Succinct Verification of Compressed Sigma Protocols in the Updatable SRS setting

Moumita Dutta, Chaya Ganesh, and Neha Jawalkar

Indian Institute of Science, Bangalore, India
{moumitadutta,chaya,jawalkarp}@iisc.ac.in

Abstract. We propose protocols in the Compressed Sigma Protocol framework that achieve a succinct verifier. Towards this, we construct a new inner product argument and cast it in the Compressed Sigma Protocol (CSP) framework as a protocol for opening a committed linear form, achieving logarithmic verification.

We then use our succinct-verifier CSP to construct a zero-knowledge argument for circuit satisfiability (under the discrete logarithm assumption in bilinear groups) in the updatable Structured Reference String (SRS) setting that achieves $O(\log n)$ proof size and $O(\log n)$ verification complexity. Our circuit zero-knowledge protocol has concretely better proof/prover/verifier complexity compared to the the state-of-the-art protocol in the updatable setting under the same assumption. Our techniques of achieving verifier-succinctness in the compression framework is of independent interest.

We then show a commitment scheme for committing to group elements using a structured commitment key. We construct protocols to open a committed homomorphism on a committed vector with verifier succinctness in the designated verifier setting. This has applications in making the verifier in compressed sigma protocols for bilinear group arithmetic circuits, succinct.

1 Introduction

Zero-knowledge proof systems [GMR89] are an important primitive in theory of computation and a fundamental building block in various cryptographic constructions. In real-world applications, the proof size and efficiency of verification are crucial efficiency parameters. *Succinct* arguments, where the proof size is logarithmic in the size of the statement were first constructed by Kilian [Kil92] based on probabilistically checkable proofs (PCP). Micali's construction [Mic94] made this non-interactive in the random oracle model (ROM). Non-interactivity in the plain model is achieved by assuming a Common Reference String (CRS) generated during a setup phase. There has been a series of works on constructing (zero-knowledge) Succinct Non-interactive ARguments of Knowledge (zk-SNARKs) [Gro10, Lip12, BCI+13, GGPR13, PHGR13, Lip13, BCTV14, Gro16], which have very short proofs and admit efficient verification. The constructions with concretely better proof sizes are in the Structured Reference String (SRS) model and require a one-time setup or preprocessing that needs to be *trusted*. A line of work attempts to reduce the degree of trust in the setup phase by constructing SNARKS in an updatable setting [GKM+18, MBKM19, GWC19, CHM+20] where the SRS is *updatable*, meaning parties can continuously contribute to the randomness of the SRS, and the assumption is that at least one of the updates was honest. SNARKs that do not need a trusted setup and the verifier randomness consists of only public coins are called transparent.

Bulletproofs [BBB+18], building on the work of [BCC+16] introduced techniques that achieve logarithmic communication complexity in discrete logarithm (DL) based zero-knowledge proofs. The beautiful work of Attema and Cramer [AC20] introduced *compressed sigma protocol* theory by using a blackbox compression technique, which also places Bulletproofs in the framework of Sigma protocol theory. The idea of employing a compression mechanism on a pivot protocol has become a versatile tool leading to compressed sigma protocol theory for lattices [ACK21], and compressed sigma protocols for bilinear group arithmetic circuits [ACR21].

Compressed Sigma Protocols (CSP) are attractive in applications due to their reliance on weaker assumptions (DL), conceptual simplicity, logarithmic proof size and transparent setup. One downside of this class of protocols is that they are only proof-succinct but not verifier-succinct – the verification is linear. In this paper we study succinct verification of compressed sigma protocols while retaining most of their advantages.

1.1 Our Contributions

In this work, we present compressed sigma protocols that are both proof and verifier-succinct in the updatable SRS model. CSP compresses a pivot Sigma protocol for proving knowledge of a long vector $\mathbf{x}$ while revealing a public linear form $L(\mathbf{x})$, given a Pedersen commitment, resulting in a protocol for opening linear forms on committed vectors with proof size $O(\log n)$ and verification complexity $O(n)$ where n is the size of $\mathbf{x}$.

Protocol for opening a committed linear form. A core building block of our succinct verifier constructions is a protocol to open a committed linear form on a committed vector. It is an inner product argument, but in the spirit of CSP, one of the vectors – the linear form, is public and not a witness. This vector is committed to only for the sake of succinctness, and looking ahead, this commitment encodes the structure of the circuit and is computed during preprocessing. We use a commitment scheme with a structured key introduced by [DRZ20]. This protocol with logarithmic proof size and logarithmic verification complexity is secure under the DL assumption (same as CSP), albeit in a bilinear group and at the cost of moving to an updatable SRS setting. We compare our inner product protocol with [DRZ20] and [Lee21] in Table 1.

Succinct verifier protocol for circuit satisfiability. We construct a succinct argument of knowledge for circuit satisfiability in the universal updatable SRS model. The proof size and verifier is logarithmic in the size of the circuit. This is secure under DL and can be made non-interactive in the ROM using the Fiat-Shamir transform. We compare the concrete costs of our protocol, with that of [DRZ20] in Table 2[1]. Since we use the same structure of SRS, the complexity of updating the SRS and verifying the updates remain the same as in [DRZ20]. Dory's [Lee21] polynomial commitment can be used to obtain a protocol for circuit-satisfiability. The exact costs will depend on the underlying information-theoretic object (Polynomial interactive oracle proof) that is compiled using Dory. For univariate polynomials of degree n, and opening one evaluation, Dory's costs are: proof size of $(4\log n + 10)\mathbb{G} + (\log n + 8)\mathbb{Z}_q$, prover's computation of $(n + \log n)E + n^{1/2}P$, verifier's computation of $4\log nE + O(1)P$. Note that Dory's prover requires pairing operations and the security relies on a decisional assumption.

Protocol	Setup	Assumption	Proof size	$\mathcal{P}$ complexity	$\mathcal{V}$ complexity
[DRZ20]	Updatable	DL	$8\log n\ \mathbb{G}$ $2\log n\ \mathbb{Z}_q$	$(8n-4)\ E^*$	$2\log n\ P$ $4\log n\ E$
[Lee21]	Transparent	SXDH	$12\log n\ \mathbb{G}$ $\log n\ \mathbb{Z}_q$	$O(n^{1/2})$ P	$9\log n\ E$ $1\ P$
This work ($\Pi_{1\text{-}\mathcal{R}}$)	Updatable	DL	$4\log n\ \mathbb{G}$ $3\log n\ \mathbb{Z}_q$	$(8n-4)\ E^*$ $2\log n\ E$	$2\log n\ P$ $2\log n\ E$
This work ($\Pi_{2\text{-}\mathcal{R}}$)	Updatable	DL	$2\log n\ \mathbb{G}$ $\log n\ \mathbb{Z}_q$	$(4n-2)\ E^*$ $\log n\ E$	$\log n\ P$ $\log n\ E$

Table 1. Comparison of our Linear Form opening (or equivalently inner product arguments) protocols for vectors of length n. We compare in terms of most expensive operations, i.e. multi-exponentiations E, pairings P and exponentiations E^* (to provide aggregate values for non-constant multi-exponentiations).

Protocol	Proof size	$\mathcal{P}$ complexity	$\mathcal{V}$ complexity
[DRZ20]	$12\log n\,\mathbb{G} + 4\log n\,\mathbb{Z}_q$	$(22+10M)n\ E^*$	$12\log n\ E^* + 8\log n\ P$
[Lee21]	$(27/2)\log n\,\mathbb{G} + 3\log n\,\mathbb{Z}_q$	$O(n^{1/2})$ P	$((27/2)\log n + O(1))\ E + O(1)\ P$
This work (Π_{csat})	$2\log n\,\mathbb{G} + \log n\,\mathbb{Z}_q$	$(4+4M)n\ E^*$	$\log n\ E + \log n\ P$

Table 2. Circuit SAT protocol for a preprocessed circuit of size n (which is roughly $3m$ for m multiplication gates). Both protocols are updatable zkSNARKs that rely on the DL assumption. As in [DRZ20], we only compare in terms of the most expensive operations (exponentiations E and pairings P), and omit constant terms. M is a parameter that determines the processed circuit's fan-in and fan-out upper bound, and can be fine-tuned to balance the prover/verifier computations.

[1] We note that other SNARKs in the universal, updatable setting that have better communication and/verifier complexity ($O_\lambda(1)$) rely on the Algebraic Group Model or Knowledge Type assumptions *in addition* to the Random Oracle Model. In this work, we are interested in constructions in the Random Oracle Model, and relying on standard assumptions.

Protocol for opening a committed homomorphism. We then construct a protocol for opening a homomorphism, where both the vector and the homomorphism are committed to. A homomorphism is given by a vector of group elements. We extend commitment schemes to group elements from [LMR19, ACR21] to one that uses a structured key and show binding based on SXDH. Succinct verifier protocols for opening homomorphisms are useful in constructing proofs of partial knowledge with succinct verifier. We then extend our protocol to general homomorphisms (on commitments to $\mathbb{Z}_q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ elements simultaneously) motivated by applications to bilinear group arithmetic circuit zero-knowledge protocols. Our constructions for opening homomorphisms are in the *designated-verifier* setting. In applications like verification of structure preserving signatures and attribute-based authentication, public verifiability might not be necessary since there is a designated credential verifier, and indeed the homomorphism itself is given by the statement to prove (signature verification algorithm) that can be committed to in a preprocessing phase. Therefore, our protocols can be used in similar applications as in [LMR19], like proving knowledge of signature for complex access structures. While [LMR19] has proof that scales logarithmically with the size of the statement, our protocol additionally yields logarithmic verification (albeit for designated verifier, which we believe is not a limitation for credential verification).

1.2 Related Work

Daza et al [DRZ20] construct inner product arguments with logarithmic verifier by replacing the unstructured CRS or commitment key with a structured one. This also yields a protocol for circuit satisfiability with logarithmic verification in the updatable setting. Our construction and the protocols of [DRZ20] achieve the same result asymptotically. However, while we crucially rely on a structured commitment key to make the verifier logarithmic like in [DRZ20], our techniques are different. The work of [DRZ20] extend the protocols of [BCC+16, BBB+18], while we take the approach of CSP. This has the advantage of applying *compression mechanism* on standard protocols for linear relations (or non-linear relations after linearization). The CSP approach also allows us to extend our techniques to other applications where compression applies in a black-box way. Second, our protocols are concretely better than [DRZ20] with smaller constants (See Table 2).

Dory [Lee21] presents a transparent protocol for inner products between committed vectors with logarithmic proof size and logarithmic verification. Dory relies on a decisional assumption (SXDH) whereas our inner product protocol relies on DL. Additionally, our prover work is only group operations as opposed to $(O(n^{1/2}))$ pairing operations required by the prover in Dory, and our constants in the proof size are better.

Other SNARKs in the updatable setting [MBKM19, GWC19, CHM+20, LSZ23] rely on knowledge-type assumptions or Algebraic Group Model (AGM), and constructions in the transparent setting with similar and better asymptotics [BFS20, AGL+23] require unknown order groups with concretely expensive operations.

Lai et al [LMR19], show a generalization of Bulletproof's circuit zero-knowledge protocol to work for bilinear group arithmetic circuits directly, without requiring these circuits to be translated into arithmetic circuits. Attema et al [ACR21] generalize compressed sigma protocols for bilinear group arithmetic circuits. Both these constructions rely on a protocol for opening a group homomorphism where the verifier is linear. Using our protocol for opening a committed homomorphism will yield a succinct verifier at the expense of making it a designated verifier system. We provide the comparison in Table 4. Note that for application like Threshold Signature Schemes (following Algorithm 4 of [ACR21]), we retain the logarithmic size of the signature similar to prior works, however we improve the verification complexity from linear to logarithmic.

Performace Comparison for MiMC and Poseidon Hash. We report the timing using a third party implementation calculator `https://zka.lc`, where we estimate using BLS12-381 curve implemented in arkworks-rs provided using Amazon Linux 2 8-core Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz, 32GB. In Table 3, we use the reported number of gates for MiMC in [AGR+16], having 1293 multiplication gates and 646 addition gates. We achieve more than 3× improvement in prover time and 7× improvement in verification time as compared to Daza et al [DRZ20]. Similarly, we achieve more than 3× improvement in prover time and 7× improvement in verification time compared to Daza et al, for Poseidon (assuming number of R1CS constraints is equal to the number of multiplication gates, with 276 R1CS constraints [zkh]).

Similarly, we obtain more than 2× (2.5×) improvement in prover time as compared to Dory [Lee21] for MiMC hash instantiation and more than 4× (4.5×) improvement in prover time for Poseidon hash instantiation, at a slight cost of verifier time. Note that for circuits of smaller sizes, we do fairly better than Dory in prover time complexity, while not losing much in verifier time complexity. Also, for both comparisons we assume that Dory has atleast $6n^{1/2}$ pairings computation by prover, and 3 pairing checks performed by the verifier. For statements that show up in practice, like proving knowledge of opening of a Merkle tree leaf using MiMC/Poseidon Hash functions, the reported number for the prover increases by a factor of depth d of the tree.

Hash	Protocol	Prover Time	Verifier Time
MiMC	[DRZ20]	729,272	151,831
	[Lee21]	586,381	7,054
	Us	232,424	19,319
Poseidon	[DRZ20]	181,058	123,610
	[Lee21]	270, 912	6, 769
	Us	59,309	15,780

Table 3. Performance of MiMC and Poseidon Hash Instantiation using https://zka.lc and in μs.

Protocol	Proof size	$\mathcal{P}$ complexity	$\mathcal{V}$ complexity
[LMR19]	$O(n)\ \mathbb{G}_T$ $+\ O(\log n)\ \mathbb{G}_1$ $+\ O(\log n)\ \mathbb{G}_2$ $+\ O(\log n)\ \mathbb{Z}_q$	$O(n)\ E$	$O(n)\ E$
[ACR21]	$O(\log n)\ \mathbb{G}_T$ $+\ O(\log n)\ \mathbb{Z}_q$	$O(n)\ E$	$O(n)\ E$
Us	$O(\log n)\ \mathbb{G}_T$ $+\ O(\log n)\ \mathbb{G}_1$ $+\ O(\log n)\ \mathbb{G}_2$ $+\ O(\log n)\ \mathbb{Z}_q$	$O(n)\ E$	$O(\log n)\ E$ $+\ O(\log n)\ P$

Table 4. Comparison of protocols for opening homomorphism for vectors of length n. We compare in terms of most expensive operations, i.e. pairings P and exponentiations E and dominant communication cost with respect to elements of the field $\mathbb{Z}_q$ and groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$. Note that our verifier complexity is $2\log n\ E\ +\ \log n\ P$.

1.3 Technical Overview

The high level idea behind the inner product argument of [BCC+16] and the compressed sigma protocol of [AC20] is to compress a vector using a Pedersen commitment, and then in each round reduce the instance and the commitment key to another one of half the size by using the verifier's challenge. At a high level, the source of the verifier's linear complexity is in having to compute the new keys at every step.

We use a structured commitment key proposed in [DRZ20] that consists of encodings of multilinear monomials of a secret vector of logarithmic length, i.e., $\dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_\ell), \overline{\mathbf{a}} = \left(\prod_{i=1}^{\ell} \dot{a}_i^{b_i}\right)_{b_i \in \{0,1\}}$, for $n = 2^\ell$, the commitment key is $g^{\overline{\mathbf{a}}}$ where $g^{\mathbf{x}} = (g^{x_1}, \ldots, g^{x_n})$ for a vector $\mathbf{x} = (x_1, \ldots, x_n)$. The commitment to a vector $\mathbf{x}$ under key $g^{\overline{\mathbf{a}}}$ is $g^{\langle \overline{\mathbf{a}}, \mathbf{x} \rangle}$. This key is updatable, a party can sample new ℓ secrets and update the encoding in a verifiable way. A compressed version of this key, $g^{\dot{\mathbf{a}}} \in \mathbb{G}_1^\ell$ allows the verifier to be logarithmic.

Linear Form Opening with Succinct Verifier. We build on the inner product arguments of [BCC+16] and [DRZ20]. The verifier's work in [BCC+16] involves computing an updated key in each round, and in [DRZ20], the verifier is only given a compressed key (logarithmic) and the prover convinces the verifier that the reduced statement in each round is with respect to a new key that is correctly updated. New commitment keys with size half of the original one are created by splitting them in half and then combining them based on the verifier's challenge. A logarithmic verifier can check that a structured key has been updated correctly using a pairing operation.

While our protocol uses the same structured key, we take a slightly different approach: we exploit the fact that we can go from a commitment to a vector with respect to the second half of the original basis to a commitment to the same vector with respect to the first half of the original basis. Now, if a prover produces commitment to both halves of a vector with respect to the first half of the basis, the verifier can perform one multiplication in the exponent to check the consistency. Consider, $\dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_\ell), \overline{\mathbf{a}} = \left(\prod_{i=1}^{\ell} \dot{a}_i^{b_i}\right)_{b_i \in \{0,1\}}$, for $n = 2^\ell$, the commitment key is $g^{\overline{\mathbf{a}}} = \left(g^{\overline{\mathbf{a}}_L} \| g^{\dot{a}_\ell \overline{\mathbf{a}}_L}\right)$, the verification key is $H^{\dot{\mathbf{a}}}$, and $P = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x})$. In each step, for $\mathbf{x} = (\mathbf{x}_L \| \mathbf{x}_R)$, if the prover produces $A_1 = \mathrm{COM}_{\overline{\mathbf{a}}_L}(\mathbf{x}_L), A_2 = \mathrm{COM}_{\overline{\mathbf{a}}_L}(\mathbf{x}_R)$, the verifier can perform the check $e(\frac{P}{A_1}, H) = e(A_2, H^{\dot{a}_\ell})$. Thus, the commitment key updates are done implicitly by simply dropping off the last element $\dot{a}_\ell$, and using the challenge only to fold the instance vectors $\mathbf{x}' = \mathbf{x}_L + c\mathbf{x}_R$. This observation allows us to shave off about 4 group elements in each round from the Daza et al inner product argument. Our protocol also has the advantage of allowing efficient batching, i.e., for vectors of length n, the prover can prove, for distinct $L_1, \ldots, L_m$ and $\mathbf{x}_1, \ldots, \mathbf{x}_m$ that $L_1(\mathbf{x}_1) = y_1, \ldots, L_m(\mathbf{x}_m) = y_m$ while incurring a cost that is $\mathcal{O}(m + \log n)$ as opposed to $\mathcal{O}(m \log n)$.

Succinct ZK Argument for Circuit Satisfiability. We construct an improved protocol for arithmetic circuit satisfiability in the universal updatable SRS setting. The CSP approach to handle multiplication gates by linearizing them renders the verifier linear. We propose a protocol for computing a commitment to the linear form that captures the multiplication gates in the circuit in a verifiable way while keeping the verifier succinct. We use the ideas from [DRZ20] to preprocess a circuit, and obtain a commitment to the linear constraints. Now, all relations are linearized, we have commitments to all linear forms, and we show how to batch all linear form openings into one protocol for opening a committed linear form on a committed vector.

The following are two key new ideas to make a CSP-like proof have succinct verification. (i) The first relates to how we handle multiplication gates. For linearizing multiplication constraints, CSP uses a linear combination of polynomial evaluations at $1, \ldots, m$ to evaluate a polynomial at a new value z rendering the verifier linear. We handle multiplication in the same way as CSP, but instead of computing the public linear form for multiplication, the verifier instead *succinctly* verifies a commitment to it. We construct a succinct-verifier protocol for obtaining a *commitment* to the linear form used for verifying multiplication constraints. In order to do this, impose some structure; specifically, we use a linear combination of polynomial evaluations at $2, 2^2, \ldots, 2^m$. This choice allows us to compute the value of a polynomial at any point z while keeping the verifier succinct. This idea gives a protocol where the prover computes a commitment to the linear form that the verifier can efficiently check. The linearization is now via a committed linear form. (ii) The second idea relates to how we handle linear gates. Here, we employ the ideas of Daza et al, by reducing the problem of verifying linear gates to checking that two committed vectors are permutations of each other; and a Hadamard product product argument. We deviate from [DRZ20] by first reducing the permutation argument to the CSP pivot of opening linear forms on committed vectors. We then use our techniques from (i) to obtain commitments to these linear forms. Finally, we take advantage of our ability to batch the openings of linear forms, which allows us to prove circuit constraints while paying the cost of essentially a single invocation of our linear form protocol.

Homomorphism Opening with Succinct Verifier. Our ideas for succinct verifier in linear form openings do not extend to opening homomorphism. First, we need to commit to a homomorphism, and we extend the commitment scheme for group elements used in [LMR19, ACR21] to a commitment scheme with a structured key in order to make the verifier logarithmic. We show that binding is implied by SXDH (same assumption as the scheme with uniform key). Since we rely on SXDH, we cannot encode the commitment key randomness in the second group as the verification key. Thus a pairing check to verify correct key updates is not possible anymore, making our constructions designated verifier. A second hurdle is that the commitment itself lives in the target group. This means that our idea to check correct updation of the key in each round of split-and-fold (which involved a pairing operation) does not work anymore. We tackle this by having the commitment key in both $\mathbb{G}_1$ and $\mathbb{G}_T$. Now, the prover updates the commitment key in $\mathbb{G}_1$ and proves that this has been done correctly. The verifier can check this using a pairing, move this updated commitment key in $\mathbb{G}_1$ to $\mathbb{G}_T$, and then finally at the end of the recursion verify the commitment with respect to the updated key in $\mathbb{G}_T$.

2 Preliminaries

Notation. A finite field is denoted by $\mathbb{F}$. Let $\mathbb{G}$ be a group of order q. We denote by λ a security parameter, by negl a negligible function. For any integer $c > 0$, there exists $n \in \mathbb{N}$, such that $\forall\ x > n$, $\mathsf{negl}(x) \leq 1/n^c$. We denote vectors by boldface letters, and inner product between $\mathbf{a}$ and $\mathbf{b}$ by $\langle \mathbf{a}, \mathbf{b} \rangle$. We define $\mathcal{L}(\mathbb{Z}_q^n)$ as $\mathcal{L}(\mathbb{Z}_q^n) = \{L : L \text{ is a linear map from } \mathbb{Z}_q^n \text{ to } \mathbb{Z}_q\}$. A linear map $L \in \mathcal{L}(\mathbb{Z}_q^n)$ is given by a vector: $L(x_1, \ldots, x_n) = a_1 x_1 + \cdots + a_n x_n$ (for $(x_1, \ldots, x_n) \in \mathbb{Z}_q^n$) is given by $L = (a_1, \ldots, a_n)$ where $a_1, \ldots, a_n \in \mathbb{Z}_q$. For $x = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, its reversal is defined as $\mathsf{rev}(x) = (x_n, \ldots, x_1)$. A vector $\mathbf{a} = (a_1, \ldots, a_n)$ naturally defines a $(n-1)$-degree polynomial by considering the vector $\mathbf{a}$ as the vector of coefficients, which gives us the polynomial $\mathbf{a}(X) = a_1 + a_2 X + a_3 X^2 + \cdots + a_n X^{n-1}$. Also, a commitment to a polynomial $\mathbf{a}(X) = a_1 + a_2 X + a_3 X^2 + \cdots + a_n X^{n-1}$ is provided by a commitment to the vector of coefficient also denoted by $\mathbf{a} = (a_1, \ldots, a_n)$. Hence, for ease of notation, we use the vector and polynomial notation interchangeably throughout the paper.

For $\mathbf{g} = (g_1, \ldots, g_n) \in \mathbb{G}^n$ and $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, the multi-exponentiation $\mathbf{g}^{\mathbf{x}}$ is defined by $\mathbf{g}^{\mathbf{x}} = g_1^{x_1} \cdots g_n^{x_n}$. Also, for $g \in \mathbb{G}$ and $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, $g^{\mathbf{x}}$ is defined by $g^{\mathbf{x}} = (g^{x_1}, \cdots, g^{x_n})$. The inner product between elements of $\mathbb{Z}_q^n$, $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$ is denoted by $\langle \mathbf{a}, \mathbf{b} \rangle = a_1 b_1 + \cdots + a_n b_n$. For $a \in \mathbb{Z}_q^m, b \in \mathbb{Z}_q^n$, $a \| b \in \mathbb{Z}_q^{m+n}$ denotes concatenation of a and b in the respective order, similarly the notation is used for the vectors in a group $\mathbb{G}$ to denote concatenation of two vectors. For $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{G}^n$ and $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{G}^n$, the hadamard product $\mathbf{a} \circ \mathbf{b}$ is defined by $\mathbf{a} \circ \mathbf{b} = (a_1 b_1, \ldots, a_n b_n)$. Also, for $v, n \in \mathbb{Z}_q$, $\mathbf{v^n}$ denotes the vector $\mathbf{v^n} = (1, \ldots, v^n)$.

A bilinear group is denoted by the tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H) \leftarrow_R \mathcal{G}(1^\lambda)$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q, G and H are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a efficiently computable bilinear map.

We define $\mathcal{ML}_{2^\ell} \in \mathbb{Z}_q^n$ to be the set of all n-length multilinear vectors of the form $(1, \dot{a}_1, \dot{a}_2, \ldots, \dot{a}_1 \cdots \dot{a}_\ell)$, determined by ℓ-mutually independent scalars $\dot{a}_1, \ldots, \dot{a}_\ell$. We denote the set of ℓ scalars by $\dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_\ell)$ and the n-length vector by $\overline{\mathbf{a}} = (1, \dot{a}_1, \dot{a}_2, \ldots, \dot{a}_1 \cdots \dot{a}_\ell)$. More formally, $\mathcal{ML}_n = \{\overline{\mathbf{a}} : \dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_\ell) \leftarrow_R \mathbb{Z}_q^\ell,\ \overline{\mathbf{a}} = (\prod_{i=1}^{\ell} \dot{a}_i^{x_i})_{x_i \in \{0,1\}}\}$.

2.1 Interactive Arguments

We consider interactive arguments for relations, where a prover P convinces the verifier that it knows a witness w such that for a public statement x, $(x, w) \in \mathcal{R}$. For a pair of PPT interactive algorithms P, V, we denote by $\langle P(w), V \rangle(x)$, the output of V on its interaction with P where w is P's private input and x is a common input. Let $\mathcal{R} = \{(x, w)\}$, be a relation and $\mathcal{L}$ be the corresponding NP language.

Definition 1 (Argument of knowledge) *An interactive argument of knowledge(AoK) for a relation $\mathcal{R}$ consists of a PPT algorithm $\mathsf{setup}(1^\lambda)$ that takes a security parameter λ and outputs public parameters srs, and a pair of PPT interactive algorithms $\langle \mathcal{P}, \mathcal{V} \rangle$. The triple $(\mathsf{setup}, \mathcal{P}, \mathcal{V})$ satisfy the following properties.*

1. *Completeness. For all $\lambda \in \mathbb{N}$, $(x, w) \in \mathcal{R}$,*

$$\Pr\left(\langle \mathcal{P}(w) \mathcal{V} \rangle(\mathsf{srs}, x) = 1 \ : \ \mathsf{srs} \leftarrow \mathsf{setup}(1^\lambda)\right) = 1.$$

2. *Knowledge Soundness. An argument system $(\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ is knowledge sound with error κ if there exists an expected polynomial time extractor Ext such that for every efficient adversary $\tilde{\mathcal{P}}$, for every $x \in \{0,1\}^*$, whenever $\tilde{\mathcal{P}}$ makes $\mathcal{V}$ accept with probability $\epsilon > \kappa$, $\mathsf{Ext}^{\tilde{\mathcal{P}}}(x)$ outputs w^* such that $(x, w^*) \in \mathcal{R}$ with probability at least $\frac{\epsilon - \kappa}{q}$ for some polynomial q.*

Definition 2 (Honest verifier zero-knowledge (HVZK)) *An argument system $(\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ is HVZK if there exists an efficient simulator $\mathcal{S}$ such that for every $(x, w) \in \mathcal{R}$, the distribution $\mathcal{S}(x)$ is identical to $\mathsf{View}_{\langle \mathcal{P}(x,w), \mathcal{V}(x) \rangle}$, where $\mathsf{View}_{\langle \mathcal{P}(x,w), \mathcal{V}(x) \rangle}$ is the distribution of the view of the verifier in the protocol on common input x and prover's witness w.*

We now recall the special soundness property which is typically simpler than knowledge soundness.

Definition 3 (Tree of transcripts) *A set of $k = \prod_{i=1}^{\ell} k_i$ accepting transcripts for an argument system $(\mathcal{P}, \mathcal{V})$ is a $(k_1, \ldots, k_\ell)$-tree of accepting transcripts if they are in the following tree structure: The nodes of the tree are formed by $\mathcal{P}$'s messages, and the edges correspond to $\mathcal{V}$'s messages. Each node at depth i has exactly k_i children corresponding to k_i distinct messages from the verifier. Each transcript is given by a path from a leaf node to the root.*

Definition 4 (Special Soundness) *A $(2\ell + 1)$ move protocol is said to be $(k_1, \ldots, k_\ell)$ special sound if there exists an extractor* Ext *that given a $(k_1, \ldots, k_\ell)$-tree of accepting transcripts for an instance x, outputs w such that $(x, w) \in \mathcal{R}$.*

Definition 5 (Succinct Argument of knowledge) *An argument system is* proof-succinct *if the communication complexity between prover and verifier is bounded by $\mathsf{poly}(\lambda)$, and* verifier-succinct *if the running time of $\mathcal{V}$ is bounded by $\mathsf{poly}(\lambda + |x|)$ and independent of the size of the circuit computing $\mathcal{R}$.*

Fiat-Shamir and Non-interactive AoK. An argument system is said to be *public coin* if the verifier's messages are uniformly random strings. Public coin interactive protocols can be heuristically compiled into non-interactive arguments by applying the Fiat-Shamir [FS87] transform (FS) in the Random Oracle Model (ROM). Since our protocols are all public coin, we show special soundness of the interactive version and then rely on FS.

Updatable SRS SNARK. Introduced by Groth at al. [GKM+18], the updatable universal structured reference string (SRS) enables parties to update the parameters, while retaining computational soundness against any probabilistic-polynomial time adversary, as long as at least one honest update is made. We follow the model used in Daza et al. [DRZ20] based on [GKM+18], where anyone can deterministically compute the circuit-specific preprocessing material given the (updated) universal SRS, which ensures that the circuit-specific preprocessing is performed publicly without any involvement of secrets.

2.2 Assumptions

Definition 6 (DLOG Assumption) *The discrete logarithm (DLOG) assumption for a group $\mathbb{G}$ states that, given a generator g of the group $\mathbb{G}$, for all PPT adversaries $\mathcal{A}$ we have*

$$\Pr\left(r = r' \mid r \leftarrow_R \mathbb{Z}_q \wedge r' \leftarrow \mathcal{A}(g^r)\right) \leq \mathsf{negl}(\lambda)$$

When the commitment key is structured, we need the following Find-rep assumption to hold in bilinear groups, which is known to follow from DLOG, as shown in [DRZ20].

Definition 7 (Find-rep Assumption) *Find-rep assumption holds with respct to a a bilinear group generator $\mathcal{G}$ for all PPT adversaries $\mathcal{A}$ we have*

$$\Pr\left(\begin{array}{c} g^{\langle \mathbf{a}, \mathbf{x} \rangle} = 1_{\mathbb{G}} \wedge \mathbf{x} \neq 0 \\ (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow_R \mathcal{G}(1^\lambda) \\ \mathbf{a} \leftarrow_R \mathcal{ML}_n, \mathbf{x} \leftarrow \mathcal{A}(g^{\mathbf{a}}, h^{\mathbf{a}}) \end{array}\right) \leq \mathsf{negl}(\lambda)$$

Definition 8 (DDH Assumption) *For a group $\mathbb{G}$, decisional Diffie-Hellman (DDH) problem is to determine, when given a tuple (g, g^a, g^b, g^c) for some $g \in \mathbb{G}$, whether $c = ab$. Decisional Diffie-Hellman (DDH) assumption in a group $\mathbb{G}$ states that DDH problem is hard in that group.*

Definition 9 (SXDH Assumption) *For $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H) \leftarrow_R \mathcal{G}(1^\lambda)$, the Symmetric External Diffie-Hellman (SXDH) assumption states that the decisional Diffie-Hellman (DDH) assumption holds for both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

3 CSP for Committed Linear Forms

We construct a protocol to reveal $L(x)$ for a committed vector x, and committed linear form L. The idea is to honestly generate a commitment to the linear form in a preprocesing phase. Once generated, a commitment to a linear form L can be used to open L on any committed vector. When used as a subprotocol for arithmetic circuit SAT, we generate these commitments during a one-time circuit-specific setup phase.

Definition 1 (Commitment to $\mathbb{Z}_q$-vectors([DRZ20])). *Let $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, H)$ be a bilinear group and let $n \geq 0$. We define a commitment scheme for vectors in $\mathbb{Z}_q^n$ with the following setup and commitment phase:*

- *Setup: Let $\dot{\boldsymbol{a}} := (\dot{a}_1, \ldots, \dot{a}_\ell) \leftarrow \mathbb{Z}_q^\ell$ where $\ell = \log(n+1)$. Let $\overline{\boldsymbol{a}} = (a_1, \ldots, a_n) \in \mathbb{Z}_q^{n+1}$ be defined as $a_j = \prod_{i=1}^{\ell} \dot{a}_i^{b_{ji}}$,where $(b_{j1}, \ldots, b_{j\ell})$ is the binary representation of j. Output $(g^{\overline{\boldsymbol{a}}}, H^{\dot{\boldsymbol{a}}})$, where $g^{\overline{\boldsymbol{a}}} \in \mathbb{G}_1^{n+1}$ is the commitment key, and $H^{\dot{\boldsymbol{a}}} \in \mathbb{G}_2^{\ell}$ is the verification key.*
- *Commit:* $\mathrm{COM} : \mathbb{Z}_q^{n+1} \to \mathbb{G}_1$, $\gamma \leftarrow_R \mathbb{Z}_q$ *and define*
  - $\mathrm{COM}_{\overline{\boldsymbol{a}}}(\mathbf{x}; \gamma) \to g^{\langle \overline{\mathbf{a}}, (\mathbf{x} \| \gamma) \rangle}$

Lemma 1. *The above scheme is perfectly hiding and computationally binding under the DLOG assumption [DRZ20].*

The proof is provided in Section A.1 of the Supplementary Material.

We start with a Σ-Protocol for opening a linear form which is similar to the initial protocol in [AC20] but using structured keys instead of uniformly random keys for the commitments. We consider the following relation

$$\mathcal{R} = \{(P \in \mathbb{G}, L \in \mathcal{L}(\mathbb{Z}_q^n), y \in \mathbb{Z}_q; \mathbf{x} \in \mathbb{Z}_q^n, \gamma \in \mathbb{Z}_q) : P = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}; \gamma) \ \wedge \ L(\mathbf{x}) = y\}\}$$

which corresponds to showing opening of a public commitment P and a public value y, obtained by operating a linear form L on a secret $\mathbb{Z}_q^n$ vector $\mathbf{x}$. This is the same relation as in [AC20] but using the commitment COM with structured commitment key $(g^{\overline{\mathbf{a}}}, H^{\dot{\mathbf{a}}})$ (Definition 1). We rely on the SXDH assumption for providing the structured key $(g^{\overline{\mathbf{a}}}, H^{\dot{\mathbf{a}}})$ while maintaining security.

Parameters

- Common parameters : $(P \in \mathbb{G}, L \in \mathcal{L}(\mathbb{Z}_q^n), y \in \mathbb{Z}_q)$, $P = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}; \gamma), y = L(\mathbf{x})$
- $\mathcal{P}$'s input : $(\mathbf{x} \in \mathbb{Z}_q^n, \gamma \in \mathbb{Z}_q)$

Protocol

1. $\mathcal{P}$ samples $\mathbf{r} \leftarrow_R \mathbb{Z}_q^n, \rho \leftarrow_R \mathbb{Z}_q$, computes $A = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{r}; \rho)$, $t = L(\mathbf{r})$ and sends A, t to $\mathcal{V}$
2. $\mathcal{V}$ samples $c \leftarrow_R \mathbb{Z}_q$ and sends c to $\mathcal{P}$
3. $\mathcal{P}$ computes $\mathbf{z} = c\mathbf{x} + \mathbf{r}$ and $\phi = c\gamma + \rho$ and sends $\mathbf{z}, \phi$ to $\mathcal{V}$
4. $\mathcal{V}$ checks if $\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{z}; \phi) = AP^c$ and $L(\mathbf{z}) = cy + t$, outputs 1 if it holds, outputs 0 otherwise.

Fig. 1: Protocol Π_0 for relation $\mathcal{R}$

Theorem 1. *Π_0 is a 3-move protocol for relation $\mathcal{R}$. It is perfectly complete, special honest-verifier zero-knowledge and computationally special sound.*

Proof. **Completeness.** If protocol steps by the prover is executed correctly, then we have $\mathbf{z} = c\mathbf{x} + \mathbf{r}$, and it satisfies the final two checks by the verifier

$$\begin{aligned} \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{z}; \phi) &= g^{\langle \overline{\mathbf{a}}, (\mathbf{z} \| \phi) \rangle}, \\ &= g^{\langle \overline{\mathbf{a}}, (c\mathbf{x} + \mathbf{r} \| c\gamma + \rho) \rangle} \\ &= g^{c\langle \overline{\mathbf{a}}, (\mathbf{x} \| \gamma) \rangle} g^{\langle \overline{\mathbf{a}}, (\mathbf{r} \| \rho) \rangle} \\ &= P^c A \end{aligned} \qquad \begin{aligned} L(\mathbf{z}) &= L(c\mathbf{x} + \mathbf{r}) \\ &= cL(\mathbf{x}) + L(\mathbf{r}) \\ &= cy + t \end{aligned}$$

Special Honest-Verifier Zero-Knowledge. We construct a simulator $\mathcal{S}$, which produces a transcript indistinguishable from the transcript of the real execution of the protocol, provided a challenge $c \in \mathbb{Z}_q$. (i) $\mathcal{S}$ samples $\mathbf{z}, \phi$ (ii) $\mathcal{S}$ computes $\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{z}; \phi)$ and sets $A = \frac{\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{z}; \phi)}{P^c}$ and $t = L(\mathbf{z}) - cy$.

The transcript produced by the simulator $\mathcal{S}$ is indistinguishable from the transcript of the real execution of the protocol due to the hiding property of the commitment scheme $\mathrm{COM}_{(.)}(.)$, which ensures that a

commitment sampled uniformly at random from the set of all possible commitments is indistinguishable from a commitment computed from a message chosen uniformly at random.

Special Soundness. We consider 2 accepting transcripts $(A, t, c_1, \mathbf{z}_1, \phi_1)$ and $(A, t, c_2, \mathbf{z}_2, \phi_2)$, such that $c_1 \neq c_2$. Then we have,

$$\begin{aligned} &\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{z}_1; \phi_1) = AP^{c_1}, L(\mathbf{z}_1) = c_1 y + t, && \text{and} \\ &\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{z}_2; \phi_2) = AP^{c_2}, L(\mathbf{z}_2) = c_2 y + t \\ \implies &g^{\langle \overline{\mathbf{a}}, (\mathbf{z}_1 \| \phi_1) \rangle} = AP^{c_1}, L(\mathbf{z}_1) = c_1 y + t, && \text{and} \\ &g^{\langle \overline{\mathbf{a}}, (\mathbf{z}_2 \| \phi_2) \rangle} = AP^{c_2}, L(\mathbf{z}_2) = c_2 y + t \end{aligned}$$

Dividing the first two equations, and subtracting the second equations, we get

$$g^{\langle \overline{\mathbf{a}}, (\mathbf{z}_1 - \mathbf{z}_2 \| \phi_1 - \phi_2) \rangle} = P^{c_1 - c_2}, L(\mathbf{z}_1 - \mathbf{z}_2) = (c_1 - c_2) y$$

We define $\mathbf{x} = (\mathbf{z}_1 - \mathbf{z}_2)/(c_1 - c_2)$ and $\gamma = (\phi_1 - \phi_2)/(c_1 - c_2)$, and this gives us $g^{\langle \overline{\mathbf{a}}, (\mathbf{x}, \gamma) \rangle} = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}, \gamma) = P$, and $L(\mathbf{x}) = y$.

3.1 Opening a Committed Linear Form

In Π_0, the communication complexity as well as the verifier complexity is linear due to the last message sent by the prover and the last check performed by the verifier. To improve both complexities, we replace the message sent in the last step of Π_0 with a proof of knowledge. We define a relation that captures this and reduce the verifier's work by committing to the linear form and compressing the check using split-and-fold technique used in [AC20]. The protocol Π_1 is in Fig 2. We compress recursively until the size of instance is constant and can be sent in the clear.

We now consider the new relation $\mathcal{R}_{\mathrm{CLF}}$ with respect to an updated linear form, where the new linear form L is defined as $L(\mathbf{z}, \phi) := L(\mathbf{z})$ and hence, the check performed by the verifier in step 5 of Π_0 (Fig 1) corresponds to the new relation, where the message sent by the prover $\mathcal{P}$ to the verifier $\mathcal{V}$ in step 4 corresponds to a witness in the new relation.

$$\begin{aligned} \mathcal{R}_{\mathrm{CLF}} = \{(P \in \mathbb{G}, Q \in \mathbb{G}, y \in \mathbb{Z}_q; \mathbf{x} \in \mathbb{Z}_q^n, L \in \mathcal{L}(\mathbb{Z}_q^n)) : \\ P = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}) \ \wedge \ Q = \mathrm{COM}_{\overline{\mathbf{a}}}(L) \ \wedge \ L(\mathbf{x}) = y\}\} \end{aligned}$$

This corresponds to showing opening of a public commitment P and a public value y, which is the output of a linear form L on a secret $\mathbb{Z}_q^n$ vector $\mathbf{x}$, given a commitment to the linear form L. We present the Σ-Protocol Π_1 for $\mathcal{R}_{\mathrm{CLF}}$ in Fig 2, and use this protocol instead of step 5 of Π_0 to improve the communication and verifier complexity.

Finally, we define $\Pi_{1\text{-}\mathcal{R}}$ as $\Pi_{1\text{-}\mathcal{R}} = \Pi_1 \circ \Pi_0$ for relation $\mathcal{R}$, whose communication and computational complexity are dominated by that of Π_1. The concatenation of the protocols Π_1 and Π_0 proceeds by replacing the last message sent in clear by the prover in Π_0 with a proof of knowledge using Π_1.

Theorem 2. *Π_1 is a $(k_1, \ldots, k_\ell)$-move protocol for relation $\mathcal{R}_{CLF}$, where $k_i = 3$, $\forall i \in [\ell], \ell = \log n$. It is perfectly complete and computationally special sound. It incurs total communication of $4 \log n$ group elements and $4 + 3 \log n$ field elements.*

Proof Sketch. Here we present the proof sketch for the special soundness of Π_1. Given 3 accepting transcripts $(A_1, A_2, B_1, B_2, y_1, y_2, c_i, \mathbf{x}_i', L_i')$ for one iteration of Π_1 (where one iteration consists of steps 1-5, and step 6 follows by sending $\mathbf{x}', L'$ instead of providing a PoK) for three distinct challenges c_1, c_2 and c_3, extractor proceeds as follows. It computes a_1, a_2, a_3 as $(a_1, a_2, a_3)^T = V^{-1}(0, 1, 0)^T$, where V is the Vandermonde matrix defined by the the challenges, and sets $\mathbf{w} = \sum_i a_i (c_i \mathbf{x}_i' \| \mathbf{x}_i')$ to be the extracted value. We show that $\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{w}) = P$; and similarly we extract a valid opening $\mathbf{m}$ of the commitment Q.

We then show that the extracted $\mathbf{w}, \mathbf{m}$ satisfy $\mathbf{x}_i' = \mathbf{w}_L + c_i \mathbf{w}_R$ and $L_i' = \mathbf{m}_L + c_i \mathbf{m}_R$ for all $i = 1, 2, 3$, which when substituted in the verification equation $\langle L_i', \mathbf{x}_i' \rangle = y_i'$ (Step 7(b)) gives us $\langle \mathbf{m}_R, \mathbf{w}_L \rangle + c_i \langle \mathbf{m}, \mathbf{w} \rangle + c_i^2 \langle \mathbf{m}_L, \mathbf{w}_R \rangle = y_1 + c_i y + c_i^2 y_2$, for the distinct challenges c_1, c_2 and c_3. Hence, $\langle \mathbf{m}, \mathbf{w} \rangle = y$ holds, which shows that $(\mathbf{w}, \mathbf{m})$ is a valid witness for $(P, Q, y) \in \mathcal{R}_{\mathrm{CLF}}$. We present the full proof in Section B.1 of the Supplementary Material.

Parameters

- **Common parameters** : $(P \in \mathbb{G}, Q \in \mathbb{G}, y \in \mathbb{Z}_q, H^{\dot{\mathbf{a}}} \in \mathbb{G}_2^\ell)$,
  - $P = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}), Q = \mathrm{COM}_{\overline{\mathbf{a}}}(L), y = L(\mathbf{x})$,
  - $n = 2^\ell, \dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_\ell), \overline{\mathbf{a}} = \left(\prod_{i=1}^{\ell} \dot{a}_i^{b_i}\right)_{b_i \in \{0,1\}}$
  - A bilinear group description $(q, \mathbb{G}, \mathbb{G}_2, \mathbb{G}_T, e, g, H)$, where $e : \mathbb{G} \times \mathbb{G}_2 \mapsto \mathbb{G}_T$ is an efficient bilinear map and g, H and $e(g, H)$ are generators of groups $\mathbb{G}, \mathbb{G}_2$ and $\mathbb{G}_T$, respectively, each of order q.
- $\mathcal{P}$**'s input** : $(g^{\overline{\mathbf{a}}} \in \mathbb{G}^n, \mathbf{x} \in \mathbb{Z}_q^n, L \in \mathcal{L}(\mathbb{Z}_q^n))$

Protocol

1. $\mathcal{P}$ parses $\mathbf{x} = (\mathbf{x}_L \| \mathbf{x}_R), L = (L_L \| L_R)$ and $g^{\overline{\mathbf{a}}} = \left(g^{\overline{\mathbf{a}}_L} \| g^{\dot{a}_\ell \overline{\mathbf{a}}_L}\right)$, and computes and sends the following to $\mathcal{V}$:
   (a) $A_1 = \mathrm{COM}_{\overline{\mathbf{a}}_L}(\mathbf{x}_L), A_2 = \mathrm{COM}_{\overline{\mathbf{a}}_L}(\mathbf{x}_R)$
   (b) $B_1 = \mathrm{COM}_{\overline{\mathbf{a}}_L}(L_L), B_2 = \mathrm{COM}_{\overline{\mathbf{a}}_L}(L_R)$
   (c) $y_1 = \langle L_R, \mathbf{x}_L \rangle, y_2 = \langle L_L, \mathbf{x}_R \rangle$
2. $\mathcal{V}$ checks the following, proceeds to step 3 if it holds, and aborts otherwise
$$e\left(\frac{P}{A_1}, H\right) = e\left(A_2, H^{\dot{a}_\ell}\right) \ \wedge \ e\left(\frac{Q}{B_1}, H\right) = e\left(B_2, H^{\dot{a}_\ell}\right)$$
3. $\mathcal{V}$ samples $c \leftarrow_R \mathbb{Z}_q$ and sends c to $\mathcal{P}$
4. $\mathcal{P}$ sets $\mathbf{x}' = \mathbf{x}_L + c\mathbf{x}_R$, $L' = cL_L + L_R$ and implicitly sets $\dot{\mathbf{a}}' = (\dot{a}_1, \ldots, \dot{a}_{\ell-1})$ and $\overline{\mathbf{a}}' = \overline{\mathbf{a}}_L$.
5. $\mathcal{P}$ and $\mathcal{V}$ both compute the following :
$$P' = A_1 A_2^c, \ Q' = B_1^c B_2, \ y' = y_1 + cy + c^2 y_2$$
6. If $\mathbf{x}' \notin \mathbb{Z}_q^2$: $\mathcal{P}$ runs PoK Π_1 to prove knowledge of $\mathbf{x}', L'$ such that $\mathrm{COM}_{\overline{\mathbf{a}}'}(\mathbf{x}') = P'$, $\mathrm{COM}_{\overline{\mathbf{a}}'}(L') = Q'$ and $\langle L', \mathbf{x}' \rangle = y'$.
   Hence, $\mathcal{P}$ and $\mathcal{V}$ runs the protocol Π_1 with updated common parameters $(P', Q', y', g^{\dot{\mathbf{a}}'})$ and prover's input $(g^{\overline{\mathbf{a}}'}, \mathbf{x}', L')$, for $(P', Q', y'; \mathbf{x}') \in \mathcal{R}_{\mathrm{CLF}}$
7. If $\mathbf{x}' \in \mathbb{Z}_q^2$:
   (a) $\mathcal{P}$ sends $\mathbf{x}', L'$ to $\mathcal{V}$
   (b) $\mathcal{V}$ outputs 1 if the following checks hold, and 0 otherwise:
$$\mathrm{COM}_{\overline{\mathbf{a}}'}(\mathbf{x}') = P' \ \wedge \ \mathrm{COM}_{\overline{\mathbf{a}}'}(L') = Q' \ \wedge \ \langle L', \mathbf{x}' \rangle = y'$$

Fig. 2: Protocol Π_1 for relation $\mathcal{R}_{\mathrm{CLF}}$

3.2 Improved Protocol for Opening a Committed Linear Form

We recall that for $x = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, $\mathsf{rev}(x)$ is defined as $\mathsf{rev}(x) = (x_n, \ldots, x_1)$. We present an alternative protocol that achieves better communication complexity at the cost of degrading soundness; it needs $2n$ transcripts to extract. Consider a modified version of the relation $\mathcal{R}_{\mathrm{CLF}}$ defined earlier, where instead of committing to the linear form we now commit to the reverse of the linear form, and define the new relation $\mathcal{R}_{\text{CLF-rev}}$ as follows :

$$\mathcal{R}_{\text{CLF-rev}} = \{(P \in \mathbb{G}, Q \in \mathbb{G}, y \in \mathbb{Z}_q; \mathbf{x} \in \mathbb{Z}_q^n, L \in \mathcal{L}(\mathbb{Z}_q^n)) : \\ P = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}) \ \wedge \ Q = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathsf{rev}(L)) \ \wedge \ L(\mathbf{x}) = y\}\}$$

where the relation $\mathcal{R}_{\text{CLF-rev}}$ corresponds to showing opening of a public commitment P and a public value y, obtained by operating a linear form L on a secret $\mathbb{Z}_q^n$ vector $\mathbf{x}$, where we also have a commitment to the reverse of linear form L which is represented as a vector. We note that the randomness used for the commitments are implicitly assumed from here onwards. We have the following protocol for the relation $\mathcal{R}_{\text{CLF-rev}}$ (Fig 3).

The protocol aims to reduce the verification of the statement $(P, Q, y; \mathbf{x}) \in \mathcal{R}_{\text{CLF-rev}}$ by prover $\mathcal{P}$ and verifier $\mathcal{V}$, to a polynomial check where we have the equation

$$\mathbf{x}(U) \cdot \mathsf{rev}(L)(U) = p_L(U) \cdot U^{-1} + y \cdot U^{n-1} + p_R(U) \cdot U^n$$

Parameters

- **Common parameters** : $(P \in \mathbb{G}, Q \in \mathbb{G}, y \in \mathbb{Z}_q, H^{\dot{\mathbf{a}}} \in \mathbb{G}^\ell)$,
  - $P = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}), Q = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathsf{rev}(L)), y = L(\mathbf{x})$,
  - $n = 2^\ell, \dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_n), \overline{\mathbf{a}} = \left(\prod_{i=1}^{\ell} \dot{a}_i^{b_i}\right)_{b_i \in \{0,1\}}$
- $\mathcal{P}$**'s input** : $(g^{\overline{\mathbf{a}}} \in \mathbb{G}^n, \mathbf{x} \in \mathbb{Z}_q^n, L \in \mathcal{L}(\mathbb{Z}_q^n))$

Protocol

1. Let us define $\mathbf{B} \in \mathbb{Z}_q^n$ as $\mathbf{B} = \mathsf{rev}(L)$. Let $\mathbf{x}(U)$ be a polynomial of degree $(n-1)$ defined with coefficient vector $\mathbf{x} = (x_1, \ldots, x_n)$, such that $\mathbf{x}(U) = \sum_{i=0}^{n-1} x_{i+1} U^i$. Similarly, we define the polynomial $\mathbf{B}(U)$ of degree $(n-1)$ for the vector $\mathbf{B}$.
2. $\mathcal{P}$ defines a $(2n-2)$ degree polynomial $\boldsymbol{p}$ by
$$\boldsymbol{p}(U) = \mathbf{x}(U) \cdot \mathbf{B}(U) = \sum_{i,j} x_{i+1} B_{j+1} U^{i+j},$$
and parses the computed polynomial as
$$\boldsymbol{p}(U) = \boldsymbol{p}_L(U) \cdot U^{-1} + y \cdot U^{n-1} + \boldsymbol{p}_R(U) \cdot U^n,$$
where p_L is a polynomial of degree $(n-1)$ and p_L is a polynomial of degree $(n-2)$ (which is trivially extended to a vector of length n by appending 0 appropriately).
3. $\mathcal{P}$ computes $A_1 = \mathrm{COM}_{\overline{\mathbf{a}}}(\boldsymbol{p}_L)$ and $A_2 = \mathrm{COM}_{\overline{\mathbf{a}}}(\boldsymbol{p}_R)$, and sends A_1, A_2 to $\mathcal{V}$
4. $\mathcal{V}$ samples $c \leftarrow_R \mathbb{Z}_q$ and sends c to $\mathcal{P}$
5. $\mathcal{P}$ computes the evaluations of the polynomials on the random challenge c as follows, and then sends z_1, z_2, z_3 and z_4 to $\mathcal{V}$: $z_1 = \mathbf{x}(c)$, $z_2 = \mathbf{B}(c)$, $z_3 = p_L(c)$, $z_4 = p_R(c)$.
6. $\mathcal{V}$ checks if the following relation holds, aborts if the check fails and continues to the next step otherwise.
$$z_3 \cdot c^{-1} + y \cdot c^{n-1} + z_4 \cdot c^n = z_1 \cdot z_2$$
7. $\mathcal{V}$ samples $t \leftarrow_R \mathbb{Z}_q$ and sends t to $\mathcal{P}$
8. $\mathcal{P}$ sets $\mathbf{w} = \mathbf{x} + t \cdot \mathbf{B} + t^2 \cdot \boldsymbol{p}_L + t^3 \cdot \boldsymbol{p}_R$ and sends $\mathbf{w}$ to $\mathcal{V}$
9. $\mathcal{P}$ and $\mathcal{V}$ both compute the following :
$$R = P \cdot Q^t \cdot A_1^{t^2} \cdot A_2^{t^3} \quad \text{and} \quad z = z_1 + t \cdot z_2 + t^2 \cdot z_3 + t^3 \cdot z_4$$
10. $\mathcal{V}$ outputs 1 if for $\mathbf{c^{n-1}} = (1, \ldots, c^{n-1}) \in \mathcal{PW}_n$ the following relation holds, and outputs 0 otherwise:
$$\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{w}) = R \quad \wedge \quad \langle \mathbf{w}, \mathbf{c^{n-1}} \rangle = z$$

Fig. 3: Protocol Π_2 for relation $\mathcal{R}_{\text{CLF-rev}}$

and we have commitment to each polynomial. The polynomials are then evaluated at the random challenge sent by the verifier $\mathcal{V}$, and the consistency of the evaluations with the equation satisfied by the polynomial is checked.

Theorem 3. *Π_2 is a protocol for relation $\mathcal{R}_{CLF\text{-}rev}$. It is perfectly complete and computationally special sound.*

The proof is given in Section B.2 of the Supplementary Material.

Now we note that the last message $\mathbf{w}$ sent by the prover to the verifier in Π_2 (Fig 3) is a witness for the relation $\mathcal{R}$, where $\mathcal{R} = \{(P \in \mathbb{G}, L \in \mathcal{L}(\mathbb{Z}_q^n), y \in \mathbb{Z}_q; \mathbf{x} \in \mathbb{Z}_q^n) : P = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}) \wedge L(\mathbf{x}) = y\}\}$, and the check computed by the verifier in step 10 of Π_2 corresponds to ensuring that $(R, \mathbf{c^{n-1}}, z; \mathbf{w}) \in \mathcal{R}$.

We state the following protocol for relation $\mathcal{R}$ which is the compressed proof of knowledge protocol stated in [AC20] with the following key differences: the linear form evaluation is checked in clear, commitment uses structured commitment key and commitment to the left and right half of the witness sent in the protocol being used to establish consistency with the commitment to the whole key which is

only possible due to the usage of structure in commitment key with keys being hidden in the exponent.

We note that even with the same protocol technique as [AC20] which inherently incurs linear computational complexity for verifier, we manage to retain a logarithmic computational complexity. This is due to the usage of structured commitment key, which does not require the verifier to compute a challenge dependent commitment key for the next iteration, and having a nicely-structured linear form which ensures that verifier can compute the challenge dependent linear form required for the next iteration efficiently. This suffices for our cause as we aim to use this protocol for providing proof of knowledge of the last message sent in step 8 of Π_2 such that it satisfies the verifier check in the step 10, which provides us a witness of the relation $\mathcal{R}$.

Parameters

- **Common parameters** : $(R \in \mathbb{G}, L_c \in \mathbb{Z}_q^n, z \in \mathbb{Z}_q, H^{\dot{\mathbf{a}}} \in \mathbb{G}^\ell)$
  - $R = \mathrm{COM}_{\overline{\mathbf{a}}}\ (\mathbf{w}), L_c = \mathbf{c^{n-1}} = (1, c, \ldots, c^{n-1}), z = L_c(\mathbf{w})$,
  - $n = 2^\ell, \dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_n), \overline{\mathbf{a}} = \left(\prod_{i=1}^{\ell} \dot{a}_i^{b_i}\right)_{b_i \in \{0,1\}}$
- $\mathcal{P}$**'s input** : $(g^{\overline{\mathbf{a}}} \in \mathbb{G}^n, \mathbf{w} \in \mathbb{Z}_q^n, L_c = \mathbf{c^{n-1}} \in \mathbb{Z}_q^n)$

Protocol

1. $\mathcal{P}$ computes and sends A_1, A_2, z' to $\mathcal{V}$
   (a) $A_1 = \mathrm{COM}_{\overline{\mathbf{a}}_L}\ (\mathbf{w}_L)$
   (b) $A_2 = \mathrm{COM}_{\overline{\mathbf{a}}_L}\ (\mathbf{w}_R)$
   (c) $z' = \langle \mathbf{w}_L, (L_c)_L \rangle = \langle \mathbf{w}_L, \mathbf{c}^{\frac{\mathbf{n-1}}{\mathbf{2}}} \rangle$
2. $\mathcal{V}$ checks if
$$e\left(\frac{R}{A_1}, g\right) = e\left(A_2, g^{\dot{a}_\ell}\right)$$
   If the check fails, $\mathcal{V}$ aborts, otherwise $\mathcal{V}$ continues.
3. $\mathcal{V}$ samples $s \leftarrow_R \mathbb{Z}_q$ and sends s to $\mathcal{P}$
4. $\mathcal{P}$ sets $\mathbf{w}' = \mathbf{w}_L + s \cdot \mathbf{w}_R, L_c' = s(L_c)_L + (L_c)_R = (s + c^{n/2})\mathbf{c^{n/2-1}}$ and implicitly sets $\dot{\mathbf{a}}' = (\dot{a}_1, \ldots, \dot{a}_{\ell-1})$ and $\overline{\mathbf{a}}' = \overline{\mathbf{a}}_L$
5. $\mathcal{P}$ and $\mathcal{V}$ both compute the following :
$$R' = A_1 A_2^s \ \text{ and } \ d = c^{n/2} \cdot z' + s \cdot z + s^2 \cdot c^{-n/2} \cdot (z - z')$$
6. If $\mathbf{w}' \notin \mathbb{Z}_q^2$: $\mathcal{P}$ runs PoK Π_2' to prove knowledge of $\mathbf{w}', L_c'$ such that $\mathrm{COM}_{\overline{\mathbf{a}}'}\ (\mathbf{w}') = P'$ and $\langle L_c', \mathbf{x}' \rangle = d$. Hence, $\mathcal{P}$ and $\mathcal{V}$ runs the protocol Π_2' with updated common parameters $(P', L_c', d, g^{\dot{\mathbf{a}}'})$ and prover's input $(g^{\overline{\mathbf{a}}'}, \mathbf{w}')$, for $(P', L_c', d; \mathbf{w}') \in \mathcal{R}$
7. If $\mathbf{w}' \in \mathbb{Z}_q^2$:
   (a) $\mathcal{P}$ sends $\mathbf{w}', L_c'$ to $\mathcal{V}$
   (b) $\mathcal{V}$ outputs 1 if the following holds, and outputs 0 otherwise:
$$\mathrm{COM}_{\overline{\mathbf{a}}'}\ (\mathbf{w}') = R' \ \wedge \ \langle L_c', \mathbf{w}' \rangle = d$$

Fig. 4: Protocol Π_2' for $(R, L_c, z; \mathbf{w}) \in \mathcal{R}$

We note that the last message vector (polynomial) sent by the prover to the verifier of Π_2 is aimed to convince the verifier that the vector is consistent with opening of a group element computed by the verifier and evaluation of the polynomial at a random field element is consistent with a public field element computed by the verifier. We provide protocol Π_2' for this.

We treat the evaluation of the polynomial $\mathbf{w}$ at a fixed point, denoted by $\mathbf{w}(c)$ as an inner-product relation with a univariate polynomial, denoted by $\langle \mathbf{w}, \mathbf{c^{n-1}} \rangle$, where $\mathbf{c^{n-1}} = (1, c, \ldots, c^{n-1})$. Now, provided that the evaluation point is fixed at $c \in \mathbb{Z}_q$, this inner product relation can be thought of as a linear form evaluation, where the public linear form $\mathbf{c^{n-1}}$ evaluation at a secret vector $\mathbf{w}$ is equal to the public value $z \in \mathbb{Z}_q$. Now, we note that, the claim in step 10 is equivalent to providing a Proof of Knowledge of witness $\mathbf{w}$ in the following relation :

$$\mathcal{R} = \{(P \in \mathbb{G}, L \in \mathcal{L}(\mathbb{Z}_q^n), y \in \mathbb{Z}_q; \mathbf{x} \in \mathbb{Z}_q^n) : P = \mathrm{COM}_{\overline{\mathbf{a}}}\ (\mathbf{x}) \ \wedge \ L(\mathbf{x}) = y\}\}$$

where we have that $(R, \mathbf{c^{n-1}}, z; \mathbf{w}) \in \mathcal{R}$.

Theorem 4. *Π_2' is a $(k_1, \ldots, k_\ell)$-move protocol for relation $(R, \mathbf{c^{n-1}}, z; \boldsymbol{w}) \in \mathcal{R}$, where $k_i = 3,\ \forall i \in [\ell]$. It is perfectly complete and computationally special sound.*

The proof is in Section B.3 of the Supplementary Material.

Theorem 5. *$(\Pi_2)_c = \Pi_2' \circ \Pi_2$ is a $(2n, 4, k_1, \ldots, k_\ell)$-move protocol for relation $\mathcal{R}_{CLF\text{-}rev}$, where $k_i = 3,\ \forall i \in [\ell]$. It is perfectly complete and computationally special sound. It incurs total communication of $(2 + 2\log n)$ group elements and $6 + 2\log n$ field elements.*

We note that $(\Pi_2)_c$ performs better than Π_1 for the relation $\mathcal{R}_{\text{CLF-rev}}$, however the pre-processing step needs a commitment to reverse of the linear forms. This is fine in our application to construct proofs for circuit satisfiability, since the commitments to the reverse of these linear forms is computed in the preprocessing phase. In case we only a commitment to the linear form, we can still use our protocol by having the prover send the commitment to the reversed linear form, together with a proof that it is indeed correct. This can be achieved by the observation that for $L \in \mathcal{L}(\mathbb{Z}_q^n)$ considered as a polynomial, being evaluated at c has equal value as that of its reverse being evaluated at c^{-1} and the result being multiplied with c^{n-1}.

$$L(c) = c^{n-1} \cdot (\mathsf{rev}(L))\left(c^{-1}\right) \quad \iff \quad \langle L, \mathbf{c^{n-1}} \rangle = c^{n-1} \cdot \langle \mathsf{rev}(L), (\mathbf{c^{-1}})^{\mathbf{n-1}} \rangle$$

Hence, if $P = \text{COM}_{\overline{\mathbf{a}}}(L)$ is computed in the pre-processing phase, then the prover can compute $Q = \text{COM}_{\overline{\mathbf{a}}}(\mathsf{rev}(L))$ and send Q along with the proof that opening of P evaluated at a random challenge c is c^{n-1} times Q evaluated at c^{-1}, at the onset of the protocol and proceed with $(\Pi_2)_c$. This gives us an overhead of 1 group element and 2 field elements. Finally, we define $\Pi_{2\text{-}\mathcal{R}} = (\Pi_2)_c \circ \Pi_0$ for relation $\mathcal{R}$, whose communication and computational complexity are dominated by that of $(\Pi_2)_c$.

4 Updatable SRS zkSNARK for Circuit Satisfiability

In this section we construct a zkSNARK with updatable SRS for circuit satisfiability by reducing a statement about a circuit with respect to a public input to opening a linear form.

We take the approach of Attema et al. [AC20] to handle multiplication gates by linearizing them, but we need to employ some new ideas to keep the verifier succinct. We recall the technique for handling multiplication gates in the Attema et al. [AC20], where we have the left input wire values $\mathbf{w}_a$, the right input wire values $\mathbf{w}_b$ and the output wire values $\mathbf{w}_o$, secret shared via packed secret sharing, where the randomness is embedded in the constant term. Let f, g and h be the polynomials with the packed secret sharing of $\mathbf{w}_a$, $\mathbf{w}_b$ and $\mathbf{w}_o$ respectively,such that $f(X) \cdot g(X) = h(X)$. Attema et al. [AC20] handles it by sending a commitment to the wire values in a long vector and opening them at a random point c, and then using Schwartz Zippel lemma to argue that the polynomials are identical if $f(c) \cdot g(c) = h(c)$ holds. However, the protocol to check $f(c) \cdot g(c) = h(c)$ renders the verifier linear, since the linear form for opening the polynomials at the random value is linear in the size of the witness. We circumvent this drawback of linear verification complexity from having to read the linear form, by obtaining *commitments* to the linear form. The goal is to commit to linear forms required for openings of f, g and h, and then invoke our succinct-verifier linear form protocol. We then proceed to prove that, given A, B and C as commitment to some secret vectors $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ respectively from $\mathbb{Z}_q^n$, the committed vectors satisfies the hadamard relation $\mathbf{a} \circ \mathbf{b} = \mathbf{c}$, i.e. $a_i b_i = c_i$ for all $i \in [n]$.

Following that, we show how to prove that given commitments A, B to two vectors $\mathbf{s}, \mathbf{r} \in \mathbb{Z}_q^n$, they are some committed permutation of each other. Concretely, $\mathbf{s}, \mathbf{r} \in \mathbb{Z}_q^n$ are such that $\mathbf{s} = \sigma(\mathbf{r})$ for some known permutation σ. Finally, we show how to put together these building blocks to construct a protocol for circuit satisfiability with logarithmic proof size and verification complexity.

4.1 Committing to a Linear Form for Multiplication Gates

Let $\rho_c = V^{-1}(1\ c\ c^2\ \cdots\ c^{n-1})^T$ for a random challenge c chosen by the verifier, where V is the Vandermonde matrix of the public evaluation points $\alpha_i, i \in [n]$. This enables us to compute $f(c) =$

Parameters

- Parameters from preprocessing:
  - $X := \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x})$ where $\mathbf{x} := (2, \ldots, 2^n) \in \mathbb{Z}_q^n$,
  - $\mathbf{1}_{\mathsf{com}} := \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{1})$ where $\mathbf{1} = (1, \ldots, 1) \in \mathbb{Z}_q^n$
- Common input:
  - V is the Vandermonde matrix defined in equation 1.

Protocol

1. $\mathcal{V}$ samples $c \leftarrow_R \mathbb{Z}_q$ and sends it to $\mathcal{P}$.
2. $\mathcal{P}$ sets ρ_c, where $\rho_c = (\rho_{c_1}, \cdots, \rho_{c_n}) = V^{-1}(1\ c\ c^2 \cdots c^{n-1})^T$ and $\rho_c{}' = (0, \rho_{c_1}, \cdots, \rho_{c_n})$, and computes $A = \mathrm{COM}_{\overline{\mathbf{a}}}(\rho_c), A' = \mathrm{COM}_{\overline{\mathbf{a}}}(\rho_c{}')$.
3. $\mathcal{P}$ sends A, A' to $\mathcal{V}$.
4. $\mathcal{V}$ samples $t \leftarrow_R \mathbb{Z}_q \setminus \{2^{-1}, \cdots, 2^{-i}, \cdots, 2^{-n}\}$ and sends t to $\mathcal{P}$.
5. $\mathcal{P}$ sets the j^{th} row of V as V_j, i.e. $V_j := (2^{j-1},\ 2^{2(j-1)}, \cdots, 2^{i(j-1)}, \cdots, 2^{n(j-1)})\ \forall j \in \{1, \cdots, n\}$, and computes $B := \mathrm{COM}_{\overline{\mathbf{a}}}(V(t))$, where $V(t) := (\mathbf{t^{n-1}})^T V = \sum_{j=0}^{n} t^{n-1} V_j$.
6. $\mathcal{P}$ sends B to $\mathcal{V}$.
7. The verifier samples $y \leftarrow_R \mathbb{Z}_q \setminus \{1, 2^{-1}\}, d \leftarrow_R \mathbb{Z}_q$ and sends y, e to $\mathcal{P}$.
8. $\mathcal{P}$ sets $\mathbf{z} = (2^i t - 1)_{i \in [n]}$, $\gamma = \langle \rho_c, \mathbf{d^{n-1}} \rangle$ and sends γ to $\mathcal{V}$.
9. $\mathcal{P}$ and $\mathcal{V}$ independently computes $Z = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{y^{n-1}} \circ \mathbf{z}) = X^t \cdot (\mathbf{1}_{\mathsf{com}})^{-1}$, $\alpha = 2^n t^n \frac{(2^n y)^n - 1}{2^n y - 1} - \frac{y^n - 1}{y - 1}$ and $\beta = \frac{(ct)^n - 1}{ct - 1}$.
10. $\mathcal{P}$ and $\mathcal{V}$ interact to prove the following relation:
    (a) runs $(\Pi_2)_c$ for $(B, A, \beta; (V(t))^T, \rho_c) \in \mathcal{R}_{\text{CLF-rev}}$
    (b) runs $\Pi_{2\text{-}\mathcal{R}}$ for $(A, \mathbf{d^{n-1}}, \gamma; \rho_c), (A', \mathbf{d^{n-1}}, d\gamma; \rho_c{}') \in \mathcal{R}$
    (c) runs $\Pi_{2\text{-}\mathcal{R}}$ for $(B, \mathbf{y^{n-1}} \circ \mathbf{z}, \alpha; V(t)) \in \mathcal{R}$

Fig. 5: Protocol $\Pi_{\mathsf{com\text{-}mult}}$ for obtaining commitment to linear form for multiplication gates.

$f \cdot V \cdot \rho_c = (f(\alpha_1) \ldots f(\alpha_n)) \cdot \rho_c$ for a polynomial $f \in \mathbb{Z}_{q_{\leq n}}[X]$. Now, instead of having the verifier compute a commitment to ρ_c (which would render it linear), we instead offload the computation of ρ_c to the prover and have the verifier *check* this computation in logarithmic time.

To check if a group element is indeed a commitment to ρ_c in logarithmic time, our key idea is to instantiate V as follows

$$V = \begin{pmatrix} 1 & \cdots & 1 & \cdots & 1 \\ 2 & \cdots & 2^i & \cdots & 2^n \\ 2^2 & \cdots & 2^{2i} & \cdots & 2^{2n} \\ \vdots & & \vdots & & \vdots \\ 2^{(n-1)} & \ldots & 2^{(n-1)i} & \ldots & 2^{(n-1)n} \end{pmatrix} \tag{1}$$

This enables us to reduce the verification of ρ_c to a series of n linear form checks, where the linear forms correspond to the rows of V. We then use the structure of V to express a random linear combination of the rows of V in a way that is easily checkable.

Let us define the relation $\mathcal{R}_{\mathsf{com\text{-}mult}}$ as follows:

$$\begin{aligned} \mathcal{R}_{\mathsf{com\text{-}mult}} = \{ \quad & (A_1 \in \mathbb{G}, A_2 \in \mathbb{G}, V \in \mathbb{Z}_q^{n \times n}, \mathbf{c^{n-1}} \in \mathbb{Z}_q^n; \rho_c, \rho_c{}') : \\ & \mathbf{c^{n-1}} = (1\ c\ \ldots\ c^{n-1}), \rho_c = V^{-1}\mathbf{c^{n-1}}, \rho_c{}' = 0 \| \rho_c, \\ & A_1 = \mathrm{COM}_{\overline{\mathbf{a}}}(\rho_c), A_2 = \mathrm{COM}_{\overline{\mathbf{a}}}(\rho_c{}')\} \end{aligned} \tag{2}$$

This relation captures obtaining commitment to a linear form consisting of public linear combination coefficients to obtain evaluation of an n-degree polynomial at a randomly chosen point c by the verifier. We note that $\rho_c{}'$ here denotes the vector (linear form) ρ_c shifted to the right by one, which is used in the protocols in subsequent sections to open polynomials defined by evaluations at the vector $(1, c_1, \ldots, c_n)$ as $(1, c_1, \ldots, c_{n-1})$ and $(c_1, \ldots, c_n)$ with the same vector description. That is, given a vector $(1, c_1, \ldots, c_n)$, we can use our relation to capture linear forms to evaluate polynomials defined by both $(1, c_1, \ldots, c_{n-1})$ and $(c_1, \ldots, c_n)$ simultaneously. Figure 5 presents the protocol $\Pi_{\mathsf{com\text{-}mult}}$ for the relation $\mathcal{R}_{\mathsf{com\text{-}mult}}$.

Note that it is easy to add zero checks to the protocol in Figure 5 to get a commitment to $\rho_{\mathsf{n}} \| \mathbf{0} \in \mathbb{Z}_q^{n'}$. Let $n' > n$ be the length of the commitment key. The verifier samples a challenge $t \leftarrow_R \mathbb{Z}_q$ and checks that the commitment P_n claimed to be to $\rho_{\mathsf{n}} \in \mathbb{Z}_q^n$ satisfies $(P_n, \mathbf{0}^n \| \mathbf{t}^{n'-n}, 0; \rho_{\mathsf{n}} \| \mathbf{0}) \in \mathcal{R}$. Moreover, it is also easy to get a commitment to the reverse of ρ_{n}. For this, the verifier samples a challenge u and asks the prover to make a claim of the form $\langle \rho_{\mathsf{n}}, \mathbf{u^n} \rangle = v$. It then checks if the commitments P_n, Q_n claimed to be to be to ρ_{n} and its reverse satisfy $(P_n, \mathbf{u^n}, v), (Q_n, \mathsf{rev}(\mathbf{u^n}), v) \in \mathcal{R}_{\text{CLF-rev}}$.

Theorem 6. *$\Pi_{\mathsf{com\text{-}mult}}$ is a $(7, 4, k_1, \ldots, k_\ell)$-move protocol for relation $\mathcal{R}_{\mathsf{com\text{-}mult}}$ (equation 2). It is perfectly complete and computationally special sound.*

Proof. **Completeness.** The prover $\mathcal{P}$ computes $V(t) = \mathbf{t^{n-1}} V = \sum_{j=1}^{n} t^{j-1} V_j$ and $B = \text{COM}_{\overline{\mathbf{a}}}(V(t))$, where V_j is the j^{th} row of V (equation 1) and $t \neq 1, \ldots, 2^{-i}, \ldots, 2^{-(n-1)}$. Then,

$$V(t) = \left(\tfrac{(2t)^n - 1}{2t-1}, \cdots, \tfrac{(2^i t)^n - 1}{2^i t - 1}, \cdots, \tfrac{(2^n t)^n - 1}{2^n t - 1} \right)$$

Let us define $\mathbf{z} = (2t - 1, \ldots, \ 2^i t - 1, \ldots, 2^n t - 1)$, then we have $V(t) \circ \mathbf{z} = ((2t)^n - 1, \ldots, (2^i t)^n - 1, \ldots, (2^n t)^n - 1)$, which ensures that, for any $y \in \mathbb{Z}_q$, we have $\langle \mathbf{y^{n-1}} \circ \mathbf{z}, V(t) \rangle = \langle \mathbf{y^{n-1}}, V(t) \circ \mathbf{z} \rangle = 2^n t^n \frac{(2^n y)^n - 1}{2^n y - 1} - \frac{y^n - 1}{y - 1} = \alpha$. This ensures that $(B, \mathbf{y^{n-1}} \circ \mathbf{z}, \alpha; V(t)) \in \mathcal{R}$. Also, $\langle (V(t))^T, \rho_c \rangle = \langle V^T \mathbf{t^{n-1}}, V^{-1} \mathbf{c^{n-1}} \rangle = \langle \mathbf{t^{n-1}}, \mathbf{c^{n-1}} \rangle = \frac{(ct)^n - 1}{ct - 1} = \beta$, ensures that $(B, A, \beta; (V(t))^T, \rho_c) \in \mathcal{R}_{\text{CLF-rev}}$.

Since $\langle \rho_c{}', \mathbf{d^{n-1}} \rangle = d \langle \rho_c, \mathbf{d^{n-1}} \rangle$, we have that $(A, \mathbf{d^{n-1}}, \gamma; \rho_c) \in \mathcal{R}$, and $(A', \mathbf{d^{n-1}}, d\gamma; \rho_c{}') \in \mathcal{R}$ for $\gamma = \langle \rho_c, \mathbf{d^{n-1}} \rangle \in \mathbb{Z}_q$.

Special Soundness. Our extractor uses the extractor for $(\Pi_2)_c$ and $\Pi_{2\text{-}\mathcal{R}}$, invoked in step 10 of $\Pi_{\mathsf{com\text{-}mult}}$, as a subroutine. Given $(2n, 4, 3, \ldots, 3)$ tree of accepting transcripts for $(\Pi_2)_c$ invoked for relation $(B, A, \beta; V(t), \rho_c) \in \mathcal{R}_{\text{CLF-rev}}$, we run the extractor for $(\Pi_2)_c$ to obtain openings of B, A and the binding of the commitments and soundness of the protocol ensures that the extracted openings are $V(t)$ and ρ_c such that $\langle \rho_c, (V(t))^T \rangle = \beta$. Similarly, given $(2, 2n, 4, 3, \ldots, 3)$ tree of accepting transcripts for $\Pi_{2\text{-}\mathcal{R}}$ invoked for relations $(A, \mathbf{d^{n-1}}, \gamma; \rho_c)$, $(A', \mathbf{d^{n-1}}, d\gamma; \rho_c{}')$, and $(B, \mathbf{y^{n-1}} \circ \mathbf{z}, \alpha; V(t)) \in \mathcal{R}$, we run the extractor for $(\Pi_2)_c$ to obtain openings of A, A', B and the binding of the commitments and soundness of the protocol ensures that the extracted openings are ρ_c, $\rho_c{}'$ and $V(t)$ such that $\langle \mathbf{d^{n-1}}, \rho_c \rangle = \gamma$, $\langle \mathbf{d^{n-1}}, \rho_c{}' \rangle = d\gamma$ and $\langle \mathbf{y^{n-1}} \circ z, V(t) \rangle = \alpha$. Hence, we get that the following relations holds:

1. $\langle \mathbf{y^{n-1}}, V(t) \circ \mathbf{z} \rangle = \langle \mathbf{y^{n-1}} \circ \mathbf{z}, V(t) \rangle = \alpha = 2^n t^n \frac{(2^n y)^n - 1}{2^n y - 1} - \frac{y^n - 1}{y - 1}$
   $\implies \langle \mathbf{y^{n-1}}, V(t) \circ \mathbf{z} \rangle = 2^n t^n \frac{(2^n y)^n - 1}{2^n y - 1} - \frac{y^n - 1}{y - 1}$.
2. $\langle V \rho_c, \mathbf{t^{n-1}} \rangle = (V \rho_c)^T \mathbf{t^{n-1}} = {\rho_c}^T (V^T \mathbf{t^{n-1}}) = {\rho_c}^T V(t)^T = \langle \rho_c, V(t)^T \rangle = \frac{(ct)^n - 1}{ct - 1} \implies V \rho_c = \mathbf{c^{n-1}}$
3. $\langle \rho_c{}', \mathbf{d^{n-1}} \rangle = d\gamma = d \langle \rho_c, \mathbf{d^{n-1}} \rangle$
   $\implies \langle \rho_c{}', \mathbf{d^{n-1}} \rangle = d \langle \rho_c, \mathbf{d^{n-1}} \rangle$

The last relation provides us, that given two polynomials ρ_c and $\rho_c{}'$ defined by their vector of coefficients, we have $\rho_c(d) = d \cdot \rho_c{}'(d)$ for some $d \in \mathbb{Z}_q$ sampled completely at random. Hence, given accepting transcripts with n-different challenges y, d and t each, following relations hold from Schwartz Zippel Lemma:

1. $V(t) \circ \mathbf{z} = ((2t)^n - 1, \ldots, (2^i t)^n - 1, \ldots, (2^n t)^n - 1)$, where $\mathbf{z} = (2^i t - 1)_{i \in [n]}$
   $\implies V(t) = \sum_{j=0}^{n} t^{n-1} V_j$, where $V_j = (2^{j-1}, \ 2^{2(j-1)}, \cdots, 2^{i(j-1)}, \cdots, 2^{n(j-1)})$
2. $V \rho_c = \mathbf{c^{n-1}}$
3. $\rho_c{}' = 0 \| \rho_c = (0, \rho_{c1}, \ldots, \rho_{cn})$ when $\rho_c = (\rho_{c1}, \ldots, \rho_{cn})$

which ensures that our extracted vectors are such that ρ_c is a linear form whose commitment is provided, and ρ_c contains the coefficient linear combinations for obtaining evaluation at c. Also, $\rho_c{}'$ is a linear form which is ρ_c shifted by one place to the right.

4.2 Hadamard Product Argument

Let $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^n$, recall that the hadamard product is defined as $\mathbf{a} \circ \mathbf{b} = (a_1 b_1, \ldots, a_n b_n) \in \mathbb{Z}_q^n$. Our goal is to prove knowledge of three vectors that satisfy the hadamard product relation, given succinct commitment to the vectors.

Concretely, given three vectors $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{Z}_q^n$ such that $\mathbf{a} \circ \mathbf{b} = \mathbf{c}$, we define $p_{\boldsymbol{a}}(X), p_{\boldsymbol{b}}(X), p_{\boldsymbol{c}}(X) \in \mathbb{Z}_q^n[X]$ such that $p_{\boldsymbol{a}}(2^i) = a_i, p_{\boldsymbol{b}}(2^i) = b_i$ for all $i \in [n]$ and $p_{\boldsymbol{c}}(X) := p_{\boldsymbol{a}}(X) \cdot p_{\boldsymbol{b}}(X)$. We define $h_{(\mathbf{c})} = (p_{\boldsymbol{c}}(2^{n+1}), \ldots, p_{\boldsymbol{c}}(2^{2n-1}))$. The protocol proceeds as follows. The prover computes commitments A, B, C to the vectors $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}' := \mathbf{c} \| h_{(\mathbf{c})}$ respectively. The verifier then samples a challenge z, and the prover responds with commitments P_n, P_{2n} to the reverse of ρ_{n} and $\rho_{2\mathsf{n}}$, where ρ_{n} and $\rho_{2\mathsf{n}}$ are defined as $\rho_{\mathsf{n}} = V^{-1}(1 \; z \; z^2 \; \cdots \; z^{n-1})^T$, $\rho_{2\mathsf{n}} = V^{-1}(1 \; z \; z^2 \; \cdots \; z^{2n-2})^T$. Then the prover opens the polynomial evaluations of $p_{\boldsymbol{a}}(X), p_{\boldsymbol{b}}(X), p_{\boldsymbol{c}}(X)$ at a random point chosen by the verifier, using the commitments to the vectors and the linear forms.

The hadamard relation $\mathcal{R}_{\mathsf{had}}$ with suitable modification to incorporate the commitments to the vectors is defined below, and the protocol Π_{had} presents the protocol for relation $\mathcal{R}_{\mathsf{had}}$. Note that to ensure zero-knowledge property of the protocol Π_{had}, to prove $\mathbf{a} \circ \mathbf{b} = \mathbf{c}$, we invoke the protocol for $(A, B, C; \mathbf{a} \| d, \mathbf{b} \| e, \mathbf{c} \| de) \in \mathcal{R}_{\mathsf{had}}$ where $d, e \leftarrow_R \mathbb{Z}_q$.

$$\mathcal{R}_{\mathsf{had}} = \{(A \in \mathbb{G}, B \in \mathbb{G}, C \in \mathbb{G}; \mathbf{a} \in \mathbb{Z}_q^n, \mathbf{b} \in \mathbb{Z}_q^n, \mathbf{c} \in \mathbb{Z}_q^n) : \\ A = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{a}), B = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{b}), \mathbf{c}' = \mathbf{c} \| h_{(\mathbf{c})}, C = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{c}')\}$$

Parameters

- Common input:
  - V is the Vandermonde matrix defined in equation 1.
  - $A = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{a}), B = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{b}), C = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{c}')$, such that $\mathbf{c}' = \mathbf{c} \| h_{(\mathbf{a} \circ \mathbf{b})}$
- $\mathcal{P}$'s input: $\mathbf{a} = \mathbf{a}^* \| d, \mathbf{b} = \mathbf{b}^* \| e, \mathbf{c} = \mathbf{c}^* \| de$ such that $\mathbf{a}^* \circ \mathbf{b}^* = \mathbf{c}^*$ and $d, e \leftarrow_R \mathbb{Z}_q$

Protocol

1. $\mathcal{P}$ computes the polynomials $p_{\boldsymbol{a}}, p_{\boldsymbol{b}} \in \mathbb{Z}_q^n[X]$ as $p_{\boldsymbol{a}}(2^i) := a_i, p_{\boldsymbol{b}}(2^i) := b_i \; \forall i \in [n]$. It defines $p_{\boldsymbol{c}}(X) := p_{\boldsymbol{a}}(X) \cdot p_{\boldsymbol{b}}(X)$.
2. $\mathcal{P}$ samples $\mathbf{u} \leftarrow_R \mathbb{Z}_q^n, \mathbf{u}' \leftarrow_R \mathbb{Z}_q^{2n-1}$ and defines $p_{\boldsymbol{u}} \in \mathbb{Z}_q^n[X], p_{\boldsymbol{u}'} \mathbb{Z}_q^{2n}[X]$ as $p_{\boldsymbol{u}}(2^i) := u_i \; \forall i \in [n]$, $p_{\boldsymbol{u}'}(2^i) := u'_i \; \forall i \in [2n]$. $\mathcal{P}$ computes $U = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{u}), U' = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{u}')$ and sends U, U' to $\mathcal{V}$.
3. $\mathcal{V}$ samples $z \leftarrow_R \mathbb{Z}_q$ and sends z to $\mathcal{P}$.
4. Define $\rho_{\mathsf{n}} = V^{-1}(1 \; z \; z^2 \; \cdots \; z^n)^T$ and $\rho_{2\mathsf{n}} = V^{-1}(1 \; z \; z^2 \; \cdots \; z^{2n-2})^T$. $\mathcal{P}$ and $\mathcal{V}$ run $\Pi_{\mathsf{com\text{-}mult}}$ to obtain commitments P_n, P_{2n} to the reverse of $\rho_{\mathsf{n}}, \rho_{2\mathsf{n}}$.
5. $\mathcal{P}$ sets w_1, w_2 as $w_1 = p_{\boldsymbol{a}}(z)$, and $w_2 = p_{\boldsymbol{b}}(z)$. $\mathcal{P}$ also sets v_1, v_2 as $v_1 = p_{\boldsymbol{u}}(z), v_2 = p_{\boldsymbol{u}'}(z)$.
6. $\mathcal{P}$ sends w_1, w_2, v_1, v_2 to $\mathcal{V}$.
7. $\mathcal{V}$ samples $r \leftarrow_R \mathbb{Z}_q$ and sends r to $\mathcal{P}$.
8. $\mathcal{P}$ and $\mathcal{V}$ independently computes $Y = UA^r B^{r^2}$, $Y' = U'C^r$, $\mathbf{y} = \mathbf{u} + r\mathbf{a} + r^2\mathbf{b}$, $\mathbf{y}' = \mathbf{u}' + r\mathbf{c}$, $q = v_1 + rw_1 + r^2 w_2$ and $q' = v_2 + rw_1 w_2$.
9. $\mathcal{P}$ and $\mathcal{V}$ runs $(\Pi_2)_c$ for
   (a) $(Y, P_n, q; \; \mathbf{y}, \rho_{\mathsf{n}}) \in \mathcal{R}_{\text{CLF-rev}}$.
   (b) $(Y', P_{2n}, q'; \; \mathbf{y}', \rho_{2\mathsf{n}}) \in \mathcal{R}_{\text{CLF-rev}}$.

Fig. 6: Protocol Π_{had} for $\mathcal{R}_{\mathsf{had}}$

Theorem 7. *Π_{had} is a protocol for $\mathcal{R}_{\mathsf{had}}$. It is perfectly complete, special honest-verifier zero-knowledge and computationally special sound.*

The proof is in Section C.1 of the Appendix.

4.3 Permutation Argument

Our starting point is the Bayer-Groth protocol [BG12] for the permutation argument. Let $\mathsf{PERM}_n = \{f : f : [n] \to [n]$ such that f is a permutation$\}$ and $\sigma \in \mathsf{PERM}_n$. For two vectors $\mathbf{r} = (r_1, \ldots, r_n) \in \mathbb{Z}_q^n$

and $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$, we aim to prove that $\sigma(\mathbf{r}) = \mathbf{s}$ for some publicly known σ. To prove the same, we leverage the technique introduced by Bayer and Groth of proving $\prod_{i=1}^{n}(r_i + i\beta + \gamma) = \prod_{i=1}^{n}(s_i + \sigma(i)\beta + \gamma)$ for verifier's choice of $\beta, \gamma \in \mathbb{Z}_q$ sampled uniformly at random.

The proof is instantiated by having the verifier choose two challenges $\beta, \gamma \in \mathbb{Z}_q$ and the prover constructing two vectors $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$ defined as $a_i = r_i + i\beta + \gamma$ and $b_i = s_i + \sigma(i)\beta + \gamma$ for all $i = 1, \ldots, n$, and providing a proof that $\prod_{i=1}^{n} a_i = \prod_{i=1}^{n} b_i$ holds. The proof for $\prod_{i=1}^{n} a_i = \prod_{i=1}^{n} b_i$ proceeds by constructing two vectors $\mathbf{c}', \mathbf{d}' \in \mathbb{Z}_q^{n+1}$ such that $c'_0 = 1, d'_0 = 1$ and $c'_j := \prod_{i=1}^{j}(r_i + i\beta + \gamma), d'_j := \prod_{i=1}^{j}(s_i + \sigma(i)\beta + \gamma)$, for all $j \in [n]$. Now we consider two circuits consisting of n multiplication gates, first circuit with vector of left inputs $\mathbf{a} = (a_1, \ldots, a_n)$, vector of right inputs $\mathbf{e} = (e_1, \ldots, e_n) = (1, c_1, \ldots, c_{n-1})$ and vector of outputs $\mathbf{c} = (c_1, \ldots, c_n)$, and second circuit with left input $\mathbf{b} = (b_1, \ldots, b_n)$, vector of right inputs $\mathbf{f} = (f_1, \ldots, f_n) = (1, d_1, \ldots, d_{n-1})$ and vector of outputs $\mathbf{d} = (d_1, \ldots, d_n)$. Our idea now is to check the hadamard product relations $\mathbf{a} \circ \mathbf{e} = \mathbf{c}$ and $\mathbf{b} \circ \mathbf{f} = \mathbf{d}$ by leveraging the shifted structure of the vectors in the hadamard products; and using protocol $\Pi_{\mathsf{com\text{-}mult}}$ yielding a succinct verifier permutation argument.

We consider the following relation $\mathcal{R}_{\mathsf{perm}}$ for the permutation argument.

$$\mathcal{R}_{\mathsf{perm}} = \{ (R, S, P; \mathbf{r}, \mathbf{s}, \sigma) : R = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{r}), S = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{s}), P = \mathrm{COM}_{\overline{\mathbf{a}}}(\sigma(I)), \\ I = (1, \ldots, n), \mathbf{s} = \sigma(\mathbf{r}) \}$$

We present the protocol Π_{perm} for the same in Fig 7. We define ρ_{n}, ρ_{2n}, δ_{n} and δ_{2n} as $\rho_{\mathsf{n}} = V^{-1}(1\ z\ z^2 \ldots z^{n-1})$, $\rho_{\mathsf{2n}} = V^{-1}(1\ z\ z^2 \ldots z^{2n-1})$, $\delta_{\mathsf{n}} = V^{-1}(1\ w\ w^2 \ldots w^{n-1})$ and $\delta_{\mathsf{2n}} = V^{-1}(1\ w\ w^2 \ldots w^{2n-1})$ where V is a Vandermonde matrix defined by the public evaluation points. We recall that the linear forms $\rho_{\mathsf{n}}, \rho_{\mathsf{2n}}$ are for computing evaluation at a random point z, and the linear forms $\delta_{\mathsf{n}}, \delta_{\mathsf{2n}}$ are for computing evaluation at a random point w. We note that we can batch the invocations of $(\Pi_2)_c$ for $\mathcal{R}_{\text{CLF-rev}}$ in each of the steps (a),(b) and (c) in Step 11 of Π_{perm} using the techniques of Attema et al. [AC20].

Theorem 8. *Π_{perm} is a protocol for $\mathcal{R}_{\mathsf{perm}}$. It is perfectly complete, special honest-verifier zero-knowledge and computationally special sound.*

We defer the proof to Section C.2 of the Supplementary Material.

4.4 Putting Things Together – zkSNARK for Circuit SAT

Given an upper bound on the circuit size n, the universal updatable SRS is generated by running COM .*Setup* to commit to $2n+2$-length vectors to obtain the commitment key $(g^{\overline{\mathbf{a}}}, H^{\hat{\mathbf{a}}})$. Here $g^{\overline{\mathbf{a}}}$ is the proving key and $H^{\hat{\mathbf{a}}}$ is the verification key. Since the SRS is universal, we need a circuit dependent setup phase so the verifier will read the circuit only once. We omit the description of algorithms for updating and verifying the SRS since this corresponds to updating and verifying the commitment key, and are the same as in Daza et al [DRZ20]. We note that the circuit-specific preprocessing material can be deterministically computed from the universal SRS and the circuit description, without any secrets.

We describe the protocol as an interactive public coin argument. The final zkSNARK construction is in the Random Oracle model using the Fiat-Shamir heuristic.

Preprocessing. We use the preprocessing phase used by Daza et al. [DRZ20] to obtain a commitment to the linear gates. They establish the existence of a circuit preprocessing methodology that effectively imposes constraints on the fan-in and fan-out of each gate in the circuit to a maximum value of M, which only incurs a linear expansion in the size of the circuit.

Let $\chi_1, \ldots, \chi_\nu$ be the public inputs of the circuit. Let m be the number of multiplication gates in the circuit. We then require a commitment key of size $n = 2m+2$. Let x_i^L, x_i^R, x_i^O denote the left input, right input and output of the i^{th} multiplication gate. Let $\mathbf{x}^L = (x_i^L)_{i\in[m]}, \mathbf{x}^R = (x_i^R)_{i\in[m]}, \mathbf{x}^O = (x_i^O)_{i\in[m]}$. Then $\mathbf{x}^L \circ \mathbf{x}^R = \mathbf{x}^O$. Additionally, there exist vectors $\mathbf{w}_i^L, \mathbf{w}_i^R \in \mathbb{Z}_q^m$ with at most M non-zero entries such that $\langle \mathbf{w}_i^L, \mathbf{x}^O \rangle + \mathbf{x}_i^L = \chi_i, \forall i \in [\nu]$, $\langle \mathbf{w}_i^L, \mathbf{x}^O \rangle = x_i^L, \forall i \in \{\nu+1, \ldots, m\}$ and $\langle \mathbf{w}_i^R, \mathbf{x}^O \rangle = x_i^R\ \forall i \in [m]$. Let $W^L, W^R \in \mathbb{Z}_q^{m\times m}$ be matrices with their i^{th} rows equal to $\mathbf{w}_i^L$ and $\mathbf{w}_i^R$ respectively. Then W^L and W^R have $\leq M$ entries in each row and each column. The following applies to W^k for $k \in \{L, R\}$.

Parameters

- Parameters from preprocessing:
  - $P = \mathrm{COM}_{\overline{\mathbf{a}}}(\sigma(I))$, $P' = \mathrm{COM}_{\overline{\mathbf{a}}}(I)$ for $I = (1, \ldots, n)$
  - $\mathsf{left} = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathsf{rev}(1\|\mathbf{0}\|\mathbf{0}))$ and $\mathsf{right} = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathsf{rev}(\mathbf{0}\|1\|\mathbf{0}))$ for linear forms $(1\|\mathbf{0}\|\mathbf{0})$ and $(\mathbf{0}\|1\|\mathbf{0})$, where $\mathbf{0} = (0, \ldots, 0) \in \mathbb{Z}_q^n$
  - $T = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{1})$, $\mathbf{1} = (1, \ldots, 1) \in \mathbb{Z}_q^n$
- Common Input:
  - $R = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{r}), S = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{s})$
- $\mathcal{P}$'s input : $(\mathbf{r}, \mathbf{s}, \sigma, g^{\overline{\mathbf{a}}})$

Protocol

1. $\mathcal{V}$ samples $\beta, \gamma \leftarrow_R \mathbb{Z}_q$ and sends β, γ to $\mathcal{P}$.
2. $\mathcal{P}$ computes $x := \prod_{i=1}^{n}(r_i + i\beta + \gamma)$ and sends x to $\mathcal{V}$.
3. $\mathcal{P}$ computes the vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^n$ such that $a_i = r_i + i\beta + \gamma$ and $b_i = s_i + \sigma(i)\beta + \gamma$ for all $i \in [n]$. $\mathcal{P}$ additionally computes $\mathbf{c}', \mathbf{d}' \in \mathbb{Z}_q^{n+1}$ such that $c'_1 = 1, d'_1 = 1$ and $c'_j := \prod_{i=1}^{j-1} a_i, d'_j := \prod_{i=1}^{j-1} b_i$, for all $j \in [n+1] \setminus \{1\}$, and defines $\mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f} \in \mathbb{Z}_q^n$ such that $c_i = c'_{i+1}, d_i = d'_{i+1}, e_i = c'_i, f_i = d'_i$ for all $i \in [n]$, i.e. $c_j := \prod_{i=1}^{j} a_i, d_j := \prod_{i=1}^{j} b_i$, for all $j \in [n]$, and $e_1 = 1, f_1 = 1$ and $e_j := \prod_{i=1}^{j-1} a_i, d_j := \prod_{i=1}^{j-1} b_i$, for all $j \in [n]$.
4. $\mathcal{P}$ computes the polynomials $p_{\boldsymbol{a}}, p_{\boldsymbol{e}}$ and $p_{\boldsymbol{c}}$ as $p_{\boldsymbol{a}}(2^i) := a_i, p_{\boldsymbol{e}}(2^i) := e_i$ and $p_{\boldsymbol{c}} := p_{\boldsymbol{a}} \cdot p_{\boldsymbol{e}}$, and similarly computes $p_{\boldsymbol{b}}, p_{\boldsymbol{f}}$ and $p_{\boldsymbol{d}}$ as $p_{\boldsymbol{b}}(2^i) := b_i, p_{\boldsymbol{f}}(2^i) := f_i$ and $p_{\boldsymbol{d}} := p_{\boldsymbol{b}} \cdot p_{\boldsymbol{f}}$.
5. $\mathcal{P}$ and $\mathcal{V}$ independently computes $A = R(P')^{\beta}T^{\gamma}$ and $B = SP^{\beta}T^{\gamma}$.
6. $\mathcal{P}$ computes $\mathbf{c}'' = \mathbf{c}'\|(p_{\boldsymbol{c}}(2^{n+1}), \ldots, p_{\boldsymbol{c}}(2^{2n}))$ and $\mathbf{d}'' = \mathbf{d}'\|(p_{\boldsymbol{d}}(2^{n+1}), \ldots, p_{\boldsymbol{d}}(2^{2n}))$, $C = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{c}'')$ and $D = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{d}'')$ and sends C, D to $\mathcal{V}$.
7. $\mathcal{V}$ samples $z, w \leftarrow_R \mathbb{Z}_q$ and sends z, w to $\mathcal{P}$.
8. $\mathcal{P}$ computes $\rho_{\mathsf{n}} = V^{-1}(1\ z\ z^2 \ldots z^{n-1})$, $\rho_{2\mathsf{n}} = V^{-1}(1\ z\ z^2 \ldots z^{2n-2})$, $\delta_{\mathsf{n}} = V^{-1}(1\ w\ w^2 \ldots w^{n-1})$ and $\delta_{2\mathsf{n}} = V^{-1}(1\ w\ w^2 \ldots w^{2n-2})$.
9. $\mathcal{P}$ and $\mathcal{V}$ runs $\Pi_{\mathsf{com\text{-}mult}}$ to obtain commitments P_n, P_{2n}, Q_n and Q_{2n} to the reverse of ρ_{n},$\rho'_{2\mathsf{n}}$,δ_{n} and $\delta'_{2\mathsf{n}}$ where $\rho'_{2\mathsf{n}} = 0\|\rho_{2\mathsf{n}}$ and $\delta'_{2\mathsf{n}} = 0\|\delta_{2\mathsf{n}}$.
10. $\mathcal{P}$ sets z_1, z_2, w_1, and w_2 as $z_1 = p_{\boldsymbol{a}}(z)$, $z_2 = p_{\boldsymbol{e}}(z)$, $w_1 = p_{\boldsymbol{b}}(w)$, and $w_2 = p_{\boldsymbol{f}}(w)$.
11. $\mathcal{P}$ and $\mathcal{V}$ run $(\Pi_2)_c$ to prove the following:
    (a) $(C, \mathsf{left}, 1; \mathbf{c}'', (1\|\mathbf{0}\|\mathbf{0}))$, $(C, \mathsf{right}, x; \mathbf{c}'', (\mathbf{0}\|1\|\mathbf{0}))$, $(D, \mathsf{left}, 1; \mathbf{d}'', (1\|\mathbf{0}\|\mathbf{0}))$, $(D, \mathsf{right}, x; \mathbf{d}'', (\mathbf{0}\|1\|\mathbf{0})) \in \mathcal{R}_{\text{CLF-rev}}$
    (b) $(A, P_n, z_1; \mathbf{a}, \rho_{\mathsf{n}})$,$(C, P_n, z_2; \mathbf{c}'', \rho_{\mathsf{n}})$, $(B, Q_n, w_1; \mathbf{b}, \delta_{\mathsf{n}})$, $(D, Q_n, w_2; \mathbf{d}'', \delta_{\mathsf{n}}) \in \mathcal{R}_{\text{CLF-rev}}$
    (c) $(C, P_{2n}, z_1 z_2; \mathbf{c}'', \rho'_{2\mathsf{n}})$, $(D, Q_{2n}, w_1 w_2; \mathbf{d}'', \delta'_{2\mathsf{n}}) \in \mathcal{R}_{\text{CLF-rev}}$.

Fig. 7: Protocol Π_{perm} for Permutation Argument

W^k can be written as the sum of M permutation matrices, i.e. $W^k = \sum_{i=1}^{M} W_i^k$, where each W_i^k is a permutation matrix.

In addition to the preprocessing of [DRZ20], additional preprocessing material is generated that is required by our sub-protocols, $\Pi_{\mathsf{com\text{-}mult}}$ Π_{had}, and Π_{perm}. The verifier obtains commitments to W_i^k, I and $\sigma(I)$, where $I = (1, \ldots, n)$, W_i^k and $\sigma_i^k : [n] \to [n]$ are as defined above. The verifier also obtains commitments to $\mathbf{1}$, $\mathbf{2}_{[n]}$, $(\mathbf{0}\|1\|\mathbf{0})$, $(\mathbf{0}\|\mathbf{0}\|1)$ where $\mathbf{0} = (0, \ldots, 0) \in \mathbb{Z}_q^n$, $\mathbf{1} = (1, \ldots, 1) \in \mathbb{Z}_q^n$, $\mathbf{2}_{[n]} = (2, \ldots, 2^n) \in \mathbb{Z}_q^n$.

Protocol Overview. Post circuit preprocessing, our circuit is now fully defined by $\tilde{\mathbf{w}}_i^k, \sigma_i^k$, where $\tilde{\mathbf{w}}_i^k$ is the vector containing the non-zero entry (if there is one) in each column of W_i^k and $\sigma_i^k : [n] \to [n]$ is the permutation that takes as input a column number j and outputs the row to which the j^{th} entry of $\mathbf{w}_i$ belongs. Our goal is to get a commitment to a random linear combination of the rows of W^k, i.e. a commitment to $W^k(c) = \sum_{i=1}^{M} \tilde{\mathbf{w}}_i^k \circ \sigma_i^k(\mathbf{c}^{\mathbf{m}})$. To do this, we first demand commitments to $\sigma_i^k(\mathbf{c}^{\mathbf{m}})$ from the prover, for a random challenge c chosen by the verifier. We can check that these commitments are honestly generated using Π_{perm}. We additionally ask the prover to provide us with commitments to $\tilde{\mathbf{w}}_i^k \circ \sigma_i^k(\mathbf{c}^{\mathbf{m}})$ and a proof $h_{(\tilde{\mathbf{w}}_i^k \circ \sigma_i(\mathbf{c}^{\mathbf{m}}))}$ that attests to the correct computation of a Hadamard product. To check this Hadamard product, we deploy our Π_{had} protocol. Since $\tilde{\mathbf{w}}_i^k, \sigma_i^k$ are public, Π_{had} can be invoked without requiring zero knowledge.

The above protocol allows us to get commitments to $W^L(c)$ and $W^R(c)$, but to show that the constraints of the circuit are satisfied, we need to prove that $\langle W^L(c) + uW^R(c), \mathbf{x}^O\rangle = \langle \mathbf{c^m}, \mathbf{x}^L\rangle - \sum_{i=1}^{\nu} c^{i-1}\chi_i + u\langle \mathbf{c^m}, \mathbf{x}^R\rangle = \langle \mathbf{c^m}, \mathbf{x}^L + u\mathbf{x}^R\rangle - \sum_{i=1}^{\nu} c^{i-1}\chi_i$ for $u \leftarrow_R \mathbb{Z}_q$. We cannot test for equality directly since that would require the prover to send out linear combinations of $\mathbf{x}^L, \mathbf{x}^R$ and $\mathbf{x}^O$, violating zero knowledge. Set $K = \sum_{i=1}^{\nu} c^{i-1}\chi_i$, $L_1 = W^L(c) + uW^R(c)$, $L_2 = \mathbf{c^m}, \mathbf{y}_1 = \mathbf{x}^O$ and $\mathbf{y}_2 = \mathbf{x}^L + u\mathbf{x}^R$. Let $\tilde{L}_2$ be the $m-1$ vector comprising of the first $m-1$ elements of L_2. Let $(L_2)_m$ be the last element of L_2. The above constraint can then be written as $\langle L_1, \mathbf{y}_1\rangle = \langle L_2, \mathbf{y}_2\rangle - K$. To prove this in zero knowledge, we have the prover sample $\mathbf{r}_1 \leftarrow \mathbb{Z}_q^m, \tilde{\mathbf{r}}_2 \leftarrow \mathbb{Z}_q^{m-1}$ and set $\mathbf{r}_2 = \tilde{\mathbf{r}}_2 || (\langle L_1, \mathbf{r}_1\rangle - \langle \tilde{L}_2, \tilde{\mathbf{r}}_2\rangle)(L_2)_m^{-1}$. This ensures that $\langle L_1, \mathbf{r}_1\rangle = \langle L_2, \mathbf{r}_2\rangle$. The protocol now proceeds as follows: the verifier samples a challenge z and the prover proves that $\langle L_1, z\mathbf{y}_1 + \mathbf{r}_1\rangle = \langle L_2, z\mathbf{y}_2 + \mathbf{r}_2\rangle - zK$. We can directly test for equality here since the prover now needs to reveal $\langle L_1, z\mathbf{y}_1 + \mathbf{r}_1\rangle$, which is a random value that reveals nothing about the input.

This allows us to conclude that the commitments to $\mathbf{x}^L, \mathbf{x}^R$ and $\mathbf{x}^O$ satisfy the linear combination constraints imposed by the circuit. Testing for multiplication, i.e. checking if $\mathbf{x}^L \circ \mathbf{x}^R = \mathbf{x}^O$ can be done by invoking our protocol Π_{had} by adding randomness to the input vectors in order to preserve zero-knowledge.

Since we reduce circuit satisfiability to opening a series of committed linear forms on committed vectors, we can optimize by batching the opening of several linear forms together. Consider two instances (P_1, Q_1, y_1) and (P_2, Q_2, y_2) claimed by the prover to belong to $\mathcal{R}_{\text{CLF-rev}}$. To prove this, we modify the protocol in Figure 3 as follows: let $\mathbf{x}_1, \mathbf{x}_2$ be the vectors to which P_1 and P_2 are commitments. Let B_1, B_2 be the linear forms to which Q_1 and Q_2 are commitments. We first demand that the prover send us commitments to $p_{L,1}, p_{R,1}, p_{L,2}$ and $p_{R,2}$ as it would in the original protocol. We then ask the prover to make claims about $\mathbf{x}_1(c), \mathbf{B}_1(c), \mathbf{p}_{L,1}(c), \mathbf{p}_{R,1}(c)$ and $\mathbf{x}_2(c), \mathbf{B}_2(c), \mathbf{p}_{L,2}(c), \mathbf{p}_{R,2}(c)$ with respect to the *same* challenge c. This allows us to combine the prover's claims to open $\mathbf{c^{n-1}}$ on a *single* vector given by $\mathbf{x}_1 + t\mathbf{B}_1 + t^2\mathbf{p}_{L,1} + t^3\mathbf{p}_{R,1} + t^4\mathbf{x}_2 + t^5\mathbf{B}_2 + t^6\mathbf{p}_{L,2} + t^7\mathbf{p}_{R,2}$ for a random challenge t. Thus, we can open $\mathcal{O}(M)$ linear forms while incurring the communication overhead of opening a single linear form. We present the complete protocol Π_{csat} in Fig. 8, and prove security in Appendix C.3.

Theorem 9. *Π_{csat} is a Public Coin, Honest Verifier Zero Knowledge Argument of Knowledge for CSAT with $\mathcal{O}(\log m)$ round complexity, $\mathcal{O}_\lambda(m)$ prover complexity, and $\mathcal{O}_\lambda(\log m)$ communication and verification complexity, where m is the number of multiplication gates in the preprocessed circuit.*

5 CSP for Committed Homomorphism

A bilinear group arithmetic circuit is a circuit in which the wire values are from $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ or $\mathbb{Z}_q$, and the gates are group operations, $\mathbb{Z}_q$-scalar multiplication, or bilinear pairings. Bilinear circuits are of interest since they directly capture relations arising in identity-based and attribute-based encryption [SW05, GPSW06], structure preserving signatures [AFG+10] etc. Handling bilinear circuits directly in a ZK system avoids expensive NP reductions or arithmetizations to represent group operations as an arithmetic circuit. The work of Attema et al. [ACR21], building on the work of Lai et al. [LMR19], gives a succinct argument system for bilinear group arithmetic circuits, by generalizing the compressed sigma protocol framework. A key building block is a protocol for opening a homomorphism on a committed vector. However, as in the case of arithmetic circuits, the verifier remains linear.

We construct a *designated-verifier* succinct argument for opening a committed homomorphism on a committed vector, where the verifier is *logarithmic*.

5.1 Commitment Scheme

In this section, we use additive notation for groups in line with prior works for bilinear circuits. We begin by generalizing the homomorphic commitment scheme of [LMR19], to work with logarithmic amount of randomness. We note that $\dot{\mathbf{a}}$ denotes $\dot{\mathbf{a}} = (a_1, \ldots, a_\ell)$ and for $g \in \mathbb{G}$ and $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, $g\mathbf{x}$ denotes $g\mathbf{x} = (gx_1, \ldots, gx_n)$ for $\mathbf{g} = (g_1, \ldots, g_n) \in \mathbb{G}^n$ and $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, inner product with scalar $\langle \mathbf{g}, \mathbf{x}\rangle$ denotes $\langle \mathbf{g}, \mathbf{x}\rangle = g_1x_1 + \ldots g_nx_n$; for $\mathbf{g} = (g_1, \ldots, g_n) \in \mathbb{G}_1^n$ and $\mathbf{h} = (h_1, \ldots, h_n) \in \mathbb{G}_2^n$, inner product $e(\mathbf{g}, \mathbf{h})$ denotes $e(\mathbf{g}, \mathbf{h}) = e(g_1, h_1) + e(g_2, h_2) + \ldots + e(g_n, h_n)$. Recall the key distribution

Universal updatable SRS: $(g^{\overline{\mathbf{a}}}, H^{\dot{\mathbf{a}}})$
Preprocessing Compute commitments to the following circuit-dependent vectors:

- $S_i^k = \text{COM}_{\overline{\mathbf{a}}}\ (\tilde{\mathbf{w}}_i^k \| \mathbf{0})\ \forall i \in [M]\ k \in \{L, R\}$
- $P_i^k = \text{COM}_{\overline{\mathbf{a}}}\ (\sigma_i^k \| \mathbf{0})\ \forall i \in [M]\ k \in \{L, R\}$
- $\texttt{ones} = \text{COM}_{\overline{\mathbf{a}}}\ (\mathbf{1^m} \| \mathbf{0})$, $P_0 = \text{COM}_{\overline{\mathbf{a}}}\ (\mathbf{m} \| \mathbf{0})$, where $\mathbf{m} = (1, 2, \cdots, m)$.

Input

- Public input $\chi_1, \ldots, \chi_n$
- $\mathcal{P}$'s input is the satisfying assignment $\mathbf{x}^L, \mathbf{x}^R, \mathbf{x}^O \in \mathbb{Z}_q^m$. $\mathcal{P}$ samples $d, e \leftarrow_R \mathbb{Z}_q$ and defines $\tilde{\mathbf{x}}^L = \mathbf{x}^L \| d$, $\tilde{\mathbf{x}}^R = \mathbf{x}^R \| e$, $\tilde{\mathbf{x}}^O = \mathbf{x}^O \| de \| h_{(\tilde{\mathbf{x}}^L \circ \tilde{\mathbf{x}}^R)} \in \mathbb{Z}_q^{2m+1}$
- V's inputs are the commitments $X^k = \text{COM}_{\overline{\mathbf{a}}}\ (\tilde{\mathbf{x}}^k; r^k)$ for $k \in \{L, R\}, X^O = \text{COM}_{\overline{\mathbf{a}}}\ (\tilde{\mathbf{x}}^O; r^O)$ with $r^L, r^R, r^O \leftarrow_R \mathbb{Z}_q$

Protocol

1. $\mathcal{V}$ sends $c \leftarrow_R \mathbb{Z}_q$ to $\mathcal{P}$.
2. $\mathcal{P}$ computes for $k \in \{L, R\}$:
   (a) $\Sigma_0 = \text{COM}_{\overline{\mathbf{a}}}\ (\mathbf{c^m} \| \mathbf{0})$
   (b) $\Sigma_i^k = \text{COM}_{\overline{\mathbf{a}}}\ (\sigma_i^k(\mathbf{c^m}) \| \mathbf{0})\ \forall i \in [M]$
   (c) $W_i^k = \text{COM}_{\overline{\mathbf{a}}}\ (\mathbf{w}_i^k \circ \sigma_i^k(\mathbf{c^m}) \| h_{(\mathbf{w}_i^k \circ \sigma_i^k(\mathbf{c^m}))})\ \forall i \in [M]$
   $\mathcal{P}$ sends all the computed commitments to $\mathcal{V}$.
3. $\mathcal{V}$ sends $u \leftarrow_R \mathbb{Z}_q$ to $\mathcal{P}$.
4. $\mathcal{P}$ samples $\mathbf{r}, \tilde{\mathbf{r}} \leftarrow_R \mathbb{Z}_q^m, s, \tilde{s} \leftarrow_R \mathbb{Z}_q$ such that $\langle W^L(c) + uW^R(c), \mathbf{r} \rangle = \langle \mathbf{c^m}, \tilde{\mathbf{r}} \rangle$ and sends $R = \text{COM}_{\overline{\mathbf{a}}}\ (\mathbf{r}; s)$ and $\tilde{R} = \text{COM}_{\overline{\mathbf{a}}}\ (\tilde{\mathbf{r}}; \tilde{s})$ to $\mathcal{V}$.
5. $\mathcal{V}$ sends $z \leftarrow_R \mathbb{Z}_q$ to $\mathcal{P}$.
6. $\mathcal{P}$ sets $L_1 = W^L(c) + uW^R(c), L_2 = \mathbf{c^m}, K = \sum_{i=1}^{\nu} c^{i-1}\chi_i$ and sends $v_1 = \langle L_1, z\mathbf{x}^O + \mathbf{r} \rangle = \langle L_2, z\mathbf{x}^L + zu\mathbf{x}^R + \tilde{\mathbf{r}} \rangle - zK$ and $\mathcal{L}_1^{rev} = \text{COM}_{\overline{\mathbf{a}}}\ (\mathsf{rev}(L_1))$ to $\mathcal{V}$.
7. $\mathcal{V}$ sends $t \neq c^{-1}, t', t_h, t_{perm} \leftarrow_R \mathbb{Z}_q$ to $\mathcal{P}$.
8. $\mathcal{P}$ and $\mathcal{V}$ invoke $\Pi_{\mathsf{com\text{-}mult}}$ to obtain commitments to the reverse of ρ_{m+1}, ρ_{2m+2} with respect to the challenge t_h and to ρ_m, ρ_{2m} with respect to the challenge t_{perm}.
9. $\mathcal{P}$ sends $w = \langle L_1, \mathbf{t'^m} \rangle$ to $\mathcal{V}$, where $\mathbf{t'^m} = (1\ t'\ t'^2\ \ldots\ t'^{m-1} \| \mathbf{0})$.
10. $\mathcal{V}$ sets $\mathcal{L}_1' = (\prod_{i=1}^M W_i^L)(\prod_{i=1}^M W_i^R)^u$. Eventually, we need a commitment to the reverse of the first m elements of the vector underlying $\mathcal{L}_1'$. This is accomplished in steps 10(d) and 10(e).
11. Set $V = (X^O)^z R, \tilde{V} = (X^L)^z (X^R)^{zu} \tilde{R}$. $\mathcal{V}$ checks if
    (a) $(\Sigma_0, \mathbf{t^{2m+2}}, \frac{(ct)^m - 1}{ct - 1}) \in \mathcal{R}_{\text{CLF-rev}}$
    (b) $(\Sigma_0, \Sigma_i^k, S_i^k) \in \mathcal{R}_{\mathsf{perm}}\ \forall i \in [M]\ \forall k \in \{L, R\}$
    (c) $(S_i^k, \Sigma_i^k, W_i^k) \in \mathcal{R}_{\mathsf{had}} \forall i \in [M]\ \forall k \in \{L, R\}$
    (d) $(\mathcal{L}_1', \mathbf{t'^m} \| \mathbf{0}, w) \in \mathcal{R}_{\text{CLF-rev}}$
    (e) $(\mathcal{L}_1^{rev}, \mathsf{rev}(\mathbf{t'^{2m+2}}), w) \in \mathcal{R}_{\text{CLF-rev}}$
    (f) $(V, \mathcal{L}_1^{rev}, v_1) \in \mathcal{R}$
    (g) $(\tilde{V}, \mathbf{c^m}, v_1 + zK) \in \mathcal{R}$
    (h) $(X^L, X^R, X^O) \in \mathcal{R}_{\mathsf{had}}$
    The checks in steps (c) and (h) use the commitments to ρ_{m+1}, ρ_{2m+2} obtained in Step 8, while the checks in step (b) use the commitments to ρ_m, ρ_{2m}. We don't need commitments to $\mathbf{t^{2m+2}}, \mathbf{t'^m}, \mathbf{c^m}$ in steps 11a, 11e and 11g because the verifier can compute a random linear combination of these vectors without help from the prover. Moreover, all the claims about the openings of linear forms made by the prover in Steps 8 and 11 can be aggregated using our protocol for batched linear form openings.

Fig. 8: Protocol Π_{csat} for Circuit Satisfiability

$\mathcal{ML}_n$, for $n = 2^\ell$,

$$\mathcal{ML}_n = \{\overline{\mathbf{a}} : \dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_\ell) \leftarrow_R \mathbb{Z}_q^\ell, \overline{\mathbf{a}} = (\prod_{i=1}^{\ell} \dot{a}_i^{x_i})_{x_i \in \{0,1\}}\}$$

We now consider a similar distribution over group elements,

$$\mathcal{ML}_n(\mathbb{G}) = \mathcal{ML}_{2^\ell}(\mathbb{G}) := \{g\overline{\mathbf{a}} : g \leftarrow_R \mathbb{G}, \dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_\ell) \leftarrow_R \mathbb{Z}_q^\ell, \overline{\mathbf{a}} = (\prod_{i=1}^{\ell} \dot{a}_i^{x_i})_{x_i \in \{0,1\}}\}$$

We define a new commitment scheme which differs from the one proposed in [LMR19] (and subsequently used in [ACR21]) only in that we sample the commitment key from $\mathcal{ML}_n(\mathbb{G})$.

Definition 2 (Commitment to $(\mathbb{Z}_q, \mathbb{G}_1, \mathbb{G}_2)$-vectors). *Let $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ be a bilinear group and $n_0, n_1, n_2 \geq 0$. We define a commitment scheme $\mathrm{COM}^{\mathbb{G}}$ for vectors in $\mathbb{Z}_q^{n_0} \times \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2}$, given by the following setup and commitment phase:*

- *Setup :* $(\mathbf{h}, \mathbf{g}) \leftarrow_R \mathcal{ML}^2_{n_0+1}(\mathbb{G}_T), \mathbf{H} \leftarrow_R \mathcal{ML}^2_{n_1}(\mathbb{G}_2), \mathbf{G} \leftarrow_R \mathcal{ML}^2_{n_2}(\mathbb{G}_1)$
  Here, $(\mathbf{h}, \mathbf{g}, \mathbf{H}, \mathbf{G}) = (\overline{\boldsymbol{a}}h, \overline{\boldsymbol{b}}g, \overline{\boldsymbol{c}}H, \overline{\boldsymbol{d}}G)$ for some structured $\mathbb{Z}_q$ vectors $\overline{\boldsymbol{a}}, \overline{\boldsymbol{b}}, \overline{\boldsymbol{c}}, \overline{\boldsymbol{d}}$, where h, g is sampled to be $h = e(h_1, H), g = e(g_1, H)$ for some $h_1, g_1 \leftarrow_R \mathbb{G}_1$.[2] Then, $(\mathsf{ck}_0 = ((\overline{\boldsymbol{a}}h_1, \overline{\boldsymbol{a}}h), (\overline{\boldsymbol{b}}g_1, \overline{\boldsymbol{b}}g)), \mathsf{ck}_1 = \overline{\boldsymbol{c}}H, \mathsf{ck}_2 = \overline{\boldsymbol{d}}G)$ are the commitment keys and $(\dot{\mathsf{ck}}_0 = (\overline{\boldsymbol{a}}H, \overline{\boldsymbol{b}}H), \dot{\mathsf{ck}}_1 = \overline{\boldsymbol{c}}G, \dot{\mathsf{ck}}_2 = \overline{\boldsymbol{d}}H)$ is the verification key.
- *Commit :* $\mathrm{COM}^{\mathbb{G}} : \mathbb{Z}_q^{n_0} \times \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2} \times \mathbb{Z}_q \to \mathbb{G}_T^2$,
  - $(\mathbf{x}, \mathbf{y}, \mathbf{z}; \gamma) \to \mathbf{h}\gamma + \langle \mathbf{g}, \mathbf{x} \rangle + e(\mathbf{y}, \mathbf{H}) + e(\mathbf{G}, \mathbf{z})$,
  - *where* $h\gamma + \langle \mathbf{g}, \mathbf{x} \rangle + e(\mathbf{y}, \mathbf{H}) + e(\mathbf{G}, \mathbf{z}) = \begin{pmatrix} h_1\gamma + \langle \mathbf{g}_1, \mathbf{x} \rangle + e(\mathbf{y}, \mathbf{H}_1) + e(\mathbf{G}_1, \mathbf{z}) \\ h_2\gamma + \langle \mathbf{g}_2, \mathbf{x} \rangle + e(\mathbf{y}, \mathbf{H}_2) + e(\mathbf{G}_2, \mathbf{z}) \end{pmatrix}$

The verification key is used to check that the commitment key has been updated by the prover, by having the prover send the first element of the commitment key ck to the verifier, and the verifier using the pairing check to ensure that the split-and-fold technique has been used correctly to update the commitment key and check that the updated commitment (sent by the prover) with respect to the updated commitment key is consistent.

We define an assumption called eGDLR assumption along the lines of GDLR assumption in [LMR19], show that it is implied by SXDH (Lemma 6) and prove binding of $\mathrm{COM}^{\mathbb{G}}$ under eGDLR.

Lemma 2. $\mathrm{COM}^{\mathbb{G}}$ *is computationally hiding under DDH in $\mathbb{G}_T$, and computationally binding under SXDH.*

The proof is presented in Lemma 4 and 5 in Appendix A.2.

5.2 Succinct Verifier Σ-Protocol for Opening Committed Homomorphism

Notation. Let $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ be a bilinear group. Let $g^{\overline{\mathbf{a}}} \in \mathbb{G}_1$ be the commitment key used to commit to a vector of $\mathbb{Z}_q$ elements in $\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}) = \langle \overline{\mathbf{a}}, \mathbf{x} \rangle g \in \mathbb{G}_1$, where $\mathbf{x} \in \mathbb{Z}_q^n, \overline{\mathbf{a}} \in \mathbb{Z}_q^n$. We consider the group homomorphism $f : \mathbb{Z}_q^n \to \mathbb{G}_2$, and define $\mathrm{HOM}(\mathbb{Z}_q^n, \mathbb{G}_2) = \{f : f \text{ is a homomorphism from } \mathbb{Z}_q^n \text{ to } \mathbb{G}_2\}$. We use $\mathrm{COM}^{\mathbb{G}}$ given in definition 2 and use a modified version to commit to element of only one source group of bilinear pairing as follows : $\mathrm{COM}^{\mathbb{G}} : \mathbb{G}_2^n \to \mathbb{G}_T$, where $\mathrm{COM}^{\mathbb{G}}(\mathbf{x}) = e(\mathbf{G}, \mathbf{x})$, for $n = 2^\ell, h \leftarrow_R \mathbb{G}_T, \dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_\ell) \leftarrow_R \mathbb{Z}_q^\ell, \overline{\mathbf{a}} = \left(\prod_{i=1}^{\ell} \dot{a}_i^{b_i}\right)_{b_i \in \{0,1\}}, \mathbf{G} = \overline{\mathbf{a}}G$, and we use the notation to $\mathrm{COM}^{\mathbb{G}}_{\overline{\mathbf{a}}}$ to explicitly specify the commitment key for ease of exposition, and define it as $\mathrm{COM}^{\mathbb{G}}_{\overline{\mathbf{a}}}(\mathbf{x}) = \mathrm{COM}^{\mathbb{G}}(\mathbf{x}) = e(\mathbf{G}, \mathbf{x})$, where $\mathbf{G} = \overline{\mathbf{a}}G$.

[2] We note that the distribution remains the same even when $\overline{\mathbf{a}}g_1$ is sampled from $\mathcal{ML}_{n_0}(\mathbb{G}_1)$ and g is then set to $g = e(g_1, H)$, making the final commitment key for $\mathbb{Z}_q$-vector to be $\mathbf{g} = \overline{\mathbf{a}}g$, as opposed to when $\mathbf{g}$ is directly sampled from $\mathcal{ML}_{n_0}(\mathbb{G}_T)$.

Opening group homomorphism. We aim to prove that a committed vector $\mathbf{x} \in \mathbb{Z}_q^n$ is opening of an element $y \in \mathbb{G}$ with respect to group homomorphism defined by $f : \mathbb{Z}_q^n \to \mathbb{G}_2$, i.e. the opening of a given commitment $\text{COM}_{\overline{\mathbf{a}}}(\mathbf{x})$, $\mathbf{x} \in \mathbb{Z}_q^n$ is such that $f(\mathbf{x}) = y$ for some $y \in \mathbb{G}_2$. We note that the homomorphism $f : \mathbb{Z}_q^n \to \mathbb{G}$ can be defined as $f \in \mathbb{G}_2^n$, and we extend the techniques discussed in Section 3. We use the commitment scheme from Definition 2 $\text{COM}^{\mathbb{G}} : \mathbb{Z}_q \times \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2} \to \mathbb{G}_T^2$ to succinctly commit to $f = (f_1, \ldots, f_n) \in \mathbb{G}_2^n$ and $\mathsf{rev}(f) = (f_n, \ldots, f_1) \in \mathbb{G}_2^n$ using the structured commitment key $g^{\overline{\mathbf{a}}} \in \mathbb{G}_1$ used to commit to the vector.

We note that while techniques of Section 3.2 for committed linear forms can extend to a committed homomorphism, there are some differences that we need to handle. First, the representation of a group homomorphism is given by group elements as opposed to field elements in linear forms, and this requires a commitment to group elements. Since the commitment scheme relies on SXDH, we cannot encode the commitment randomness in the second group anymore. This is however crucial to verify that the commitment key is updated correctly in each step of split-and-fold. This makes our protocol designated verifier since the encoding of the randomness is available only to the verifier and binding still holds under SXDH. We define the relation $\mathcal{R}$ for opening a group homomorphism f below, and then present the protocol $\Pi_{\mathsf{0\text{-}hom}}$ for relation $\mathcal{R}$.

$$\mathcal{R} = \{(P \in \mathbb{G}_1, f \in \text{HOM}(\mathbb{Z}_q^n, \mathbb{G}_2), y \in \mathbb{G}_2; \mathbf{x} \in \mathbb{Z}_q^n, \gamma \in \mathbb{Z}_q) : P = \text{COM}_{\overline{\mathbf{a}}}(\mathbf{x}; \gamma) \ \wedge \ f(\mathbf{x}) = y\}$$

Parameters

- Common parameters : $(P \in \mathbb{G}_1, f \in \text{HOM}(\mathbb{Z}_q^n, \mathbb{G}_2), y \in \mathbb{G}_2),\ P = \text{COM}_{\overline{\mathbf{a}}}(\mathbf{x}; \gamma), y = f(\mathbf{x})$
- $\mathcal{P}$'s input : $(\mathbf{x} \in \mathbb{Z}_q^n, \gamma \in \mathbb{Z}_q)$

Protocol

1. $\mathcal{P}$ samples $\mathbf{r} \leftarrow_R \mathbb{G}, \rho \leftarrow_R \mathbb{Z}_q$, computes $A = \text{COM}_{\overline{\mathbf{a}}}(\mathbf{r}; \rho),\ t = f(\mathbf{r})$ and sends A, t to $\mathcal{V}$.
2. $\mathcal{V}$ samples $c \leftarrow_R \mathbb{Z}_q$ and sends c to $\mathcal{P}$
3. $\mathcal{P}$ computes $\mathbf{z} = c\mathbf{x} + \mathbf{r}$ and $\phi = c\gamma + \rho$ and sends $\mathbf{z}, \phi$ to $\mathcal{V}$
4. $\mathcal{V}$ checks if $\text{COM}_{\overline{\mathbf{a}}}(\mathbf{z}; \phi) = A + cP$ and $f(\mathbf{z}) = cy + t$, outputs 1 if it holds, outputs 0 otherwise.

Fig. 9: Protocol $\Pi_{\mathsf{0\text{-}hom}}$ for relation $\mathcal{R}$

Theorem 10. *$\Pi_{\mathsf{0\text{-}hom}}$ (Fig 9) is a 3-move protocol for relation $\mathcal{R}$. It is perfectly complete, special honest-verifier zero-knowledge and computationally special sound.*

Note that this theorem follows from the fact that this protocol is identical to the one introduced in [ACR21], and the properties of the protocol relies on the hiding and binding of the commitment scheme which are satisfied by our commitment scheme 2 used here.

Now, we note that the last message sent in step 5 of $\Pi_{\mathsf{0\text{-}hom}}$ (Fig 9) along with the check computed by the verifier can be captured by the relations defined below, and we provide a Proof of Knowledge of the last message instead with the protocols for the following relation.

$$\mathcal{R}_{\text{CH}} = \{(P \in \mathbb{G}_1, Q \in \mathbb{G}_T, y \in \mathbb{G}_2, g \in \mathbb{G}_1; f, \mathbf{x} \in \mathbb{Z}_q^n) : \\ P = \text{COM}_{\overline{\mathbf{a}}}(\mathbf{x}) \ \wedge \ Q = \text{COM}^{\mathbb{G}}_{\overline{\mathbf{a}}}(f) \ \wedge \ f(\mathbf{x}) = y\}$$

Note that in the above, $P = \langle \overline{\mathbf{a}}, \mathbf{x} \rangle g \ \wedge \ Q = e(g\overline{\mathbf{a}}, f)$

We provide the protocol $\Pi_{\mathsf{1\text{-}hom}}$ for handling $\mathcal{R}_{\text{CH}}$ in Fig 10. Note that the protocol starts with having value of the common parameter intended as the first element of the commitment key as equal to the generator of the group, i.e. $g = G$, and it is later updated accordingly to encompass the commitment key updates in the protocol. We provide the proof of Theorem 11 in Appendix D.

Theorem 11. *$\Pi_{\mathsf{1\text{-}hom}}$ is a $(k_1, \ldots, k_\ell)$-move protocol for relation $\mathcal{R}_{CH}$, where $k_i = 3,\ \forall i \in [\ell], \ell = \log n$. It is perfectly complete and computationally special sound. It incurs total communication of $3 \log n$ $\mathbb{G}_1$ elements, $2 \log n + 2$ $\mathbb{G}_2$ elements, $2 \log n$ $\mathbb{G}_T$ elements, and $\log n + 2$ field elements.*

Parameters

- **Common parameters** : $(P \in \mathbb{G}_1, Q \in \mathbb{G}_T, y \in \mathbb{G}_2, g \in \mathbb{G}_1)$,
  - $P = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}), Q = \mathrm{COM}^{\mathbb{G}}_{\overline{\mathbf{a}}}(\mathsf{rev}(f)), y = f(\mathbf{x})$
  - $n = 2^\ell, \dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_\ell), \overline{\mathbf{a}} = \left(\prod_{i=1}^{\ell} \dot{a}_i^{b_i}\right)_{b_i \in \{0,1\}}$
  - $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ is a bilinear map.
- $\mathcal{P}$**'s input** : $(\overline{\mathbf{a}}g \in \mathbb{G}_1^n, \mathbf{x} \in \mathbb{Z}_q^n, f \in \mathrm{HOM}(\mathbb{Z}_q^n, \mathbb{G}_2))$
- $\mathcal{V}$**'s input** : $\dot{\mathbf{a}}H \in \mathbb{G}_2^\ell$

Protocol

1. Let us define k as $k = \mathsf{rev}(f)$. $\mathcal{P}$ parses $\mathbf{x} = (\mathbf{x}_L \| \mathbf{x}_R)$, $f = (f_L \| f_R)$ and $\overline{\mathbf{a}}g = (\overline{\mathbf{a}}_L g \| (\dot{a}_\ell \overline{\mathbf{a}}_L) g)$ and computes and sends the following to $\mathcal{V}$:
   (a) $A_1 = \mathrm{COM}_{\overline{\mathbf{a}}_R}(\mathbf{x}_L), A_2 = \mathrm{COM}_{\overline{\mathbf{a}}_L}(\mathbf{x}_R)$
   (b) $B_1 = \mathrm{COM}^{\mathbb{G}}_{\overline{\mathbf{a}}_R}(k_L), B_2 = \mathrm{COM}^{\mathbb{G}}_{\overline{\mathbf{a}}_L}(k_R)$
   (c) $y_1 = f_R(\mathbf{x}_L), y_2 = f_L(\mathbf{x}_R)$
2. $\mathcal{V}$ samples $c \leftarrow_R \mathbb{Z}_q$ and sends c to $\mathcal{P}$
3. $\mathcal{P}$ sets $\mathbf{x}' = \mathbf{x}_L + c\mathbf{x}_R$, $f' = cf_L \circ f_R$, $g' = (c + \dot{a}_\ell)g$ and implicitly sets $\dot{\mathbf{a}}' = (\dot{a}_1, \ldots, \dot{a}_{\ell-1})$ and $\overline{\mathbf{a}} = \overline{\mathbf{a}}_L$. Note that this also implicity sets $k' = k_L \circ ck_R$.
4. $\mathcal{P}$ sends g' to $\mathcal{V}$ and $\mathcal{V}$ checks the following, proceeds to step 5 if it holds, and aborts otherwise
$$e\left(\frac{g'}{cg}, H\right) = e\left(g, \dot{a}_\ell H\right)$$
5. $\mathcal{P}$ and $\mathcal{V}$ both compute the following :
$$P' = A_1 + cP + c^2 A_2,\ Q' = B_1 + cQ + c^2 B_2,\ y' = y_1 + cy + c^2 y_2$$
6. If $\mathbf{x}' \notin \mathbb{Z}_q^2$: $\mathcal{P}$ runs PoK Π_1 to prove knowledge of $\mathbf{x}', f'$ such that $\mathrm{COM}_{\overline{\mathbf{a}}'}(\mathbf{x}') = P'$, $\mathrm{COM}^{\mathbb{G}}_{\overline{\mathbf{a}}'}(k') = Q'$ and $f'(\mathbf{x}') = y'$.
   Hence, $\mathcal{P}$ and $\mathcal{V}$ runs the protocol Π_1 with updated common parameters (P', Q', y', g'), prover's input $(\overline{\mathbf{a}}'(g'), \mathbf{x}', f')$, and verifier's input $(\dot{\mathbf{a}}'H)$ for $(P', Q', y'; \mathbf{x}') \in \mathcal{R}_{\mathrm{CH}}$
7. If $\mathbf{x}' \in \mathbb{Z}_q^2$:
   (a) $\mathcal{P}$ sends $\mathbf{x}', f'$ to $\mathcal{V}$
   (b) $\mathcal{V}$ computes $k' = \mathsf{rev}(f')$ and checks the following :
$$\mathrm{COM}_{\overline{\mathbf{a}}'}(\mathbf{x}') = P' \ \wedge\ \mathrm{COM}^{\mathbb{G}}_{\overline{\mathbf{a}}'}(k') = Q' \ \wedge\ f'(\mathbf{x}') = y'$$
   and outputs 1 if it holds, and outputs 0 otherwise.

Fig. 10: Protocol $\Pi_{\text{1-hom}}$ for relation $\mathcal{R}_{\mathrm{CH}}$

We note that since we run the protocol $\Pi_{\text{1-hom}}$ as an alternative for steps 4 and 5 of $\Pi_{\text{0-hom}}$ to avoid having to send a linear-sized vector, $\Pi_{\text{1-hom}}$ does not require zero-knowledge property as the final message of the protocol $\Pi_{\text{0-hom}}$ is intended to be sent in clear. The compressed sigma protocol for proving knowledge of homomorphism on a committed vector is given by the compressed protocol $\Pi_{c\text{-hom}}$, which is defined by $\Pi_{c\text{-hom}} = \Pi_{\text{1-hom}} \circ \Pi_{\text{0-hom}}$. The compressed protocol for relation $\mathcal{R}$ is given by $\Pi_{c\text{-hom}}$, whose communication and computational complexities are dominated by that of $\Pi_{\text{1-hom}}$, and hence we obtain a designated verifier succinct argument of knowledge for the relation $\mathcal{R}$.

Proof of Knowledge of k-out-of-n discrete logarithms. The fundamental contribution of [ACF21] of proving Proof of Knowledge of k-out-of-n discrete logarithms (Protocol 3 of [ACF21]) relies on its ability to provide a compressed sigma protocol for opening a general homomorphism as a building block in a black box manner, and our techniques show how to do this with a succinct verifier. We expect that by relying on their techniques to amortize the protocol for opening multiple homomorphisms (which is done by using a challenge provided by the verifier to perform the check on a random linear combination of the homomorphisms) which is then deployed as a black box for the protocol to obtain proof of knowledge of k-out-of-n discrete logarithms, we can obtain a succinct verifier version of the proof in [ACF21].

5.3 Compressed Σ-Protocol for Opening General Homomorphisms

We now extend our protocol to opening homomorphisms on committed vectors with coefficients in multiple groups. We believe that using our protocols in applications of CSP to Threshold Signature Schemes and circuit zero-knowledge protocols with bilinear gates [ACR21] will result in analogs with succinct verifier after an appropriate preprocessing phase.

We first describe the Σ-Protocol of [ACR21], while using our updated commitment scheme with logarithmic verification, for proving knowledge of a witness $\mathbf{x}$ which opens a public homomorphism f to a public element y and opens the known commitment $\text{COM}^{\mathbb{G}}$ to a public element P, i.e. $y = f(\mathbf{x})$ and $P = \text{COM}^{\mathbb{G}}(\mathbf{x}, \gamma)$. Here, we assume $\mathbf{x} \in \mathbb{G}_S = \mathbb{Z}_q^{n_0} \times \mathbb{G}_1^{n_1} \times \mathbb{G}_k^{n_k}$, and we have access to a homomorphic commitment scheme $\text{COM}^{\mathbb{G}} : \mathbb{G}_S \times \mathbb{Z}_q^r \mapsto \mathbb{G}_{\mathcal{C}}$, and a public homomorphism $f : \mathbb{G}_S \mapsto \mathbb{Z}_q \times \mathbb{G}_1 \times \cdots \times \mathbb{G}_k$. The relation is given by $\mathcal{R} = \{(P, f, y; \mathbf{x}, \gamma) : P = \text{COM}^{\mathbb{G}}(\mathbf{x}, \gamma), y = f(\mathbf{x})\}$, and the POK $\Pi_{\text{0-gen-hom}}$ for $\mathcal{R}$ is in Fig 11).

Parameters

- Common parameters : $P = \text{COM}^{\mathbb{G}}(\mathbf{x}, \gamma), y = f(\mathbf{x})$
- $\mathcal{P}$'s input : $(\mathbf{x}, \gamma)$

Protocol

1. $\mathcal{P}$ samples $\mathbf{r} \leftarrow_R \mathbb{G}_S, \rho \leftarrow_R \mathbb{Z}_q$
2. $\mathcal{P}$ computes $A = \text{COM}^{\mathbb{G}}(\mathbf{r}, \rho), t = f(\mathbf{r})$ and sends it to $\mathcal{V}$
3. $\mathcal{V}$ samples $c \leftarrow_R \mathbb{Z}_q$ and sends it to $\mathcal{P}$
4. $\mathcal{P}$ computes $\mathbf{z} = c\mathbf{x} + \mathbf{r}$ and $\phi = c\gamma + \rho$ and sends it to $\mathcal{V}$
5. $\mathcal{V}$ checks if $\text{COM}^{\mathbb{G}}(\mathbf{z}, \phi) = A + cP$ and $f(\mathbf{z}) = cy + t$, outputs 1 if it holds, outputs 0 otherwise.

Fig. 11: POK $\Pi_{\text{0-gen-hom}}$ for $\mathcal{R}$ [ACR21]

In protocol $\Pi_{\text{0-gen-hom}}$, we note that step 4 renders the communication complexity linear, and that along with step 5 makes the verifier's complexity linear. We now reduce the complexities by running a compressing protocol $\Pi_{\text{1-gen-hom}}$ where we compress while relying on the compatibility of compression provided by the compactness of the commitment scheme for committing to the elements of the groups $\mathbb{Z}_q, \mathbb{G}_1, \ldots, \mathbb{G}_k$. In $\Pi_{\text{1-gen-hom}}$, compress the part of co-domain of $\text{COM}^{\mathbb{G}}$ which is compact, we parse $\text{COM}^{\mathbb{G}}$ as COM_1 and COM_2, where COM_1 contains the compressible (compact) co-domain of the commitment, and COM_2 contains the incompressible (non-compact) co-domain of the commitment. Hence, COM_1 is a compact commitment scheme and COM_2 is not (takes n-dimensional element to $n+1$-dimensional element). Hence, for some r_1 and r_2 such that $r_1 + r_2 = r$, we have

$$\begin{aligned}
&\text{COM}^{\mathbb{G}} : \mathbb{Z}_q^{n_0} \times \mathbb{G}_1^{n_1} \times \mathbb{G}_k^{n_k} \times \mathbb{Z}_q^r \mapsto \mathbb{G}_{\mathcal{C}} \\
&\text{COM}_1 : \mathbb{Z}_q^{n_0} \times \mathbb{G}_1^{n_1} \times \mathbb{G}_{k-1}^{n_{k-1}} \times \mathbb{Z}_q^{r_1} \mapsto \mathbb{G}_{\mathcal{C}_1} \\
&\text{COM}_2 : \mathbb{G}_k^{n_k} \times \mathbb{Z}_q^{r_2} \mapsto \mathbb{G}_{\mathcal{C}_2}
\end{aligned}$$

where size of $\mathbb{G}_{\mathcal{C}_1}$ is independent of the input dimensions in the domain, and size of $\mathbb{G}_{\mathcal{C}_1}$ is dependent on the input dimensions in the domain (increases by one with respect to the input dimensions in the domain).

We now implement the aforementioned idea by parsing the the witness $\mathbf{x}$ as $\mathbf{x} = (\mathbf{x}_S, \mathbf{x}_T)$ where $\mathbf{x}_S = (\mathbf{x}_0, \ldots, \mathbf{x}_{k-1})$ contains the compressible co-domain of the commitment, and $\mathbf{x}_T = \mathbf{x}_k$ contains the incompressible co-domain of the commitment. Since we want the verifier complexity to be sublinear, we also provide commitments to the homomorphism by treating the homomorphism description as a vector containing elements of $\mathbb{Z}_q, \mathbb{G}_1, \ldots, \mathbb{G}_k$. While we commit to the homomorphism f, we parse f as $f = (f_S, f_T)$, where f_S contains the compressible co-domain of the commitment, and f_T contains the incompressible co-domain of the commitment.

Notation for $\Pi_{\text{1-gen-hom}}$. We denote the commitment key for COM_1 which commits to elements of $\mathbb{Z}_q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_k$ by $\mathsf{ck}_0, \ldots, \mathsf{ck}_{k-1}$. For example, for $(\mathbf{x}, \mathbf{y}, \mathbf{x}_S; \gamma) \in \mathbb{Z}_q^{n_0} \times \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2}$, we have $\text{COM}_1(\mathbf{x}, \mathbf{y}, \mathbf{x}_S) =$

$\langle \mathbf{g}, \mathbf{x} \rangle + e(\mathbf{y}, \mathbf{H}) + e(\mathbf{G}, \mathbf{x}_S) \in \mathbb{G}_T^2$, we set $\mathsf{ck}_0 = \mathbf{g}$, $\mathsf{ck}_1 = \mathbf{H}$, $\mathsf{ck}_2 = \mathbf{G}$, and $(\dot{\mathsf{ck}}_0, \dot{\mathsf{ck}}_1, \dot{\mathsf{ck}}_2)$ is the verification key. For example, in $\Pi_{1\text{-hom}}$ described in section 5.2, ck denotes the commitment key held by the prover $\mathsf{ck} = g^{\overline{\mathbf{a}}}$ and $\dot{\mathsf{ck}}$ denotes the randomness encoded in the other group held by the verifier $\dot{\mathsf{ck}} = H^{\dot{\mathbf{a}}}$ where $n = 2^\ell, \dot{\mathbf{a}} = (\dot{a}_1, \ldots, \dot{a}_\ell)$, $\overline{\mathbf{a}} = \left(\prod_{i=1}^{\ell} \dot{a}_i^{b_i}\right)_{b_i \in \{0,1\}}$.

$$\begin{aligned}\mathcal{R}'_{CH} = \{(P, Q, y; \mathbf{x}, f) : \mathbf{x} = (\mathbf{x}_S, \mathbf{x}_T), y = (y_1, y_2, y_3, y_4), f = (f_S, f_T),\\ y_1 = f_S(\mathbf{x}_S), y_2 = f_S(\mathbf{x}_T), y_3 = f_T(\mathbf{x}_S), y_4 = f_T(\mathbf{x}_T),\\ Q = \mathrm{COM}_1(\mathsf{rev}(f_S)), P = (P_1, P_2), P_1 = \mathrm{COM}_1(\mathbf{x}_S), P_2 = \mathrm{COM}_2(\mathbf{x}_T)\}\end{aligned}$$

We present the PoK $\Pi_{1\text{-gen-hom}}$ for $\mathcal{R}'_{CH}$ in Fig 12. We provide the proof of Theorem 12 in Appendix D.

Theorem 12. *$\Pi_{1\text{-hom}}$ is a $(k_1, \ldots, k_\ell)$-move protocol for relation $\mathcal{R}_{CH}$, where $k_i = 3$, $\forall i \in [\ell], \ell = \log m$, $m = \max_{i=0}^{k-1} n_i$. It is perfectly complete and computationally special sound. It incurs total communication of $O(\log m)$ source (compressible) group elements (including $\mathbb{Z}_q$), $O(\log m + n_k)$ target group elements.*

Similar to earlier protocols, we aim to run the protocol $\Pi_{1\text{-gen-hom}}$ as an alternative for steps 4 and 5 of $\Pi_{0\text{-gen-hom}}$ to avoid having to send a linear-sized vector, $\Pi_{1\text{-gen-hom}}$ does not require zero-knowledge property as the final message of the protocol $\Pi_{0\text{-gen-hom}}$ is intended to be sent in clear. The compressed sigma protocol for proving knowledge of homomorphism on a committed vector is given by the compressed protocol $\Pi_{c\text{-gen-hom}}$, which is defined by :

$$\Pi_{c\text{-gen-hom}} = \Pi_{1\text{-gen-hom}} \circ \Pi_{0\text{-gen-hom}}$$

Hence, the compressed protocol for relation $\mathcal{R}$ is given by $\Pi_{c\text{-gen-hom}}$, whose communication and computational complexities are dominated by that of $\Pi_{1\text{-gen-hom}}$, hence we obtain a designated verifier succinct argument of knowledge for the relation $\mathcal{R}$.

Parameters

- **Common parameters** : $P, Q, y, \mathsf{ck}_{0,1}, \mathsf{ck}_{1,1}, \ldots, \mathsf{ck}_{k-1,1}$ (where $\mathsf{ck}_{i,1}$ is the first element of ck_i, $i = 0, \ldots, k-1$)
  - $\mathbf{x} = (\mathbf{x}_S, \mathbf{x}_T), y = (y_1, y_2, y_3, y_4), f = (f_S, f_T)$,
  - $y_1 = f_S(\mathbf{x}_S), y_2 = f_S(\mathbf{x}_T), y_3 = f_T(\mathbf{x}_S), y_4 = f_T(\mathbf{x}_T)$,
  - $Q = \mathrm{COM}_1(\mathsf{rev}(f_S)), P = (P_1, P_2), P_1 = \mathrm{COM}_1(\mathbf{x}_S), P_2 = \mathrm{COM}_2(\mathbf{x}_T)$
- $\mathcal{P}$**'s input** : $\mathsf{ck}_0, \mathsf{ck}_1, \ldots, \mathsf{ck}_{k-1}, \mathbf{x} = (\mathbf{x}_S, \mathbf{x}_T), \mathbf{x}_S = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1}), \mathbf{x}_T = (\mathbf{x}_k), f = (f_S, f_T)$
- $\mathcal{V}$**'s input** : $\dot{\mathsf{ck}}_0, \dot{\mathsf{ck}}_1, \ldots, \dot{\mathsf{ck}}_{k-1}$

Protocol

1. $\mathcal{P}$ parses $\mathbf{x}_i = (\mathbf{x}_{i,L} \| \mathbf{x}_{i,R})$, for $i = 0, \ldots, k$, $\mathbf{x}_{S,\alpha} = (\mathbf{x}_{0,\alpha}, \ldots, \mathbf{x}_{k-1,\alpha})$, and $\mathbf{x}_{T,\alpha} = (\mathbf{x}_{k,\alpha})$ for $\alpha = L, R$, and $f_S = (f_{S,L} \| f_{S,R}), f_T = (f_{T,L} \| f_{T,R})$.
2. Similarly, $\mathcal{P}$ parses the commitment keys for $\mathbf{x}_S$ and f_S as $\mathsf{ck}_S = (\mathsf{ck}_0, \ldots, \mathsf{ck}_{k-1})$ and commitment keys for $\mathbf{x}_T$ and f_T as ck_T.
3. $\mathcal{P}$ sets $k = \mathsf{rev}(f_S)$ and computes the following :
   (a) $A_1 = \mathrm{COM}_1(0, \mathbf{x}_{S,L}), A_2 = \mathrm{COM}_1(\mathbf{x}_{S,R}, 0)$
   (b) $B_1 = \mathrm{COM}_1(0, k_L), B_2 = \mathrm{COM}_1(k_R, 0)$
   (c) $a_1 = f_{S_R}(\mathbf{x}_{S,L}), a_2 = f_{S_L}(\mathbf{x}_{S,R})$
   (d) $b_1 = f_{S_R}(\mathbf{x}_{T,L}), b_2 = f_{S_L}(\mathbf{x}_{T,R})$
   (e) $d_1 = f_{T_R}(\mathbf{x}_{S,L}), d_2 = f_{T_L}(\mathbf{x}_{S,R})$
4. $\mathcal{P}$ sends the computed values a_2, b_2, A_1, A_2, B_1 and B_2 to $\mathcal{V}$
5. $\mathcal{V}$ samples $c \leftarrow_R \mathbb{Z}_q$ and sends it to $\mathcal{P}$
6. $\mathcal{P}$ sets the updated commitment key as $\mathsf{ck}'_i = c \cdot \mathsf{ck}_{i,L} + \mathsf{ck}_{i,R}$ for all $i = 0, 1, \ldots, k-1$
7. $\mathcal{P}$ sends the first element of all updated commitment keys $\mathsf{ck}'_{i,1}$ to $\mathcal{V}$, and $\mathcal{V}$ checks the following for each $\mathsf{ck}_{i,1}$, proceeds to step 8 if it holds, and aborts otherwise

$$e\left(\frac{\mathsf{ck}'_{i,1}}{\mathsf{ck}^c_{i,1}}, \mathsf{gen}_i\right) = e\left(\mathsf{ck}_{i,1}, \dot{\mathsf{ck}}_{i,\ell}\right), \quad i \in \{0, 1, \ldots, k-1\}$$

   where gen_i is the generator of the group containing $\dot{\mathsf{ck}}_i$.
8. $\mathcal{P}$ sets $\mathbf{x}'_S = \mathbf{x}_{S,L} + c\mathbf{x}_{S,R}$, $f'_S = cf_{S,L} + f_{S,R}$, and implicitly updates the randomness for each updated commitment key $\dot{\mathsf{ck}}'_i$ by dropping the last element $\dot{\mathsf{ck}}_{i,\ell}$ from $\dot{\mathsf{ck}}_i$.
9. $\mathcal{P}$ and $\mathcal{V}$ both compute the following :
   (a) $P'_1 = A_1 + cP_1 + c^2 A_2$, $Q' = B_1 + cQ + c^2 B_2$
   (b) $y'_1 = a_1 + cy_1 + c^2 a_2, y'_4 = y_4$,
   (c) $y'_2 = b_1 + cy_2 + c^2 b_2$, $y'_3 = d_1 + cy_3 + c^2 d_2$
10. If $\mathbf{x}'_S$ contains more than 2 elements from any group : $\mathcal{P}$ and $\mathcal{V}$ runs the protocol $\Pi_{\mathsf{1\text{-}gen\text{-}hom}}$ with updated common parameters $(P', Q', y'), P' = (P'_1, P_2), y' = (y'_1, y'_2, y'_3, y'_4)$, prover's input $(\mathsf{ck}'_0, \mathsf{ck}'_1, \ldots, \mathsf{ck}'_{k-1}, \mathbf{x}' = (\mathbf{x}'_S, \mathbf{x}_T), f' = (f'_S, f_T)$, and verifier's input $(\mathsf{ck}_{0,1}, \mathsf{ck}_{1,1}, \ldots, \mathsf{ck}_{k-1,1}, \dot{\mathsf{ck}}_1, \ldots, \dot{\mathsf{ck}}_{k-1})$ for $(P', Q', y'; \mathbf{x}', f') \in \mathcal{R}'_{CH}$
11. Otherwise :
   (a) $\mathcal{P}$ sends $\mathbf{x}' = (\mathbf{x}'_S, \mathbf{x}_T), f' = (f'_S, f_T)$ to $\mathcal{V}$
   (b) $\mathcal{V}$ computes $k' = \mathsf{rev}(f'_S)$ and checks the following where COM'_1 is the commitment with updated commitment keys $\mathsf{ck}'_i, i = 0, \ldots, k-1$
      i. $\mathrm{COM}'_1(\mathbf{x}'_S) = P'_1$, $\mathrm{COM}'_1(k') = Q'$, $\mathrm{COM}_2(\mathbf{x}_T) = P_2$
      ii. $f'_S(\mathbf{x}'_S) = y'_1$, $f_T(\mathbf{x}_T) = y_4$
      iii. $(f'_S, cf'_S)(\mathbf{x}_T) = y'_2$, $f_T(c\mathbf{x}'_S, \mathbf{x}'_S) = y'_3$

   and outputs 1 if it holds, and outputs 0 otherwise.

Fig. 12: PoK of $\Pi_{\mathsf{1\text{-}gen\text{-}hom}}$ for $\mathcal{R}'_{CH}$

References

AC20. Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 513–543. Springer, Heidelberg, August 2020.

ACF21. Thomas Attema, Ronald Cramer, and Serge Fehr. Compressing proofs of k-out-of-n partial knowledge. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 65–91, Virtual Event, August 2021. Springer, Heidelberg.

ACK21. Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed Σ-protocol theory for lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 549–579, Virtual Event, August 2021. Springer, Heidelberg.

ACR21. Thomas Attema, Ronald Cramer, and Matthieu Rambaud. Compressed Σ-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 526–556. Springer, Heidelberg, December 2021.

AFG+10. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010.

AGL+23. Arasu Arun, Chaya Ganesh, Satya Lokam, Tushar Mopuri, and Sriram Sridhar. Dew: A transparent constant-sized polynomial commitment scheme. In *Public-Key Cryptography – PKC 2023: 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7–10, 2023, Proceedings, Part II*, page 542–571, Berlin, Heidelberg, 2023. Springer-Verlag.

AGR+16. Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 191–219. Springer, Heidelberg, December 2016.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.

BCC+16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.

BCI+13. Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013.

BCTV14. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 781–796. USENIX Association, August 2014.

BFS20. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020.

BG12. Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280. Springer, Heidelberg, April 2012.

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

DRZ20. Vanesa Daza, Carla Ràfols, and Alexandros Zacharakis. Updateable inner product argument with logarithmic verifier and applications. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 527–557. Springer, Heidelberg, May 2020.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

GKM+18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://ia.cr/2019/953.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732, 1992.

Lee21. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography*, pages 1–34, Cham, 2021. Springer International Publishing.

Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012.

Lip13. Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, December 2013.

LMR19. Russell W. F. Lai, Giulio Malavolta, and Viktoria Ronge. Succinct arguments for bilinear group arithmetic: Practical structure-preserving cryptography. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2057–2074. ACM Press, November 2019.

LSZ23. Helger Lipmaa, Janno Siim, and Michał Zając. Counting vampires: From univariate sumcheck tonbsp;updatable zk-snark. In *Advances in Cryptology – ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part II*, page 249–278, Berlin, Heidelberg, 2023. Springer-Verlag.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

Mic94. Silvio Micali. Cs proofs. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 436–453. IEEE, 1994.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

SW05. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

zkh. https://www.zellic.io/blog/zk-friendly-hash-functions.

Appendix

A More Preliminaries

A.1 Proof of Lemma 1

Lemma 3. *The commitment scheme in Definition 1 is perfectly hiding and computationally binding under the DLOG assumption.*

Proof. The prover is given the (public) commitment key $g^{\overline{\mathbf{a}}}$ where $\overline{\mathbf{a}} = \{\overline{a}_1, \ldots, \overline{a}_{n+1}\}$, i.e. $g^{\overline{\mathbf{a}}} = (g^{\overline{a}_1}, \ldots, g^{\overline{a}_{n+1}})$.

Hiding. We note that, given a commitment P to any $\mathbb{Z}_q$ vector, any vector of elements of $\mathbb{Z}_q$, $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, could have been the element that is chosen to compute P, if the uniformly randomly chosen randomness γ for obtaining the commitment is such that $g^{\langle \overline{\mathbf{a}}, (\mathbf{x} \| \gamma) \rangle} = P$ i.e. γ satisfies :

$$\left(g^{\overline{a}_{n+1}}\right)^{\gamma} = \frac{P}{\left(g^{\overline{a}_1}\right)^{x_1} \cdots \left(g^{\overline{a}_n}\right)^{x_n}}$$

Hence, the aforementioned scheme is perfectly hiding.

Binding. If the binding of this commitment scheme is broken, then we have $\mathbf{x} = (x_1, \ldots, x_n), \gamma$ and $\mathbf{y} = (y_1, \ldots, y_n), \delta$ such that $\mathbf{x} \neq \mathbf{y}$ and

$$\begin{aligned} \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}; \gamma) &= \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{y}; \delta) \\ \implies g^{\langle \overline{\mathbf{a}}, (\mathbf{x} \| \gamma) \rangle} &= g^{\langle \overline{\mathbf{a}}, (\mathbf{y} \| \delta) \rangle} \\ \implies g^{\langle \overline{\mathbf{a}}, (\mathbf{x}-\mathbf{y} \| \gamma-\delta) \rangle} &= 1_{\mathbb{G}_1} \\ \implies \left(g^{\overline{a}_1}\right)^{(x_1-y_1)} \cdots \left(g^{\overline{a}_n}\right)^{(x_n-y_n)} \cdot \left(g^{\overline{a}_{n+1}}\right)^{(\gamma-\delta)} &= 1_{\mathbb{G}_1} \end{aligned}$$

which breaks the (extended) discrete logarithm assumption.

A.2 Hardness Assumption

We recall the GDLR assumption from [LMR19].

Definition 3 (Generalized Discrete Logarithm Representation Assumption [LMR19]).

Let $m \geq 1$ and $n_1, n_2, n_T \geq 0$ (not all zero). (m, n_T, n_1, n_2)-GDLR assumption holds in $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ if for any PPT algorithm $\mathcal{A}$, we have

$$\Pr\left[\begin{array}{c} e(\mathbf{a}_1, \mathbf{B}_2) + e(\mathbf{B}_1, \mathbf{a}_2) + \langle \mathbf{B}_T, \mathbf{a}_T \rangle = \mathbf{0}_T \wedge (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_T) \neq (\mathbf{0}, \mathbf{0}, \mathbf{0}) \\ \overline{\boldsymbol{b}}_1 \leftarrow_R \mathbb{Z}_q^{m \times n_2}, \overline{\boldsymbol{b}}_2 \leftarrow_R \mathbb{Z}_q^{m \times n_1}, \overline{\boldsymbol{b}}_T \leftarrow_R \mathbb{Z}_q^{m \times n_T} \\ \mathbf{B}_1 = G\overline{\boldsymbol{b}}_1, \mathbf{B}_2 = H\overline{\boldsymbol{b}}_2, \mathbf{B}_T = K\overline{\boldsymbol{b}}_T \\ (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_T) \leftarrow \mathcal{A}(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H, \mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_T) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathbf{a}_1 \in \mathbb{G}_1^{n_2}, \mathbf{a}_2 \in \mathbb{G}_2^{n_1}, \mathbf{a}_T \in \mathbb{Z}_q^{n_T}$.

We now introduce the eGDLR assumption.

Definition 4 (extended Generalized Discrete Logarithm Representation Assumption). *Let $m \geq 1$ and $n_1, n_2, n_T \geq 0$ (not all zero). $(m, n_T, n_1, n_2)-$eGDLR assumption holds in $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ if for any PPT $\mathcal{A}$, we have*

$$\Pr\left[\begin{array}{c} e(\mathbf{a}_1, \mathbf{B}_2) + e(\mathbf{B}_1, \mathbf{a}_2) + \langle \mathbf{B}_T, \mathbf{a}_T \rangle = \mathbf{0}_T \wedge (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_T) \neq (\mathbf{0}, \mathbf{0}, \mathbf{0}) \\ \overline{\boldsymbol{b}}_1 \leftarrow_R \mathcal{ML}_{n_1}^m, \overline{\boldsymbol{b}}_2 \leftarrow_R \mathcal{ML}_{n_2}^m, \dot{\boldsymbol{b}}_T \leftarrow_R \mathcal{ML}_{n_0}^m \\ \mathbf{B}_1 = G\overline{\boldsymbol{b}}_1, \mathbf{B}_2 = H\overline{\boldsymbol{b}}_2, \mathbf{B}_T = K\overline{\boldsymbol{b}}_T \\ (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_T) \leftarrow \mathcal{A}(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H, \mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_T) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathbf{a}_1 \in \mathbb{G}_1^{n_2}, \mathbf{a}_2 \in \mathbb{G}_2^{n_1}, \mathbf{a}_T \in \mathbb{Z}_q^{n_T}$.

Lemma 4. $\mathrm{COM}^{\mathbb{G}}$ *(Definition 2) is computationally binding under eGDLR assumption.*

Proof. If binding of the aforementioned commitment scheme is broken, then we get $\mathbf{x}, \mathbf{y}, \mathbf{z}, \gamma$ and $\mathbf{x}', \mathbf{y}', \mathbf{z}', \gamma'$ where $\mathbf{x} \neq \mathbf{x}'$ or $\mathbf{y} \neq \mathbf{y}'$ or $\mathbf{z} \neq \mathbf{z}'$ or $\gamma \neq \gamma'$, such that $\mathrm{COM}^{\mathbb{G}}(\mathbf{x}, \mathbf{y}, \mathbf{z}; \gamma) = \mathrm{COM}^{\mathbb{G}}(\mathbf{x}', \mathbf{y}', \mathbf{z}'; \gamma')$

$$\implies \begin{pmatrix} h_1\gamma + \langle \mathbf{g}_1, \mathbf{x} \rangle + e(\mathbf{y}, \mathbf{H}_1) + e(\mathbf{G}_1, \mathbf{z}) \\ h_2\gamma + \langle \mathbf{g}_2, \mathbf{x} \rangle + e(\mathbf{y}, \mathbf{H}_2) + e(\mathbf{G}_2, \mathbf{z}) \end{pmatrix} = \begin{pmatrix} h_1\gamma' + \langle \mathbf{g}_1, \mathbf{x}' \rangle + e(\mathbf{y}', \mathbf{H}_1) + e(\mathbf{G}_1, \mathbf{z}') \\ h_2\gamma' + \langle \mathbf{g}_2, \mathbf{x}' \rangle + e(\mathbf{y}', \mathbf{H}_2) + e(\mathbf{G}_2, \mathbf{z}') \end{pmatrix}$$

$$\implies \begin{pmatrix} h_1(\gamma - \gamma') + \langle \mathbf{g}_1, (\mathbf{x} - \mathbf{x}') \rangle + e((\mathbf{y} - \mathbf{y}'), \mathbf{H}_1) + e(\mathbf{G}_1, (\mathbf{z} - \mathbf{z}')) \\ h_2(\gamma - \gamma') + \langle \mathbf{g}_2, (\mathbf{x} - \mathbf{x}') \rangle + e((\mathbf{y} - \mathbf{y}'), \mathbf{H}_2) + e(\mathbf{G}_2, (\mathbf{z} - \mathbf{z}')) \end{pmatrix} = \begin{pmatrix} \mathbf{0}_T \\ \mathbf{0}_T \end{pmatrix}$$

which breaks the $(2, n_0 + 1, n_1, n_2)$-eGDLR assumption.

Lemma 5. $\mathrm{COM}^{\mathbb{G}}$ *(Definition 2) is computationally hiding under DDH assumption in* $\mathbb{G}_T$*.*

Proof. Hiding of $\mathrm{COM}^{\mathbb{G}}$ follows from the fact that $(h_1, h_2, h_1^\gamma, h_2^\gamma)$, where $h_1, h_2 \in \mathbb{G}_T$ and $\gamma \leftarrow_R \mathbb{Z}_q$, is computationally indistinguishable from $(h_1, h_2, h_1^\gamma, r)$, where $h_1, h_2 \in \mathbb{G}_T$ and $\gamma \leftarrow_R \mathbb{Z}_q$, $r \leftarrow_R \mathbb{G}_T$, when DDH holds in $\mathbb{G}_T$.

We construct a DDH adversary $\mathcal{A}$ for $\mathbb{G}_T$ given that we have a distinguisher $\mathcal{B}$ which distinguishes $(h_1, h_2, h_1^\gamma, h_2^\gamma)$ from $(h_1, h_2, h_1^\gamma, r)$, where $h_1, h_2 \in \mathbb{G}_T$ and $\gamma \leftarrow_R \mathbb{Z}_q$, $r \leftarrow_R \mathbb{G}_T$.

1. $\mathcal{A}$ receives a DDH challenge $ch = (g, g^a, g^b, g^c)$
2. $\mathcal{A}$ sends the challenge vector ch to $\mathcal{B}$
3. If $\mathcal{B}$ outputs that ch is of the form $(h_1, h_2, h_1^\gamma, h_2^\gamma)$, then $\mathcal{A}$ outputs $c = ab$; otherwise $\mathcal{A}$ outputs $c \neq ab$.

$\mathcal{A}$ succeeds with overwhelming probability, if $\mathcal{B}$ does, which follows from the following observation :

- if $c = ab$, then the challenge vector $ch = (g, g^a, g^b, g^c) = (h_1, h_2, h_1^\gamma, h_2^\gamma)$ where $h_1 = g, h_2 = g^a, \gamma = b$, and
- if $c \neq ab$ and $c \in_R \mathbb{Z}_q$, then $ch = (g, g^a, g^b, g^c) = (h_1, h_2, h_1^\gamma, r)$, where $h_1 = g, h_2 = g^a, \gamma = b$ and $r = g^c$.

Hence, we have that $(h_1, h_2, h_1^\gamma, h_2^\gamma)$ and $(h_1, h_2, h_1^\gamma, r)$ are computationally indistinguishable under DDH in $\mathbb{G}_T$, where $h_1, h_2 \in \mathbb{G}_T$ and $\gamma \leftarrow_R \mathbb{Z}_q$, $r \leftarrow_R \mathbb{G}_T$. From the above property, we note that use of h_1^γ, h_2^γ to re-randomize the two components of $\mathrm{COM}^{\mathbb{G}}$ is indistinguishable from using completely random elements to re-randomize the same, and hence $\mathrm{COM}^{\mathbb{G}}$ is hiding under DDH in $\mathbb{G}_T$.

We now show that eGDLR is implied by SXDH.

Lemma 6. *Let q be such that $1/q = \mathsf{negl}$. Let $m = 2$; $n_i \geq 0, i = 0, 1, 2$ are not all zero. Then, the (m, n_0, n_1, n_2)-eGDLR assumption holds if the SXDH assumption holds.*

Proof. We know that, for q where $1/q = \mathsf{negl}$ and $m \geq 2$; $n_i \geq 0, i = 0, 1, 2$ are not all zero, (m, n_0, n_1, n_2)-GDLR assumption holds, if the SXDH assumption holds [LMR19]. From the previous statement, we can infer that SXDH assumption implies that $(2, 1, 1, 1)$-GDLR assumption holds. Now, we wish to prove that, for $m = 2$, (m, n_0, n_1, n_2)-eGDLR assumption holds if $(2, 1, 1, 1)$-GDLR assumption holds.

We construct an adversary $\mathcal{A}$ for $(2, 1, 1, 1)$-GDLR assumption, given an adversary $\mathcal{B}$ for $(2, n_0, n_1, n_2)$-eGDLR assumption, as follows :

1. $\mathcal{A}$ receives a challenge for $(2, 1, 1, 1)$-GDLR assumption,
   $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H, (g_1, h_1), (g_2, h_2), (g_T, h_T))$
   such that $g_i, h_i \in \mathbb{G}_i, i \in \{1, 2, T\}$
2. $\mathcal{A}$ samples the keys for obtaining challenges for $\mathcal{B}$: $\bar{\mathbf{r}} \leftarrow_R \mathcal{ML}_{n_0}$, $\bar{\mathbf{s}} \leftarrow_R \mathcal{ML}_{n_1}$, and $\bar{\mathbf{t}} \leftarrow_R \mathcal{ML}_{n_2}$
3. $\mathcal{A}$ sends $\mathcal{B}$ the challenge $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H, (g_1\bar{\mathbf{r}}, h_1\bar{\mathbf{r}}), (g_2\bar{\mathbf{s}}, h_2\bar{\mathbf{s}}), (g_T\bar{\mathbf{t}}, h_T\bar{\mathbf{t}}))$
4. $\mathcal{B}(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H, (g_1\bar{\mathbf{r}}, h_1\bar{\mathbf{r}}), (g_2\bar{\mathbf{s}}, h_2\bar{\mathbf{s}}), (g_T\bar{\mathbf{t}}, h_T\bar{\mathbf{t}})) \rightarrow (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_T)$

5. $\mathcal{A}$ computes $x_1 = \langle \mathbf{a}_1, \overline{\mathbf{r}} \rangle, x_2 = \langle \mathbf{a}_2, \overline{\mathbf{s}} \rangle$ and $x_T = \langle \overline{\mathbf{t}}, \mathbf{a}_T \rangle$ and outputs (x_1, x_2, x_T), where the first two operations are inner product with scalar for groups $\mathbb{G}_2$ and $\mathbb{G}_1$, and the third operation is inner product of elements of $\mathbb{Z}_q$.

We claim that $\mathcal{A}$ succeeds with overwhelming probability, if $\mathcal{B}$ does.

We note that, if $\mathcal{B}$ succeeds, then its output $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_T)$ is such that $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_T) \neq (0, 0, 0)$ and

$$g_T \langle \overline{\mathbf{t}}, \mathbf{a}_T \rangle + e(g_1 \overline{\mathbf{r}}, \mathbf{a}_1) + e(\mathbf{a}_2, g_2 \overline{\mathbf{s}}) = 0 \text{ , and } h_T \langle \overline{\mathbf{t}}, \mathbf{a}_T \rangle + e(h_1 \overline{\mathbf{r}}, \mathbf{a}_1) + e(\mathbf{a}_2, h_2 \overline{\mathbf{s}}) = 0$$

hence we have, for $x_1 = \langle \mathbf{a}_1, \overline{\mathbf{r}} \rangle, x_2 = \langle \mathbf{a}_2, \overline{\mathbf{r}} \rangle$ and $x_T = \langle \overline{\mathbf{t}}, \mathbf{a}_T \rangle$:

$$g_T x_T + e(g_1, x_1) + e(x_2, g_2) = 0 \text{ , and } h_T x_T + e(h_1, x_1) + e(x_2, h_2) = 0$$

We analyse that, $\mathcal{A}$'s breaks the assumption if $\mathcal{B}$ does, by showing that along with the satisfied equation, $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_T) \neq (0, 0, 0)$ ensures $(x_1, x_2, x_T) \neq (0, 0, 0)$, except with negligible probability.

If $\mathbf{a}_T \neq 0$, then we have $x_T = \langle \overline{\mathbf{t}}, \mathbf{a}_T \rangle \neq 0$ w.h.p. as otherwise we have $g_T \langle \overline{\mathbf{t}}, \mathbf{a}_T \rangle = 0$ which breaks dlog in $\mathbb{G}_T$, hence it ensures $(x_1, x_2, x_T) \neq (0, 0, 0)$.

If $\mathbf{a}_1 \neq 0$, and if $x_1 = \langle \mathbf{a}_1, \overline{\mathbf{r}} \rangle = e_2$, then we have $e(\langle g_1, \overline{\mathbf{r}} \rangle, \mathbf{a}_1) = e(g_1, \langle \mathbf{a}_1, \overline{\mathbf{r}} \rangle) = e(g_1, e_2) = e(g_1, q e_2) = e(q g_1, e_2) = e(e_1, e_2) = e_T$, which breaks the $(e)n$-BP assumption that holds when SXDH holds, as DDH is hard in $\mathbb{G}_1$.

Similarly, we can argue for x_2 being non-zero, when $\mathbf{a}_2$ is non-zero.

Definition 5 (n-BP Assumption). *For all non-uniform PPT Adversary $\mathcal{A}$,*

$$\Pr \left[\begin{array}{c} e(\boldsymbol{X}, \boldsymbol{Y}) = e_T, \boldsymbol{X} \in \mathbb{G}_1^n, \boldsymbol{Y} \in \mathbb{G}_2^n \\ \mathbf{x} \leftarrow_R \mathbb{Z}_q^n, \boldsymbol{X} = G\mathbf{x} \\ \boldsymbol{Y} \leftarrow \mathcal{A}(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H, \boldsymbol{X}) \end{array} \right] = \mathsf{negl}(\lambda)$$

Definition 6 ($(e)n$-BP Assumption). *For all non-uniform PPT Adversary $\mathcal{A}$,*

$$\Pr \left[\begin{array}{c} e(\boldsymbol{X}, \boldsymbol{Y}) = e_T, \boldsymbol{X} \in \mathbb{G}_1^n, \boldsymbol{Y} \in \mathbb{G}_2^n \\ \mathbf{x} \leftarrow_R \mathcal{ML}_n, \boldsymbol{X} = G\mathbf{x} \\ \boldsymbol{Y} \leftarrow \mathcal{A}(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H, \boldsymbol{X}) \end{array} \right] = \mathsf{negl}(\lambda)$$

Lemma 7. *Let q be such that $1/q = \mathsf{negl}$. n-BP Assumption, for $n \in \mathbb{N}$, holds when DDH is hard in $\mathbb{G}_1$.*

Proof. We construct a DDH adversary $\mathcal{A}$ of $\mathbb{G}_1$, given a n-BP adversary $\mathcal{B}$, as follows:

1. $\mathcal{A}$ receives a DDH-challenge $(g, g \cdot a, g \cdot b, g \cdot c)$ of $\mathbb{G}_1$
2. $\mathcal{A}$ samples $\mathbf{r} = (r_1, \ldots, r_{n-1}) \leftarrow_R \mathbb{Z}_q^{n-1}$, and sets $\mathbf{x} = (x_0, \ldots, x_{n-1})$, where $x_0 = c$ and $x_i = a r_i, i = 1, \ldots, n-1$
3. $\mathcal{A}$ computes $\mathbf{X} = G\mathbf{x}$ and sends $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H, \mathbf{X})$ to $\mathcal{B}$
4. $\mathcal{A}$ receives $\mathbf{Y}$ from $\mathcal{B}$
5. $\mathcal{A}$ sets $\mathbf{z} = (z_0, \ldots, z_{n-1})$, where $z_0 = b$ and $z_i = r_i, i = 1, \ldots, n-1$, and sets $\mathbf{Z} = G\mathbf{z}$
6. If $e(\mathbf{X}, \mathbf{Y}) = e_T$ and $e(\mathbf{Z}, \mathbf{Y}) = e_T$, then $\mathcal{A}$ outputs 1, otherwise it outputs 0.

We claim that $\mathcal{A}$ succeeds with overwhelming probability, if $\mathcal{B}$ does.

We note that if $\mathcal{B}$ succeeds then we have $e(\mathbf{X}, \mathbf{Y}) = e_T$, which implies, assuming $\mathbf{Y} = H\mathbf{y}$, where $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in \mathbb{Z}_q^n$

$$e(G\mathbf{x}, H\mathbf{y}) = e_T \implies e(G \cdot x_1, H \cdot y_1) \cdots e(G \cdot x_{n-1}, H \cdot y_{n-1}) = e_T \implies e(G, H)\langle \mathbf{x}, \mathbf{y} \rangle = e_T$$

which gives us $\langle x, y \rangle = 0$.

If $c = ab$, then we have that $cy_0 + ar_1y_1 + \cdots + ar_{n-1}y_{n-1} = 0 \implies aby_0 + ar_1y_1 + \cdots + ar_{n-1}y_{n-1} = 0 \implies by_0 + r_1y_1 + \cdots + r_{n-1}y_{n-1} = 0 \implies e(\mathbf{Z}, \mathbf{Y}) = e_T$, then the adversary outputs 1.

If $c \neq ab$, then the adversary outputs 0 with high probability, as $c \neq ab$ and output 1 by adversary $\implies cy_0 + ar_1y_1 + \cdots + ar_{n-1}y_{n-1} = 0$ and $by_0 + r_1y_1 + \cdots + r_{n-1}y_{n-1} = 0$, both equations are independently satisfied by $\mathbf{y}$, which happens with probability $O(1/q)$.

Lemma 8. *Let q be such that $1/q = \mathsf{negl}$. $(e)n$-BP Assumption holds when n-BP Assumption holds for all $n \in \mathbb{N}$.*

Proof. We construct an adversary $\mathcal{A}$ for n-BP Assumption, given an adversary $\mathcal{B}$ for the $(e)n$-BP Assumption, as follows :

- $\mathcal{A}$ receives a challenge for n-BP Assumption $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H, \mathbf{X})$ where $\mathbf{X} = (x_1, \ldots, x_n)$
- $\mathcal{A}$ samples the keys from obtaining challenges for $\mathcal{B} : \bar{\mathbf{r}}_i \leftarrow_R \mathcal{ML}_{n'}$, and computes $x_i\bar{\mathbf{r}}$, for all $i \in [n]$
- $\mathcal{A}$ sends $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H, x_i\bar{\mathbf{r}})$ to $\mathcal{B}$, for all $i \in [n]$
- $\mathcal{B}$ returns $\mathbf{Z}_i \in \mathbb{G}_2^{n'}$, for all $i \in [n]$
- $\mathcal{A}$ computes $\mathbf{Y} = (y_1, \ldots, y_n)$ such that $y_i = \langle \mathbf{Z}_i, \bar{\mathbf{r}} \rangle \in \mathbb{G}_2$, for all $i \in [n]$
- $\mathcal{A}$ outputs $\mathbf{Y}$

We claim that $\mathcal{A}$ succeeds with overwhelming probability if $\mathcal{B}$ does. Note that if $\mathcal{B}$ succeeds for all $i \in [n]$, then its output $\mathbf{Z}_i$ satisfies

$$e(x_i\bar{\mathbf{r}}, \mathbf{Z}_i) = e_T, \text{ for all } i \in [n]$$

which gives us $e(x_i, \langle \mathbf{Z}_i, \bar{\mathbf{r}} \rangle) = e_T$. Hence, we have that $e(x_i, y_i) = e_T$, for all $i \in [n]$, and $\mathcal{A}$ succeeds since $e(\mathbf{X}, \mathbf{Y}) = e_T$.

B Proofs from Section 3

B.1 Proof of Theorem 2

Completeness. If the protocol is correctly executed by the prover $\mathcal{P}$, then we have

1. $A_1 = \mathrm{COM}_{\bar{\mathbf{a}}_L}(\mathbf{x}_L) = g^{\langle \bar{\mathbf{a}}_L, \mathbf{x}_L \rangle}$, $A_2 = \mathrm{COM}_{\bar{\mathbf{a}}_L}(\mathbf{x}_R) = g^{\langle \bar{\mathbf{a}}_L, \mathbf{x}_R \rangle}$
2. $B_1 = \mathrm{COM}_{\bar{\mathbf{a}}_L}(L_L) = g^{\langle \bar{\mathbf{a}}_L, L_L \rangle}$, $B_2 = \mathrm{COM}_{\bar{\mathbf{a}}_L}(L_R) = g^{\langle \bar{\mathbf{a}}_L, L_R \rangle}$
3. $y_1 = \langle \mathbf{x}_L, L_R \rangle$, $y_2 = \langle \mathbf{x}_R, L_L \rangle$
4. $\mathbf{x}' = \mathbf{x}_L + c\mathbf{x}_R$, $L' = cL_L + L_R$

Hence, the following verifier checks are satisfied as shown below :

$$\begin{aligned} e\left(\frac{P}{A_1}, H\right) &= e\left(\frac{g^{\langle \bar{\mathbf{a}}, \mathbf{x} \rangle}}{g^{\langle \bar{\mathbf{a}}_L, \mathbf{x}_L \rangle}}, H\right) \\ &= e(g^{\langle \bar{\mathbf{a}}_R, \mathbf{x}_R \rangle}, H) \\ &= e(g^{\langle \dot{a}_\ell \bar{\mathbf{a}}_L, \mathbf{x}_R \rangle}, H) \\ &= e(g^{\langle \bar{\mathbf{a}}_L, \mathbf{x}_R \rangle}, H^{\dot{a}_\ell}) \\ &= e(A_2, , H^{\dot{a}_\ell}) \end{aligned} \qquad \begin{aligned} e\left(\frac{Q}{B_1}, H\right) &= e\left(\frac{g^{\langle \bar{\mathbf{a}}, L \rangle}}{g^{\langle \bar{\mathbf{a}}_L, L_L \rangle}}, H\right) \\ &= e(g^{\langle \bar{\mathbf{a}}_R, L_R \rangle}, H) \\ &= e(g^{\langle \dot{a}_\ell \bar{\mathbf{a}}_L, L_R \rangle}, H) \\ &= e(g^{\langle \bar{\mathbf{a}}_L, L_R \rangle}, H^{\dot{a}_\ell}) \\ &= e(B_2, , H^{\dot{a}_\ell}) \end{aligned}$$

For each iteration of PoK Π_1 (where one iteration consists of steps 1-5, and step 6 follows by sending $\mathbf{x}', L'$ instead of providing a PoK), we have

$$\begin{aligned} \mathrm{COM}_{\bar{\mathbf{a}}'}(\mathbf{x}') &= g^{\langle \bar{\mathbf{a}}', \mathbf{x}' \rangle} \\ &= g^{\langle \bar{\mathbf{a}}_L, \mathbf{x}_L + c\mathbf{x}_R \rangle} \\ &= g^{\langle \bar{\mathbf{a}}_L, \mathbf{x}_L \rangle} g^{c\langle \bar{\mathbf{a}}_L, \mathbf{x}_R \rangle} \\ &= A_1 A_2^c \end{aligned} \qquad \begin{aligned} \mathrm{COM}_{\bar{\mathbf{a}}'}(L') &= g^{\langle \bar{\mathbf{a}}', L' \rangle} \\ &= g^{\langle \bar{\mathbf{a}}_L, cL_L + L_R \rangle} \\ &= g^{c\langle \bar{\mathbf{a}}_L, L_L \rangle} g^{\langle \bar{\mathbf{a}}_L, L_R \rangle} \\ &= B_1^c B_2 \end{aligned}$$

$$\begin{aligned}\langle L', \mathbf{x}' \rangle &= \langle cL_L + L_R, \mathbf{x}_L + c\mathbf{x}_R \rangle \\ &= c\langle L_L, \mathbf{x}_L \rangle + c^2 \langle L_L, \mathbf{x}_R \rangle + \langle L_R, \mathbf{x}_L \rangle + c\langle L_R, \mathbf{x}_R \rangle \\ &= \langle L_R, \mathbf{x}_L \rangle + c(\langle L_L, \mathbf{x}_L \rangle + \langle L_R, \mathbf{x}_R \rangle) + c^2 \langle L_L, \mathbf{x}_R \rangle \\ &= \langle L_R, \mathbf{x}_L \rangle + c\langle L, \mathbf{x} \rangle + c^2 \langle L_L, \mathbf{x}_R \rangle \\ &= y_1 + cy + c^2 y_2\end{aligned}$$

Special Soundness. We first illustrate the extraction of the witness, and thereafter proceed with the argument of correctness of the extracted value. We begin with 3 accepting transcripts for one iteration of PoK Π_1 (where one iteration consists of steps 1-5, and step 6 follows by sending $\mathbf{x}', L'$ instead of providing a PoK) as follows, where c_1, c_2, c_3 are all distinct challenges:

$$\begin{aligned}&(A_1, A_2, B_1, B_2, y_1, y_2, c_1, \mathbf{x}'_1, L'_1) \\ &(A_1, A_2, B_1, B_2, y_1, y_2, c_2, \mathbf{x}'_2, L'_2) \\ &(A_1, A_2, B_1, B_2, y_1, y_2, c_3, \mathbf{x}'_3, L'_3)\end{aligned}$$

Extraction. The extraction proceeds as follows. We aim to find $\mathbf{w}, \mathbf{m}$ such that $\langle \mathbf{m}, \mathbf{w} \rangle = y$, and $\mathbf{w}, \mathbf{m}$ are openings of P and Q. We note that, as c_1, c_2 and c_3 are such that $c_i \neq c_j$ for all $(i \neq j)$ $i, j \in \{1, 2, 3\}$, the Vandermonde matrix V described below is invertible.

$$V = \begin{pmatrix} 1 & 1 & 1 \\ c_1 & c_2 & c_3 \\ c_1^2 & c_2^2 & c_3^2 \end{pmatrix}$$

Hence, we can compute a_1, a_2, a_3 as $(a_1, a_2, a_3)^T = V^{-1}(0, 1, 0)^T$. The computed a_1, a_2, a_3 satisfy $\sum_i a_i = 0, \sum_i a_i c_i = 1$ and $\sum_i a_i c_i^2 = 0$. Define $\mathbf{z}_i = (c_i \mathbf{x}'_i \| \mathbf{x}'_i)$. Now let $\mathbf{w} = a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + a_3\mathbf{z}_3$ be the extracted value. Following a similar procedure, we extract $\mathbf{m}$ and output $(\mathbf{w}, \mathbf{m})$ as the witness for $\mathcal{R}_{\text{CLF}}$.

Proof of correctness of extracted value. We first prove that the extracted $\mathbf{w}$ and $\mathbf{m}$ are openings of commitments P and Q, respectively. Then we prove that $\mathbf{w}$ and $\mathbf{m}$ also satisfies the constraint $\langle \mathbf{m}, \mathbf{w} \rangle = y$. This shows that the extracted $(\mathbf{w}, \mathbf{m})$ is a valid witness for $(P, Q, y) \in \mathcal{R}_{\text{CLF}}$.

We recall that $\mathbf{w} = a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + a_3\mathbf{z}_3$ by definition, and $\text{COM}_{\overline{\mathbf{a}}'}(\mathbf{x}'_i) = A_1 A_2^{c_i}$ holds from verification equation in step 6.

$$\begin{aligned}\text{COM}_{\overline{\mathbf{a}}}(\mathbf{w}) &= g^{\langle \overline{\mathbf{a}}, \mathbf{w} \rangle} \\ &= g^{\langle \overline{\mathbf{a}}, a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + a_3\mathbf{z}_3 \rangle} \\ &= g^{\langle \overline{\mathbf{a}}_L \| \overline{\mathbf{a}}_R, a_1(c_1\mathbf{x}'_1 \| \mathbf{x}'_1) + a_2(c_2\mathbf{x}'_2 \| \mathbf{x}'_2) + a_3(c_3\mathbf{x}'_3 \| \mathbf{x}'_3) \rangle} \\ &= g^{\langle \overline{\mathbf{a}}_L, a_1c_1\mathbf{x}'_1 + a_2c_2\mathbf{x}'_2 + a_3c_3\mathbf{x}'_3 \rangle} g^{\langle \overline{\mathbf{a}}_R, a_1\mathbf{x}'_1 + a_2\mathbf{x}'_2 + a_3\mathbf{x}'_3 \rangle} \\ &= g^{\langle \overline{\mathbf{a}}', a_1c_1\mathbf{x}'_1 + a_2c_2\mathbf{x}'_2 + a_3c_3\mathbf{x}'_3 \rangle} g^{\langle \dot{a}_\ell \overline{\mathbf{a}}', a_1\mathbf{x}'_1 + a_2\mathbf{x}'_2 + a_3\mathbf{x}'_3 \rangle} \\ &= (A_1 A_2^{c_1})^{a_1c_1} (A_1 A_2^{c_2})^{a_2c_2} (A_1 A_2^{c_3})^{a_3c_3} (A_1 A_2^{c_1})^{a_1\dot{a}_\ell} (A_1 A_2^{c_2})^{a_2\dot{a}_\ell} (A_1 A_2^{c_3})^{a_3\dot{a}_\ell} \text{ (from step 6)} \\ &= A_1^{(a_1c_1 + a_2c_2 + a_3c_3) + \dot{a}_\ell(a_1 + a_2 + a_3)} A_2^{(a_1c_1^2 + a_2c_2^2 + a_3c_3^2) + \dot{a}_\ell(a_1c_1 + a_2c_2 + a_3c_3)} \\ &= A_1 A_2^{\dot{a}_\ell} \\ &= P \quad \text{(since we have } e\left(\frac{P}{A_1}, H\right) = e\left(A_2, H^{\dot{a}_\ell}\right) \text{ which ensures } P = A_1 A_2^{\dot{a}_\ell})\end{aligned}$$

Hence, the extracted $\mathbf{w}$ is an opening of the commitment P. Similarly we can prove that $\mathbf{m}$ is an opening of the commitment Q. From the binding of the commitment scheme, we can ensure that $\mathbf{w} = \mathbf{x}$ and $\mathbf{m} = L$ except with negligible probability.

Let $\mathbf{b} \in \mathbb{Z}_q^n$ be such that $\mathbf{x}'_i = \mathbf{b}_L + c_i\mathbf{b}_R$ holds for all $i = 1, 2, 3$. Then our defined $\mathbf{z}_i$ can be interpreted as $\mathbf{z}_i = (c_i\mathbf{x}'_i \| \mathbf{x}'_i) = (0 \| \mathbf{b}_L) + c_i(\mathbf{b}_L \| \mathbf{b}_R) + c_i^2(\mathbf{b}_R \| 0)$ for all $i = 1, 2, 3$. Now, given a_1, a_2, a_3 that satisfy

$\sum_i a_i = 0, \sum_i a_i c_i = 1$ and $\sum_i a_i c_i^2 = 0$, we have $\mathbf{w} = \sum_i a_i \mathbf{z}_i = \sum_i a_i (0 \| \mathbf{b}_L) + \sum_i a_i c_i (\mathbf{b}_L \| \mathbf{b}_R) + \sum_i a_i c_i^2 (\mathbf{b}_R \| 0) = \mathbf{b}$. Hence, the extracted $\mathbf{w}$ satisfies $\mathbf{x}'_i = \mathbf{w}_L + c_i \mathbf{w}_R$. Similarly, the extracted value $\mathbf{m}$ also satisfies $L'_i = \mathbf{m}_L + c_i \mathbf{m}_R$ for all $i = 1, 2, 3$.

Since the transcripts are accepting, step 7(b) of the verification equation, $\langle L'_i, \mathbf{x}'_i \rangle = y'_i$ holds; that is $\langle L'_i, \mathbf{x}'_i \rangle = y_1 + c_i y + c_i^2 y_2$, for all $i = 1, 2, 3$. Substituting the values of L'_i and $\mathbf{x}'_i$, we get that $\langle \mathbf{m}_R, \mathbf{w}_L \rangle + c_i \langle \mathbf{m}, \mathbf{w} \rangle + c_i^2 \langle \mathbf{m}_L, \mathbf{w}_R \rangle = y_1 + c_i y + c_i^2 y_2$ holds for all $i = 1, 2, 3$. Hence, we obtain that $\langle \mathbf{m}, \mathbf{w} \rangle = y$.

B.2 Proof of Theorem 3

Completeness follows directly.

Special Soundness. We consider 4 accepting transcripts for one iteration of PoK Π_2 with different challenges $t_i, i \in \{1, 2, 3\}$ as follows, where t_1, t_2, t_3 are all distinct challenges:

$$
\begin{aligned}
&(A_1, A_2, c, z_1, z_2, z_3, z_4, t_1, \mathbf{w}_1) \\
&(A_1, A_2, c, z_1, z_2, z_3, z_4, t_2, \mathbf{w}_2) \\
&(A_1, A_2, c, z_1, z_2, z_3, z_4, t_3, \mathbf{w}_3) \\
&(A_1, A_2, c, z_1, z_2, z_3, z_4, t_4, \mathbf{w}_4)
\end{aligned}
$$

We note that, as t_1, t_2, t_3 and t_4 are such that $t_i \neq t_j$ for all $(i \neq j)$ $i, j \in \{1, 2, 3, 4\}$, the matrix V described below is invertible.

$$
V = \begin{pmatrix}
1 & 1 & 1 & 1 \\
t_1 & t_2 & t_3 & t_4 \\
t_1^2 & t_2^2 & t_3^2 & t_4^2 \\
t_1^3 & t_2^3 & t_3^3 & t_4^3
\end{pmatrix}
$$

Let us denote $e_i, i \in \{1, 2, 3, 4\}$ where j^{th} entry of e_i is 1 for $j = i$, and 0 otherwise. Let us consider a vector $\boldsymbol{\rho}^j = (\rho_1^j, \rho_2^j, \rho_3^j, \rho_4^j)$ for $j = 1, 2, 3, 4$. Hence, we can compute $(\boldsymbol{\rho}^j)^T = V^{-1} e_j^T$. The computed $\rho_1^1, \rho_2^1, \rho_3^1, \rho_4^1$ satisfy $\sum_i \rho_i^1 = 1, \sum_i \rho_i^1 t_i = 0, \sum_i \rho_i^1 t_i^2 = 0$ and $\sum_i \rho_i^1 t_i^3 = 0$. Similarly, it holds for $j = 2, 3, 4$.

We define $\mathbf{r}_x$ to be the extracted value of $\mathbf{x}$ and compute it as $\mathbf{r}_x = \rho_1^1 \mathbf{w}_1 + \rho_2^1 \mathbf{w}_2 + \rho_3^1 \mathbf{w}_3 + \rho_4^1 \mathbf{w}_4$, given that $\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{w}_i) = P \cdot Q^{t_i} \cdot A_1^{t_i^2} \cdot A_2^{t_i^3}$ then we consider

$$
\begin{aligned}
\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{r}_x) &= g^{\langle \overline{\mathbf{a}}, \mathbf{r}_x \rangle} \\
&= g^{\langle \overline{\mathbf{a}}, \rho_1^1 \mathbf{w}_1 + \rho_2^1 \mathbf{w}_2 + \rho_3^1 \mathbf{w}_3 + \rho_4^1 \mathbf{w}_4 \rangle} \\
&= (PQ^{t_1} A_1^{t_1^2} A_2^{t_1^3})^{\rho_1} (PQ^{t_2} A_1^{t_2^2} A_2^{t_2^3})^{\rho_2} (PQ^{t_3} A_1^{t_3^2} A_2^{t_3^3})^{\rho_3} (PQ^{t_4} A_1^{t_4^2} A_2^{t_4^3})^{\rho_4} \\
&= P^{\sum_i \rho_i^1} Q^{\sum_i \rho_i^1 t_i} A_1^{\sum_i \rho_i^1 t_i^2} A_2^{\sum_i \rho_i^1 t_i^3} \\
&= P
\end{aligned}
$$

Hence, the extracted $\mathbf{r}_x$ is an opening of the commitment P.

Similarly, we define $\mathbf{r}_B, \mathbf{r}_{p_L}$ and $\mathbf{r}_{p_R}$ to be the extracted value of $\mathbf{B}, \boldsymbol{p}_L$ and $\boldsymbol{p}_R$,and compute them as $\mathbf{r}_B = \rho_1^2 \mathbf{w}_1 + \rho_2^2 \mathbf{w}_2 + \rho_3^2 \mathbf{w}_3 + \rho_4^2 \mathbf{w}_4$, $\mathbf{r}_{p_L} = \rho_1^3 \mathbf{w}_1 + \rho_2^3 \mathbf{w}_2 + \rho_3^3 \mathbf{w}_3 + \rho_4^3 \mathbf{w}_4$ and $\mathbf{r}_{p_R} = \rho_1^4 \mathbf{w}_1 + \rho_2^4 \mathbf{w}_2 + \rho_3^4 \mathbf{w}_3 + \rho_4^4 \mathbf{w}_4$. We also know that $\mathbf{w}(c) = z_1 + t_i \cdot z_2 + t_i^2 \cdot z_3 + t_i^3 \cdot z_4$ holds for all $i = 1, 2, 3, 4$, since the

transcripts are accepting transcripts.

$$\begin{aligned}
\mathbf{r}_x(c) &= \rho_1^1\mathbf{w}_1(c) + \rho_2^1\mathbf{w}_2(c) + \rho_3^1\mathbf{w}_3(c) + \rho_4^1\mathbf{w}_4(c) \\
&= \sum_i \rho_i^1 z_1 + \sum_i \rho_i^1 t_i \cdot z_2 + \sum_i \rho_i^1 t_i^2 \cdot z_3 + \sum_i \rho_i^1 t_i^3 \cdot z_4 \quad = z_1 \\
\mathbf{r}_B(c) &= \rho_1^2\mathbf{w}_1(c) + \rho_2^2\mathbf{w}_2(c) + \rho_3^2\mathbf{w}_3(c) + \rho_4^2\mathbf{w}_4(c) \\
&= \sum_i \rho_i^2 z_1 + \sum_i \rho_i^2 t_i \cdot z_2 + \sum_i \rho_i^2 t_i^2 \cdot z_3 + \sum_i \rho_i^2 t_i^3 \cdot z_4 \quad = z_2 \\
\mathbf{r}_{p_L}(c) &= \rho_1^3\mathbf{w}_1(c) + \rho_2^3\mathbf{w}_2(c) + \rho_3^3\mathbf{w}_3(c) + \rho_4^3\mathbf{w}_4(c) \\
&= \sum_i \rho_i^3 z_1 + \sum_i \rho_i^3 t_i \cdot z_2 + \sum_i \rho_i^3 t_i^2 \cdot z_3 + \sum_i \rho_i^3 t_i^3 \cdot z_4 \quad = z_3 \\
\mathbf{r}_{p_R}(c) &= \rho_1^4\mathbf{w}_1(c) + \rho_2^4\mathbf{w}_2(c) + \rho_3^4\mathbf{w}_3(c) + \rho_4^4\mathbf{w}_4(c) \\
&= \sum_i \rho_i^4 z_1 + \sum_i \rho_i^4 t_i \cdot z_2 + \sum_i \rho_i^4 t_i^2 \cdot z_3 + \sum_i \rho_i^4 t_i^3 \cdot z_4 \quad = z_4
\end{aligned}$$

Hence, we have that the extracted polynomials $\mathbf{r}_x, \mathbf{r}_B, \mathbf{r}_{p_L}$ and $\mathbf{r}_{p_R}$ satisfies the following constraint :

$$\begin{aligned}
& z_3 \cdot c^{-1} + y \cdot c^{n-1} + z_4 \cdot c^n = z_1 \cdot z_2 && \text{(from accepting transcripts)} \\
\implies & \mathbf{r}_{p_L}(c) \cdot c^{-1} + y \cdot c^{n-1} + \mathbf{r}_{p_R}(c) \cdot c^n = \mathbf{r}_x(c) \cdot \mathbf{r}_B(c)
\end{aligned}$$

Now, we consider $2n$ such transcripts with different verifier challenges $c_i, i \in \{1, \ldots, 2n\}$, each with 4 different challenges $t_{ij}, j \in \{1, \ldots, 4\}, i \in \{1, \ldots, 2n\}$. Hence, the above constraint is satisfied by $2n-1$ random challenges, i.e. the polynomials evaluations are consistent with the constraint at $2n$ evaluation points, where the highest degree of the polynomial is $2n-1$. Hence, polynomials identically satisfies the constraints at all points, which implies that $\langle \mathbf{r}_x, \mathsf{rev}(\mathbf{r}_B) \rangle = y$ holds for the aforementioned polynomials.

B.3 Proof of Theorem 4

Theorem 13. *Π_2' is a $(k_1, \ldots, k_\ell)$-move protocol for relation $(R, \mathbf{c^{n-1}}, z; \boldsymbol{w}) \in \mathcal{R}$, where $k_i = 3,\ \forall i \in [\ell]$. It is perfectly complete and computationally special sound.*

Completeness follows directly.

Special Soundness. We consider 3 accepting transcripts for one iteration of PoK Π_2' (where one iteration consists of steps 1-5, and step 6 follows by sending $\mathbf{x}', L'$ instead of providing a PoK) as follows, where s_1, s_2, s_3 are all distinct challenges. :

$$\begin{aligned}
&(A_1, A_2, z', s_1, \mathbf{w}_1') \\
&(A_1, A_2, z', s_2, \mathbf{w}_2') \\
&(A_1, A_2, z', s_3, \mathbf{w}_3')
\end{aligned}$$

Let us consider $\mathbf{z}_i = (s_i\mathbf{w}_i' \| \mathbf{w}_i')$. We note that, as s_1, s_2 and s_3 are such that $s_i \neq s_j$ for all $(i \neq j)$ $i, j \in \{1, 2, 3\}$, the matrix V described below is invertible.

$$V = \begin{pmatrix} 1 & 1 & 1 \\ s_1 & s_2 & s_3 \\ s_1^2 & s_2^2 & s_3^2 \end{pmatrix}$$

Hence, we can compute $(a_1, a_2, a_3)^T = V^{-1}(0, 1, 0)^T$. The computed a_1, a_2, a_3 satisfy $\sum_i a_i = 0, \sum_i a_i c_i = 1$ and $\sum_i a_i c_i^2 = 0$.

Let us consider $\mathbf{x} = a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + a_3\mathbf{z}_3$, given that $\mathrm{COM}_{\overline{\mathbf{a}}'}(\mathbf{w}'_i) = A_1 A_2^{s_i}$ then we consider

$$\begin{aligned}
\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{x}) &= g^{\langle \overline{\mathbf{a}}, \mathbf{x}\rangle} \\
&= g^{\langle \overline{\mathbf{a}}, a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + a_3\mathbf{z}_3\rangle} \\
&= g^{\langle \overline{\mathbf{a}}_L \| \overline{\mathbf{a}}_R, a_1(s_1\mathbf{w}'_1 \| \mathbf{w}'_1) + a_2(s_2\mathbf{w}'_2 \| \mathbf{w}'_2) + a_3(s_3\mathbf{w}'_3 \| \mathbf{w}'_3)\rangle} \\
&= g^{\langle \overline{\mathbf{a}}_L, a_1 s_1\mathbf{w}'_1 + a_2 s_2\mathbf{w}'_2 + a_s s_3\mathbf{w}'_3\rangle} g^{\langle \overline{\mathbf{a}}_R, a_1\mathbf{w}'_1 + a_2\mathbf{w}'_2 + a_3\mathbf{w}'_3\rangle} \\
&= g^{\langle \overline{\mathbf{a}}', a_1 s_1\mathbf{w}'_1 + a_2 s_2\mathbf{w}'_2 + a_3 s_3\mathbf{w}'_3\rangle} g^{\langle \dot{a}_\ell \overline{\mathbf{a}}', a_1\mathbf{w}'_1 + a_2\mathbf{w}'_2 + a_3\mathbf{w}'_3\rangle} \\
&= (A_1 A_2^{s_1})^{a_1 s_1} (A_1 A_2^{s_2})^{a_2 s_2} (A_1 A_2^{s_3})^{a_3 s_3} (A_1 A_2^{s_1})^{a_1 \dot{a}_\ell} (A_1 A_2^{s_2})^{a_2 \dot{a}_\ell} (A_1 A_2^{s_3})^{a_3 \dot{a}_\ell} \\
&= A_1^{(a_1 s_1 + a_2 s_2 + a_3 s_3) + \dot{a}_\ell(a_1 + a_2 + a_3)} A_2^{(a_1 s_1^2 + a_2 s_2^2 + a_3 s_3^2) + \dot{a}_\ell(a_1 s_1 + a_2 s_2 + a_3 s_3)} \\
&= A_1 A_2^{\dot{a}_\ell} \\
&= R \quad (\text{since we have } e\left(\frac{R}{A_1}, H\right) = e\left(A_2, H^{\dot{a}_\ell}\right) \text{ which ensures } R = A_1 A_2^{\dot{a}_\ell})
\end{aligned}$$

Hence, the extracted $\mathbf{x}$ is an opening of the commitment R. From the binding of the commitment scheme, we can ensure that $\mathbf{x} = \mathbf{w}$ except with negligible probability.

From the accepting transcripts, we have that $\langle (L'_c)_i, \mathbf{x}'_i\rangle = y_1 + s_i y + s_i^2 y_2$ for $i = 1, 2, 3$. Now, we consider the following :

$$\begin{aligned}
&\langle (L'_c)_i, \mathbf{w}'_i\rangle = \langle s_i(L_c)_L + (L_c)_R, \mathbf{w}_L + s_i\mathbf{w}_R\rangle && \forall i \in \{1,2,3\} \\
\implies\ & y_1 + s_i y + s_i^2 y_2 = \langle (L_c)_R, \mathbf{w}_L\rangle + s_i\langle (L_c), \mathbf{w}\rangle + s_i^2\langle (L_c)_L, \mathbf{w}_R\rangle && \forall i \in \{1,2,3\} \\
\implies\ & y = \langle L_c, \mathbf{w}\rangle
\end{aligned}$$

C Proofs from Section 4

C.1 Proof of Theorem 7

The proof of completeness is straightforward to argue.

Special Soundness. Let $\mathbf{y}, \mathbf{y}'$ be defined as $\mathbf{y} = \mathbf{u} + r\mathbf{a} + r^2\mathbf{b}$, $\mathbf{y}' = \mathbf{u}' + r\mathbf{c}'$, $q = v_1 + rw_1 + r^2 w_2$ and $q' = v_2 + rw_1 w_2$. Now we note that the we invoke $(\Pi_2)_c$ for $(UA^r B^{r^2}, P_n, v_1 + rw_1 + r^2 w_2; \mathbf{u} + r\mathbf{a} + r^2\mathbf{b}, \rho_{\mathsf{n}})$ and $(U'C^r, P_{2n}, v_2 + rw_1 w_2; \mathbf{u}' + r\mathbf{c}', \rho_{2\mathsf{n}}) \in \mathcal{R}_{\text{CLF-rev}}$. Our extractor invokes the extractor for $(\Pi_2)_c$ to extract $\mathbf{y}, \mathbf{y}', \rho_{\mathsf{n}}$ and $\rho_{2\mathsf{n}}$ such that $\langle \rho_{\mathsf{n}}, \mathbf{y}\rangle = q, \langle \rho_{2\mathsf{n}}, \mathbf{y}'\rangle = q'$. Extracting $\mathbf{y}_1, \mathbf{y}'_1, \mathbf{y}_2, \mathbf{y}'_2, \mathbf{y}_3, \mathbf{y}'_3$ for three distinct challenges e_1, e_2, e_3, our extractor additionally computes $\mathbf{u}, \mathbf{u}', \mathbf{a}, \mathbf{b}, \mathbf{c}'$ such that $\langle \rho_{\mathsf{n}}, \mathbf{u}\rangle = v_1, \langle \rho_{2\mathsf{n}}, \mathbf{u}'\rangle = v_2, \langle \rho_{\mathsf{n}}, \mathbf{a}\rangle = w_1, \langle \rho_{\mathsf{n}}, \mathbf{b}\rangle = w_2, \langle \rho_{2\mathsf{n}}, \mathbf{c}'\rangle = w_1 w_2$. Note that the binding of the commitment ensures that the correct $\mathbf{u}, \mathbf{u}', \mathbf{a}, \mathbf{b}, \mathbf{c}', \rho_{\mathsf{n}}$ and $\rho_{2\mathsf{n}}$ have been extracted.

Our extractor again invokes the knowledge extractor of $\Pi_{\mathsf{com\text{-}mult}}$ to extract $\rho_{\mathsf{n}}, \rho_{2\mathsf{n}}$ such that $\rho_{\mathsf{n}} = V^{-1}(1\ z\ z^2\ \cdots\ z^n)^T$ and $\rho_{2\mathsf{n}} = V^{-1}(1\ z\ z^2\ \cdots\ z^{2n+1})^T$, and the binding of the commitment ensures the consistency of the extracted openings. Hence, it follows that the extracted witnesses satisfies $p_{\boldsymbol{u}}(z) = v_1, p_{\boldsymbol{u}'}(z) = v_2, p_{\boldsymbol{a}}(z) = w_1, p_{\boldsymbol{b}}(z) = w_2, p_{\boldsymbol{c}}(z) = w_1 w_2$.

We can extract $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}'_i$ for $i \in [2n+2]$ distinct challenges z_i such that $p_{\mathbf{c}'_i}(z_i) = p_{\mathbf{a}_i}(z_i) p_{\mathbf{b}_i}(z_i)$. If any of the extracted $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}'_i$ differ, we will have broken binding and so we have that except with negligible probability, all the extracted $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}'_i$ are identical. Thus, we have extracted $\mathbf{a}, \mathbf{b}, \mathbf{c}'$ that satisfy $p_{\mathbf{c}'}(z) = p_{\mathbf{a}}(z) p_{\mathbf{b}}(z)$ for $2n+2$ distinct z. This allows us to conclude (from the Schwartz-Zippel Lemma) that $p_{\mathbf{c}'}(X) = p_{\mathbf{a}}(X) p_{\mathbf{b}}(X)$.

Zero-Knowledge. Given access to verifier's randomness z, r, the simulator $\mathcal{S}_{\mathsf{had}}$ proceeds as follows:

1. $\mathcal{S}_{\mathsf{had}}$ samples $w_1, w_2 \leftarrow_R \mathbb{Z}_q$, $\mathbf{y} \leftarrow_R \mathbb{Z}_q^{n+1}, \mathbf{y}' \leftarrow_R \mathbb{Z}_q^{2n+1}$, and sends $w_1, w_2, U = \frac{\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{y})}{A^r B^{r^2}}, U' = \frac{\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{y}')}{C^r}, v_1 = \langle \rho_{\mathsf{n}}, \mathbf{y}\rangle - rw_1 - r^2 w_2, v_2 = \langle \rho_{2\mathsf{n}}, \mathbf{y}'\rangle - rw_1 w_2$ to $\mathcal{V}$.
2. $\mathcal{S}_{\mathsf{had}}$ sets $Y = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{y}), Y' = \mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{y}'), q = \langle \rho_{\mathsf{n}}, \mathbf{y}\rangle, q' = \langle \rho_{2\mathsf{n}}, \mathbf{y}'\rangle$.

3. $\mathcal{S}_{\mathsf{had}}$ then honestly executes $(\Pi_2)_c$ to show that $(Y, P_n, q; \mathbf{y}, \rho_{\mathsf{n}})$, $(Y', P_{2n}, q'; \mathbf{y}', \rho_{2\mathsf{n}}) \in \mathcal{R}_{\text{CLF-rev}}$ in Step 9.

We now argue that the distribution of the simulated transcript is indistinguishable from the transcript obtain from real protocol execution. Since the underlying vector $\mathbf{y}, \mathbf{y}', w_1$ and w_2 are sampled uniformly at random, the computed U, U', v_1, v_2 subject to the constraints $\text{COM}_{\overline{\mathbf{a}}}(\mathbf{y}) = UA^r B^{r^2}$, $\text{COM}_{\overline{\mathbf{a}}}(\mathbf{y}') = U'C^r$, $q = v_1 + rw_1 + r^2 w_2$, and $q' = v_2 + rw_1 w_2$ outlined in Step 9, where $r \leftarrow_R \mathbb{Z}_q$, are distributed uniformly at random in the transcript. The remaining computations are performed honestly and are thus indistinguishable from an actual protocol execution.

C.2 Proof of Theorem 8

The proof of completeness is straightforward to argue.

Special Soundness. We rely on the sub-routine $\Pi_{2\text{-}\mathcal{R}}$ invoked in step 10 to obtain openings of A, B, C and D provided appropriate $(2, 2n, 4, 3, \ldots, 3)$ tree of accepting transcripts. We denote $\mathbf{f}_a$, $\mathbf{f}_b, \mathbf{f}_c$ and $\mathbf{f}_d$ to denote the openings of A, B, C and D respectively which satisfies the following constraints, $\langle \rho_{\mathsf{n}}, \mathbf{f}_a \rangle \cdot \langle \rho_{\mathsf{n}}, \mathbf{f}_c \rangle = \langle \rho'_{2\mathsf{n}}, \mathbf{f}_c \rangle$ and $\langle \delta_{\mathsf{n}}, \mathbf{f}_b \rangle \cdot \langle \delta_{\mathsf{n}}, \mathbf{f}_d \rangle = \langle \delta'_{2\mathsf{n}}, \mathbf{f}_d \rangle$.

The binding of the commitment scheme ensures that $\mathbf{f}_a = \mathbf{a}$, $\mathbf{f}_b = \mathbf{b}, \mathbf{f}_c = \mathbf{c}''$ and $\mathbf{f}_d = \mathbf{d}''$. The constraints of the linear forms from $(A, P_n, z_1; \mathbf{a}, \rho_{\mathsf{n}})$,$(C, P_n, z_2; \mathbf{c}'', \rho_{\mathsf{n}})$, $(B, Q_n, w_1; \mathbf{b}, \delta_{\mathsf{n}})$, $(D, Q_n, w_2; \mathbf{d}'', \delta_{\mathsf{n}})$, $(C, P_{2n}, z_1 z_2; \mathbf{c}'', \rho'_{2\mathsf{n}})$, $(D, Q_{2n}, w_1 w_2; \mathbf{d}'', \delta'_{2\mathsf{n}}) \in \mathcal{R}_{\text{CLF}}$ and the verifier check in step 9 further ensure that the polynomials satisfy $p_{\boldsymbol{c}} = p_{\boldsymbol{a}} \cdot p_{\boldsymbol{e}}$ at a random point, i.e. the polynomials are identical with high probability via Schwartz-Zippel Lemma. Hence, the evaluations at each point satisfy the relation with high probability, which gives us $c_i = a_i \cdot e_i = a_i \cdot c_{i-1}$ (considering $c_0 = 1$). We get $\mathbf{a}(z) \circ \mathbf{c}_{\mathsf{left}}(z) = \mathbf{c}_{\mathsf{right}}(z)$ and $\mathbf{b}(w) \circ \mathbf{d}_{\mathsf{left}}(w) = \mathbf{d}_{\mathsf{right}}(w)$ where $\mathbf{c}_{\mathsf{left}} = (1, c, \ldots, c^{n-1}) = \mathbf{e}$, $\mathbf{c}_{\mathsf{right}} = (c, \ldots, c^n) = \mathbf{c}$, $\mathbf{d}_{\mathsf{left}} = (1, d, \ldots, d^{n-1})$, and $\mathbf{d}_{\mathsf{right}} = (d, \ldots, d^n)$, that is $a_i \cdot c''_{i-1} = c''_i$ and $b_i \cdot d''_{i-1} = d''_i$ holds for all $i \in [n]$. Additionally, the constraints of the linear forms from $(C, \mathsf{left}, 1; \mathbf{c}'', (1 \| \mathbf{0} \| \mathbf{0}))$, $(C, \mathsf{right}, x; \mathbf{c}'', (\mathbf{0} \| 1 \| \mathbf{0}))$, $(D, \mathsf{left}, 1; \mathbf{d}'', (1 \| \mathbf{0} \| \mathbf{0}))$, $(D, \mathsf{right}, x; \mathbf{d}'', (\mathbf{0} \| 1 \| \mathbf{0})) \in \mathcal{R}_{\text{CLF}}$ ensures that the $c''_1 = 1$ and $d''_1 = 1$. The constraints of the linear forms from $(C, \mathsf{right}, x; \mathbf{c}'')$, $(D, \mathsf{right}, x; \mathbf{d}'') \in \mathcal{R}$ ensures that the $c''_n = d''_n = x$ which provides us $\prod_{i=1}^n a_i = x = \prod_{i=1}^n b_i$.

We define $\mathbf{r}, \mathbf{s} \in \mathbb{Z}_q^n$ as $r_i = a_i - i\beta - \gamma$, $s_i = b_i - \sigma(i)\beta - \gamma$ for all $i \in [n]$, for the public permutation σ. We rely on the [BG12] to ensure that given $\prod_{i=1}^n a_i = x = \prod_{i=1}^n b_i$ holds which implies $\prod_{i=1}^n (r_i + i\beta + \gamma) = \prod_{i=1}^n (s_i + \sigma(i)\beta + \gamma)$ holds, we can ensure that the computed (extracted) vectors $\mathbf{r}, \mathbf{s}$ are such that $\sigma(\mathbf{r}) = \mathbf{s}$. The argument follows provided we have $O(n)$ accepting transcripts of Π_{perm}.

C.3 Proof of Theorem 9

Completeness follows directly.

Special soundness. We invoke the extractor of $(\Pi_2)_c$ to extract witnesses for $2m + 2$ distinct ts in Step 10(a). Given these witnesses, we can either conclude that Σ_0 is a commitment to $\mathbf{c}^{\mathbf{m}} \| \mathbf{0}$ or break binding of the commitment scheme. Further, we can invoke the extractor of Π_{perm} to extract $\forall k \in \{L, R\}$ $\forall i \in [M]$ vectors $\mathbf{u}_i^k$ such that $\mathbf{u}_i^k = \sigma_i^k(\mathbf{c}^{\mathbf{m}})$. We invoke the extractor of Π_{had} to extract $\forall k \in \{L, R\}$ $\forall i \in [M]$ vectors $\mathbf{t}_i^k = \sigma_i^k(\mathbf{c}^{\mathbf{m}}) \circ \mathbf{w}_i^k \| h_{(\sigma_i^k(\mathbf{c}^{\mathbf{m}}) \circ \mathbf{w}_i^k)}$. In steps 10(d) and 10(e), given witnesses for $2m + 2$ distinct challenges, we either extract vectors L'_1, L''_1 such that $L'_1 = \sum_{i=1}^M \mathbf{t}_i^L + u\mathbf{t}_i^R$ and $L''_1 = (\mathbf{0} \| ((L'_1)_{m-i+1})_{i \in [M]}) = \mathsf{rev}(L_1)$ or break binding.

In steps 10(f) and 10(g), given witnesses for distinct z_1, z_2 that satisfy the given constraints, we can extract vectors $\mathbf{y}, \tilde{\mathbf{x}}^O, \mathbf{r}, \tilde{\mathbf{r}}$ such that $\forall i \in [2]$, $\langle L_1, z_i \tilde{\mathbf{x}}^O + \mathbf{r} \rangle = \langle \mathbf{c}^{\mathbf{m}}, z_i \mathbf{y} + \tilde{\mathbf{r}} \rangle - z_i K$. This implies that $\langle L_1, \tilde{\mathbf{x}}^O \rangle = \langle \mathbf{c}^{\mathbf{m}}, \mathbf{y} \rangle - K$. Given accepting $\mathbf{y}$ for distinct challenges u_1, u_2, we can extract $W^L(c), W^R(c), \tilde{\mathbf{x}}^L, \tilde{\mathbf{x}}^R$ such that $\langle W^L(c), \tilde{\mathbf{x}}^O \rangle = \langle \mathbf{c}^{\mathbf{m}}, \tilde{\mathbf{x}}^L \rangle - K$ and $\langle W^R(c), \tilde{\mathbf{x}}^O \rangle = \langle \mathbf{c}^{\mathbf{m}}, \tilde{\mathbf{x}}^R \rangle$. If the $\tilde{\mathbf{x}}^L, \tilde{\mathbf{x}}^R, \tilde{\mathbf{x}}^O$ obtained in this way are different from the ones extracted by the extractor of Π_{had} in Step (h), we will have broken binding of the commitment scheme.

Given $\tilde{\mathbf{x}}^L, \tilde{\mathbf{x}}^R, \tilde{\mathbf{x}}^O$ for $2m+1$ distinct c, we either break binding or conclude that $\tilde{\mathbf{x}}^L, \tilde{\mathbf{x}}^R, \tilde{\mathbf{x}}^O$ satisfy the linear constraints of the circuit as well as the multiplicative constraints, and so $\mathbf{x}^O = (\tilde{\mathbf{x}}^O)_{i \in [m]}$ must be a satisfying assignment.

Special HVZK. We describe a simulator that given commitments to a satisfying assignment and the randomness of the verifier, computes a transcript which is perfectly indistinguishable from the transcript of a real execution. The simulator $\mathcal{S}$ acts as follows:

- It computes all commitments honestly in Step 2.
- In Step 4, it samples $\mathbf{r}, \tilde{\mathbf{r}} \leftarrow_R \mathbb{Z}_q^m, s, \tilde{s} \leftarrow_R \mathbb{Z}_q$ such that $\langle W^L(c) + uW^R(c), \mathbf{r}\rangle = \langle \mathbf{c^m}, \tilde{\mathbf{r}}\rangle - K$. It sets $R = \frac{\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{r}, s)}{(X^O)^z}$, $\tilde{R} = \frac{\mathrm{COM}_{\overline{\mathbf{a}}}(\tilde{\mathbf{r}}, \tilde{s})}{(X^L)^z (X^R)^{zu}}$.
- In Step 6, it sets $v_1 = \langle L_1, \mathbf{r}\rangle$.
- It honestly executes Step 8.
- It honestly executes Steps 10(a)-10(e), and invokes the simulator $\mathcal{S}$, $\mathcal{S}_{\mathsf{had}}$ of Π_0 with the corresponding verifier randomness in Steps 10(f)-(h).

We analyze the distribution of the transcript. In both executions, the elements $R, \tilde{R}, v_1$ are distributed uniformly at random. The indistinguishability of steps 10(f)-(h) follows from the simulators of Π_0 and $\Pi_{hadamard}$. All other computations are honestly executed.

D Proofs from Section 5

D.1 Proof of Theorem 11

Proof. **Special Soundness.** We consider 3 accepting transcripts for one iteration of PoK $\Pi_{\text{1-hom}}$ (where one iteration consists of steps 1-5, and step 6 follows by sending $\mathbf{x}', L'$ instead of providing a PoK) as follows, where c_1, c_2, c_3 are all distinct challenges. :

$$(A_1, A_2, B_1, B_2, y_1, y_2, c_1, g_1', \mathbf{x}_1', f_1')$$
$$(A_1, A_2, B_1, B_2, y_1, y_2, c_2, g_2', \mathbf{x}_2', f_2')$$
$$(A_1, A_2, B_1, B_2, y_1, y_2, c_3, g_3', \mathbf{x}_3', f_3')$$

Let us consider $\mathbf{z}_i = (c_i \mathbf{x}_i' \| \mathbf{x}_i')$. We note that, as c_1, c_2 and c_3 are such that $c_i \neq c_j$ for all $(i \neq j)$ $i, j \in \{1, 2, 3\}$, the matrix V described below is invertible.

$$V = \begin{pmatrix} 1 & 1 & 1 \\ c_1 & c_2 & c_3 \\ c_1^2 & c_2^2 & c_3^2 \end{pmatrix}$$

Hence, we can compute $(a_1, a_2, a_3)^T = V^{-1}(0, 1, 0)^T$. The computed a_1, a_2, a_3 satisfy $\sum_i a_i = 0, \sum_i a_i c_i = 1$ and $\sum_i a_i c_i^2 = 0$.

We define $\mathbf{w}$ to be the extracted value of $\mathbf{x}$ and compute it as $\mathbf{w} = a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + a_3\mathbf{z}_3$, given that $\mathrm{COM}_{\overline{\mathbf{a}}'}(\mathbf{x}_i') = A_1 + c_i P + c_i^2 A_2$ then we consider

$$\begin{aligned}
\mathrm{COM}_{\overline{\mathbf{a}}}(\mathbf{w}) &= (\langle \overline{\mathbf{a}}, \mathbf{w}\rangle) g \\
&= (\langle \overline{\mathbf{a}}, a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + a_3\mathbf{z}_3\rangle) g \\
&= (\langle \overline{\mathbf{a}}_L \| \overline{\mathbf{a}}_R, a_1(c_1\mathbf{x}_1' \| \mathbf{x}_1') + a_2(c_2\mathbf{x}_2' \| \mathbf{x}_2') + a_3(c_3\mathbf{x}_3' \| \mathbf{x}_3')\rangle) g \\
&= (\langle \overline{\mathbf{a}}_L, a_1c_1\mathbf{x}_1' + a_2c_2\mathbf{x}_2' + a_3c_3\mathbf{x}_3'\rangle) g + (\langle \overline{\mathbf{a}}_R, a_1\mathbf{x}_1' + a_2\mathbf{x}_2' + a_3\mathbf{x}_3'\rangle) g \\
&= (\langle \overline{\mathbf{a}}', a_1c_1\mathbf{x}_1' + a_2c_2\mathbf{x}_2' + a_3c_3\mathbf{x}_3'\rangle) g (\langle \dot{a}_\ell \overline{\mathbf{a}}', a_1\mathbf{x}_1' + a_2\mathbf{x}_2' + a_3\mathbf{x}_3'\rangle) g \\
&= (a_1\langle (c_1 + \dot{a}_\ell)\overline{\mathbf{a}}', \mathbf{x}_1'\rangle + a_2\langle (c_2 + \dot{a}_\ell)\overline{\mathbf{a}}', \mathbf{x}_2'\rangle + a_3\langle (c_3 + \dot{a}_\ell)\overline{\mathbf{a}}', \mathbf{x}_3'\rangle) g \\
&= (a_1\langle \overline{\mathbf{a}}', \mathbf{x}_1'\rangle)(g_1') + (a_2\langle \overline{\mathbf{a}}', \mathbf{x}_2'\rangle)(g_2') + (a_3\langle \overline{\mathbf{a}}', \mathbf{x}_3'\rangle)(g_3') \qquad \text{(from Step 4)} \\
&= a_1\left(A_1 + c_1P + c_1^2A_2\right) + a_2\left(A_1 + c_2P + c_2^2A_2\right) + a_3\left(A_1 + c_3P + c_3^2A_2\right) \text{(from last check)} \\
&= (a_1 + a_2 + a_3)A_1 + (a_1c_1 + a_2c_2 + a_3c_3)P + (a_1c_1^2 + a_2c_2^2 + a_3c_3^2)A_2 \quad = P
\end{aligned}$$

Hence, the extracted $\mathbf{w}$ is an opening of the commitment P. Similarly we can extract an opening $\mathbf{m}$ of the commitment Q. From the binding of the commitment scheme, we have that $\mathbf{w} = \mathbf{x}$ and $\mathbf{m} = \mathsf{rev}(f)$ except with negligible probability.

From the accepting transcripts, we have that $f'_i(\mathbf{x}'_i) = y_1 + c_i y + c_i^2 y_2$ for $i = 1, 2, 3$. Now, we consider the following :

$$\begin{aligned} & f'_i(\mathbf{x}'_i) = (c_i f_L + f_R)(\mathbf{x}_L + c_i \mathbf{x}_R) && \forall i \in \{1,2,3\} \\ \implies & y_1 + c_i y + c_i^2 y_2 = f_R(\mathbf{x}_L) + c_i f(\mathbf{x}) + c_i^2 f_L(\mathbf{x}_R) && \forall i \in \{1,2,3\} \\ \implies & y = f(\mathbf{x}) \end{aligned}$$

D.2 Proof of Theorem 12

Special Soundness. We consider 3 accepting transcripts for one iteration of PoK $\Pi_{\mathsf{1\text{-}gen\text{-}hom}}$ (where one iteration consists of steps 1-9, and step 10 follows by sending $\mathbf{x}', L'$ instead of providing a PoK) as follows, where c_1, c_2, c_3 are all distinct challenges. :

$$\begin{aligned} & (A_1, A_2, B_1, B_2, a_1, a_2, b_1, b_2, d_1, d_2, c_1, (\mathsf{ck}'^{(1)}_{i,1})_{i=0,\ldots,k-1}, \mathbf{x}'_1, f'_1) \\ & (A_1, A_2, B_1, B_2, a_1, a_2, b_1, b_2, d_1, d_2, c_2, (\mathsf{ck}'^{(2)}_{i,1})_{i=0,\ldots,k-1}, \mathbf{x}'_2, f'_2) \\ & (A_1, A_2, B_1, B_2, a_1, a_2, b_1, b_2, d_1, d_2, c_3, (\mathsf{ck}'^{(3)}_{i,1})_{i=0,\ldots,k-1}, \mathbf{x}'_3, f'_3) \end{aligned}$$

Let us consider $\mathbf{z}_i = (c_i \mathbf{x}'_{i,S} \| \mathbf{x}'_{i,S})$. We note that, as c_1, c_2 and c_3 are such that $c_i \neq c_j$ for all $(i \neq j)$ $i, j \in \{1, 2, 3\}$, the matrix V described below is invertible.

$$V = \begin{pmatrix} 1 & 1 & 1 \\ c_1 & c_2 & c_3 \\ c_1^2 & c_2^2 & c_3^2 \end{pmatrix}$$

Hence, we can compute $(a_1, a_2, a_3)^T = V^{-1}(0, 1, 0)^T$. The computed a_1, a_2, a_3 satisfy $\sum_i a_i = 0, \sum_i a_i c_i = 1$ and $\sum_i a_i c_i^2 = 0$.

We define $\mathbf{w}$ to be the extracted value of $\mathbf{x}$ and compute it as $\mathbf{w} = a_1 \mathbf{z}_1 + a_2 \mathbf{z}_2 + a_3 \mathbf{z}_3$, given that $\mathrm{COM}'_1(\mathbf{x}'_{i,S}) = A_1 + c_i P + c^2 A_2$ then we note that $\mathrm{COM}_1(\mathbf{x}_{i,S}) = P$ analogous to the proof of $\Pi_{\mathsf{1\text{-}hom}}$. Hence, the extracted $\mathbf{w}$ is an opening of the commitment P. Similarly we can extract an opening $\mathbf{m}$ of the commitment Q. From the binding of the commitment scheme, we have that $\mathbf{w} = \mathbf{x}_S$ and $\mathbf{m} = \mathsf{rev}(f_S)$ except with negligible probability.

From the accepting transcripts, we have that $f'_{i,S}(\mathbf{x}'_{i,S}) = a_1 + c_i y_1 + c^2 a_2$, $f_T(\mathbf{x}_T) = y_4$, $(f'_{i,S}, c_i f'_{i,S})(\mathbf{x}_T) = b_1 + c y_2 + c^2 b_2$, and $f_T(c\mathbf{x}'_{i,S}, \mathbf{x}'_{i,S}) = d_1 + c y_3 + c^2 d_2$ for $i = 1, 2, 3$. Now, we consider the following :

$$\begin{aligned} & f'_{i,S}(\mathbf{x}'_{i,S}) = (c_i f_{S,L} + f_{S,R})(\mathbf{x}_{S,L} + c_i \mathbf{x}_{S,R}) && \forall i \in \{1,2,3\} \\ \implies & a_1 + c_i y_1 + c^2 a_2 = f_{S,R}(\mathbf{x}_{S,L}) + c_i f_S(\mathbf{x}_S) + c_i^2 f_{S,L}(\mathbf{x}_{S,R}) && \forall i \in \{1,2,3\} \\ \implies & y_1 = f_S(\mathbf{x}_S) \end{aligned}$$

Similarly, we can ensure that $y_2 = f_S(\mathbf{x}_T), y_3 = f_T(\mathbf{x}_S), y_4 = f_T(\mathbf{x}_T)$.