x_i^{-1}, [\mathbf{a}_{t1}]_t \cdot x_i + [\mathbf{a}_{t0}]_t \cdot x_i^{-1} \right\rangle \\
=& \left(\left\langle [\mathbf{a}_{10}^{(p)}]_1, [\mathbf{a}_{21}^{(p)}]_2 \right\rangle + \left\langle \mathbf{a}_{00}^{(p)}, [\mathbf{a}_{t1}]_t \right\rangle \right) \cdot x_i^2 \\
&+ \left\langle [\mathbf{a}_0^{(p)}]_1, [\mathbf{a}_1^{(p)}]_2 \right\rangle + \left\langle \mathbf{a}_2^{(p)}, [\mathbf{a}_t]_t \right\rangle \\
&+ \left(\left\langle [\mathbf{a}_{11}^{(p)}]_1, [\mathbf{a}_{20}^{(p)}]_2 \right\rangle + \left\langle \mathbf{a}_{01}^{(p)}, [\mathbf{a}_{t0}]_t \right\rangle \right) \cdot x_i^{-2}
\end{aligned}$$

which implies, since (x_1, x_2, x_t, x_4) are distinct with all but negligible probability, that

$$[c]_t = \left\langle [\mathbf{a}_0^{(p)}]_1, [\mathbf{a}_1^{(p)}]_2 \right\rangle + \left\langle \mathbf{a}_2^{(p)}, [\mathbf{a}_t]_t \right\rangle$$

as desired. Note that the extractor runs in time $4^{\log(n)} = n^2$ and it is therefore efficient. □