FLIP-and-Prove R1CS

Anca Nitulescu[1], Nikitas Paslis[2], and Carla Ràfols[2]

[1] Input Output anca.nitulescu@iohk.io
[2] Universitat Pompeu Fabra {nikitas.paslis,carla.rafols}@upf.edu

Abstract. In this work, we consider the setting where one or more users with low computational resources would like to outsource the task of proof generation for SNARKs to one external entity, named Prover. We study the scenario in which Provers have access to all statements and witnesses to be proven beforehand. We take a different approach to proof aggregation and design a new protocol that reduces simultaneously proving time and communication complexity, without going through recursive proof composition. Our two main contributions: We first design FLIP, a communication efficient folding scheme where we apply the Inner Pairing Product Argument to fold R1CS instances of the same language into a single relaxed R1CS instance. Then, any proof system for relaxed R1CS language can be applied to prove the final instance. As a second contribution, we build a novel variation of Groth16 with the same communication complexity for relaxed R1CS and two extra pairings for verification, with an adapted trusted setup.
Compared to SnarkPack – a prior solution addressing scaling for multiple Groth16 proofs – our scheme improves on prover complexity by orders of magnitude, if we consider the total cost to generate the SNARK proofs one by one and the aggregation effort.
An immediate application of our solution is Filecoin, a decentralized storage network based on incentives that generates more than 6 million SNARKs for large circuits of 100 million constraints per day.

1 Introduction

Succinct Arguments of Knowledge or SNARKs are short proofs that allow to publicly verify the correctness of a statement (e.g., the result of some computation, claims of storage etc.) at lower cost than examining the entire witness or rerunning the entire computation. SNARKs are widely used in decentralized settings such as blockchains to scale or conceal information published on the chain. Some of the multiple use cases are for anonymous transactions [HBHW21] (Zcash), fast light clients or compact blockchain (Celo, Mina), and provable decentralized storage [Lab18] (Filecoin).

Nevertheless, due to their rapid and massive adoption, systems that use SNARKs are facing scalability challenges: proofs are expensive to generate, many of these are produced constantly. So in the decentralized scenario, all the nodes in the network have to process each proof individually to agree on a final state, so some verification fees are required for each proof posted.

To avoid overcharging public bulletin boards with such proofs, many blockchains have started considering aggregation layers which are practically "Proving Service" nodes that collect multiple proofs or compute and generate directly instances to be proven, while running offchain. Users in the network can use such Proving Services to prove many instances and hopefully obtain a single smaller proof efficiently verifiable to post on the main chain. Recently, many schemes were developed and offer different features to solve this problem through aggregation, recursion or folding. However, most of the existing solutions alleviate the communication and verification costs, while sacrificing the prover efficiency.

Aggregation schemes, such as SnarkPack [GMN22a] deployed in Filecoin blockchain allow to compute a unique short proof for many statements, starting from already existing individual proofs that are aggregated via Inner Product Arguments [BMM+21] into a logarithmic size final proof that has verification costs logarithmic in the number of initial statements.

Another approach to handle multiple instances is by recursion, or incremental verifiable computation (IVC): this allows to compute sequentially proofs for each of the individual instances and continue the process until a unique final proof is obtained. Nevertheless, even by instantiating such IVC by using efficient generic SNARK schemes, we obtain very unsatisfactory solutions. This is because such construction would require recursion on the verifier algorithm, but the verification of SNARKs are usually costly and include operations such as elliptic curve pairings that are not easy to represent in an arithmetic circuit over a field. So implementing the verification logic in the SNARK proving circuit incurs a significant overhead.

Some schemes, starting with Halo work [BGH19] show how to save on costs by using aggregation schemes to batch the expensive operations in the proof verification and run them only once at the end. This reduces both the recursive verification costs for the prover and the verifier's effort. However, such approaches require non-standard structures such as cycle of elliptic curves. This restricts the choice of tools that can be used by the recursive scheme, since known impossibility results show there are no optimal (in terms of parameter sizes) pairing-friendly of secure elliptic curves, the one allowing for the most efficient constructions. However, hybrid cycles and plain cycles of curves have been proposed and better studied in the last years, but the absence of optimized implementations of recursion using this accumulation approach is still a problem to realizing practical recursive SNARK schemes.

A recent line of work demonstrates a new methodology in order to defer the expensive operations in the proof verification, via folding schemes. Nova [KST22], HyperNova [KS23], Protostar [BC23], etc. fold together at each step the statements to be proven and end up with a single proof system supporting some relaxed relation (generalizing the initial relation to be proven). These ideas are quite new, but promising, while extending and applying such methodology to new settings may have important efficiency implications.

The present work takes a step forward and shows a new way to generate succinct proofs for multiple instances of the same relation by folding the initial

statements all together (and not sequentially as prior works) into a single final proof for the relaxed relation. It also shows an alternative way of proving final folded statements, by designing a new version of the most efficient SNARK to date, Groth 16 [Gro16] that can be used by prior works to replace their last step and obtain better proof sizes and verification times.

1.1 Our Contribution

In the setting we consider, one single Prover needs to generate many different SNARKs for the same relation and then submit these proofs for validation. For the solution to be meaningful, the main requirement is that this external Prover should be able to generate proofs at a faster speed than the individual users and compress the proofs in some way.

We design a new solution that allows to prove together multiple R1CS instances for the same language at an amortized cost. We present our contribution in two building blocks that have independent interest for other applications as well.

First, we introduce FLIP, a new technique to folding R1CS instances that has minimal communication costs, efficient prover and verifier and does not require recursion. Our solution drives inspiration from Nova paper [KST22], that shows how to fold R1CS relations of the form $\mathbf{Az} \circ \mathbf{Bz} = \mathbf{Cz}$ with instance-witness pairs $\mathbf{z}$, avoiding cross terms. They define a new relaxed R1CS relation that preserves the same structure after folding: $\mathbf{Az} \circ \mathbf{Bz} = u\mathbf{Cz} + \mathbf{e}$, where u is a scaling constant and $\mathbf{e}$ is an error vector. Our scheme FLIP uses the same relaxed R1CS relations, but realizes their folding through inner pairing products [BMM+21] (IPP) instead of recursion. In contrast with Nova, the folding in FLIP does not require foreign-arithmetic to embed the verification algorithm in a proving circuit or expensive cycle of elliptic curves. Our folding step works over any efficient pairing-friendly curve. We leave it as an open question to realize folding for other relations such as Plonk via IPP to reduce communication complexity.

A second contribution and building block is an adaptation of the state-of-the-art constant-size SNARK system, Groth16 for proving relaxed R1CS relations. This new efficient constant-size Groth16-like proving system is compatible with other folding schemes using relaxed R1CS. It can be used to replace currently used Spartan [Set20] in the last step of Nova when the last step is in a bilinear group without the need of foreign-field arithmetic. This allows us to achieve very short proofs and constant verification time for the final folded proof, which is important in the context of limited communication costs of the blockchain scenario.

Our new scheme also needs a trusted setup, a structured reference string srs that is a slight modification of the original Groth16 trusted setup to make it a commit-and-prove SNARK with minimal communication complexity and optimized verification for the relaxed relation. It allows to prove multiple instances with a constant-size proof overhead (on top of FLIP proof size).

Reusing the same trusted setup is a strong requirement from an engineering point of view, as ceremonies to generate such setups are expensive and challenging to organize. Therefore, our version of Groth16 cannot be immediately deployed in practice for R1CS relations that have already a trusted setup srs for classic Groth16. The change in setup described in Section 4 comes from making Groth16 implicitly commit-and-prove and as succinct as possible, and to implicitly extract the witness and error terms. The alternative to achieve constant size proof without a new setup is to adapt a universal SNARK for R1CS such as Marlin [CHM+20] to the relaxed R1CS relation and to make it commit-and-prove.

1.2 Technical Overview

Our techniques consist in showing how to first fold many committed R1CS instances via Inner Pairing Argument using a two-tiered commitment scheme [AFG+10] resulting in a single relaxed R1CS instance-witness pair. Secondly, we design a SNARK proof for the folded relaxed R1CS language by starting from Groth16 system [Gro16].

Concretely, first we propose a new scheme, FLIP that combines folding with Inner Pairing Product Arguments (IPP). We first commit to the instance-witness pairs using a two-tiered commitment scheme similar to the ones proposed by [BMM+21, LMR19, GMN22b].

The construction is based on the folding techniques in Nova [KST22] for relaxed R1CS instances in order to fold k instances. Our construction builds on the fact that at the end of an IPP, if the input is a committed vector of group elements, the result is a linear combination with coefficients that have tensor structure. With this idea, we view IPP as a derandomization strategy for folding: at the end, instead of a fully random linear combination, the randomness has tensor structure. We think this point of view might be useful in other settings.

We show how the "flipping" of all k instances can be done in $\log k$ rounds and $O(\log k)$ communication (without counting the communication of the statement) by using Inner-Pairing Product techniques [BMM+21, BMM+21], which build on Bulletproofs [BBB+18, BCC+16] and two-tiered commitments.

For the generalization of Groth16, the main idea is that although the setup is non-universal, what makes it circuit specific is the way to prove linear relations, so there is hope to design a single proof system for relaxed R1CS relations with the same matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$. Our construction shows that this is possible without increasing the communication complexity.

Our construction results in massive savings in prover computation costs compared to simply applying Groth16 and SnarkPack combination. The folding step for k instance-witness pairs has roughly the same cost as applying SnarkPack to k distinct proofs. A prover has to commit to k R1CS instances and then fold them, while in SnarkPack the prover computes and commits to k Groth16 proofs and aggregates them (similarly to folding). Then, there is one Groth16 proof to run on the final folded statement (the size of the final statement is the same as the size of each original statements). This avoids the prover to compute

$k-1$ extra Groth16 proofs, saving on FFTs and multi-exponentiations (around 7 FFTs, and 3 multiexponentiations in G_1 and 1 multiexponentiation in G_2 per statement). For concrete values $k = 2^{10}$ and R1CS for circuits of size 2^{16}, we have estimated $2^{10}s$ overhead for computing the $k-1$ proofs in SnarkPack in [ark19] library.

Remark that our approach, as opposed to Nova folding [KST22], does not need to embed verification of correct folding in the final circuit to be proven via a SNARK, which is an important overhead in Nova, as this involves to compile and prove in circuit operations that are not SNARK-friendly, such as hashing. Also, we do not require cycles of curves, as in our last proving step, one has to show that the committed values satisfy a relaxed R1CS relation and this is done over the same curve.

1.3 Related works

Other techniques that aim at scaling SNARKs in the context of multi-instances proofs have been proposed in the recent years. These are great advancements into the research area, but very different in the setting they address or the improvement features they achieve to our result.

SnarkPack [GMN22b], as discussed previously is an aggregation scheme, that is designed to work externally of the SNARK proof generation, so the main advantage of SnarkPack offers is that aggregation does not require to know the original witnesses, but just take valid (or invalid) proofs and "pack" them together into a final proof to save on communication and verifier computation. Our advantage leverages the fact that in our scenario a single prover is computing all proofs, while in SnarkPack prover does not have to access the witnesses, but only behaves as a pot-proving aggregation node.

SnarkFold [LGZX23] is a very recent proof aggregation scheme that uses recursive proof composition to aggregate Groth16 proofs. This involves some recursion for proving that prior statements were correctly folded together and then generate the new proof. Such techniques allow to reduce the aggregated proof size and verification time, but at a high overhead for the prover that has to prove a recursive circuit that implement the verification of folding in order to save this computation on the verifier side.

Other schemes using folding for recursion, such as Supernova [KS22] and Hypernova [KS23] are recent generalizations of Nova. The first introduces a folding scheme that can handle multiple relations at once, while the latter constructs one that can handle arithmetizations with custom gates and lookup arguments. Protostar [BC23] achieves a combination of the previous two and Protogalaxy [EG23] serves as a more efficient version of the previous one by closely examining the trade-offs in prover, verifier and decider complexity when folding different amounts of instances in each step. Although recursion can achieve proof aggregation, this paper takes a simpler approach to tackle the problem.

1.4 Application

Filecoin proof-of-space blockchain. Filecoin network is currently the biggest SNARK consumer. It generates a large number of proofs each day: approximately 500,000 Groth16 proofs, representing 15 PiB of storage, that are aggregated in batches using SnarkPack.

Filecoin protocol [Lab18] is a descentralized storage network based on incentives, where storage providers have to prove they have encoded some data and they allocate space on their disk in order to win rewards. To accomplish this, storage providers need to prove they encoded each unit of storage (a partition of 32 GiB) using a complex encoding function, prove this initial computation and then show the persistence of the storage over time by answering queries on the encoded storage.

The specificity of Filecoin provers is that they generate all the proofs themselves and have access to all the statements and witnesses beforehand. SnarkPack, the solution used today in the Filecoin protocol, does not exploit this fact, and uses techniques that allow to aggregate proofs potentially created by different provers.

Our solution allows to save in cost generation for Filecoin proofs, leading to a more efficient storage provider (prover) and a reduction in the fees spent in the storage onboarding. A storage provider in Filecoin can simply save on proof generation as follows: they first encode the units of storage to onboard, generating the instance-witness pairs to be proven and then use our FLIP to fold the pairs and finally prove one single instance. This removes the need of computing many different proofs before the aggregation time.

Moreover, our work can unlock more interesting applications such as "Proving as a Service" nodes in the network, that allow users with low-computational resources to onboard storage, by delegating the proving part to this external Prover nodes. This is a very appealing scenario, since today, storage providers in Filecoin have to purchase extra specialized hardware in order to generate the proofs for the storage. Removing such requirements from the storage providers by outsourcing the proving to a specialized entity can make storage onboarding more accessible and allow for further network growth.

2 Preliminaries

Bilinear Groups. A bilinear group is given by a description $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathcal{P}_1, \mathcal{P}_2)$ such that

- p is prime, so $\mathbb{Z}_p = \mathbb{F}$ is a field.
- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p.
- Elements of these groups are sometimes given in implicit notation, i.e. $[a]_1 = a\mathcal{P}_1$, $[a]_2 = a\mathcal{P}_2$ and $[a]_T = ae(\mathcal{P}_1, \mathcal{P}_2)$.
- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map (pairing), meaning that $\forall a, b \in \mathbb{Z}_p$, $e([a]_1, [b]_2) \coloneqq [ab]_T$.
- Membership in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ can be efficiently decided, group operations and the pairing $e(\cdot, \cdot)$ are efficiently computable, and it is assumed that there is

no efficient algorithm to compute an isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$, and the descriptions of the groups and group elements each have linear size.

Polynomial-Time Algorithms. Unless otherwise specified, all the algorithms defined throughout this work are assumed to be probabilistic Turing machines with running time bounded by a polynomial in their input size, where the expectation is taken over the random coins of the algorithm - i.e., PPT. We denote the computational security parameter with $\lambda \in \mathbb{N}$.

2.1 Algebraic Group Model

The Algebraic Group Model (AGM), [FKL18] is an idealized model that captures the assumption that it is enough to prove security against adversaries that are algebraic, i.e. adversaries that output only group elements that are computed as a linear combination of previously seen group elements. Stronger models which give the adversary the additional ability to sample elements obliviously in the group [LPS23] will not be considered for simplicity.

In an algebraic security game $\mathbf{G}_{\mathsf{gk}}$, the parameters are set to a fixed group description, in this case an asymmetric bilinear group gk of order p. Adversaries are assumed to be algebraic:

- adversaries take as input the group description gk, a string of group elements $\boldsymbol{\sigma}$ (where each element is in one of the groups $\mathbb{G}_s$, $s \in \{1, 2, T\}$), and other inputs that are independent from all group elements;
- for each element Z in $\mathbb{G}_s$ that the adversary outputs, it must also provide a vector $\mathbf{z}$ of coefficients in $\mathbb{F}$ that explains Z with the valid operations in gk in terms of $\boldsymbol{\sigma}$, i.e. if $Z \in \mathbb{G}_1$, Z should be a linear combination of elements of $\mathbb{G}_1$ with coefficients $\mathbf{z}$, if $Z \in \mathbb{G}_2$, Z should be a linear combination of elements of $\mathbb{G}_2$ with coefficients $\mathbf{z}$ or if $Z \in \mathbb{G}_T$, it should be a linear combination with coefficients $\mathbf{z}$ of the inputs in $\mathbb{G}_T$ and the result of pairing any input of $\mathbb{G}_1$ with an input of $\mathbb{G}_2$.

2.2 Assumptions

Definition 1. *For a fixed integer q, the q-DLOG assumption holds if for every polynomial time adversary, the following probability is negligible in λ:*

$$\mathsf{Pr}\left[\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda), x \leftarrow \mathbb{Z}_p^* \ : \ x \leftarrow \mathcal{A}\left(\mathsf{gk}, \{[x^i]_1, [x^i]_2\}_{i=1}^q\right)\right].$$

We recall the following definition:

Definition 2. *([MRV16]) Let $\mathcal{D}_k$ be a distribution over vectors $\mathbf{y} = (y_0, \ldots, y_{k-1}) \in \mathbb{F}^k$. The $\mathcal{D}_k$-Kernel Matrix Diffie-Hellman Assumption holds in $\mathbb{G}_2$ if the following probability is negligible in λ:*

$$\mathsf{Pr}\left[\begin{matrix}\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda) \\ \mathbf{y} \leftarrow \mathcal{D}_k\end{matrix}, ([w_0]_1, \ldots, [w_{k-1}]_1) \leftarrow \mathcal{A}\left(\mathsf{gk}, [\mathbf{y}]_2\right) : \sum_{i=0}^{k-1} e([w_i]_1, [y_i]_2) = [0]_T\right].$$

For several distributions $\mathcal{D}_k$, this assumption directly allows to build binding two-tiered commitments [AFG+10], i.e. commitments in $\mathbb{G}_T$ to commitments in $\mathbb{G}_1$ that can be used with inner pairing product arguments. However, we want to use two-tiered commitments in a context where the adversary has vector $\mathbf{y}$ in both source groups. In this case, such a commitment is no longer binding, i.e. the previous assumption does not hold. For example, taking $k = 2$, given $\{[y_0]_1, [y_1]_1, [y_0]_2, [y_1]_2\}$, set $[w_0]_1 = -[y_1]_1$, and $[w_1]_1 = [y_0]_1$, then:

$$e([w_0]_1, [y_0]_2) + e([w_1]_1, [y_1]_2) = [0]_T.$$

A standard solution is to use two different vector of commitment keys $\mathbf{y}$ and $\mathbf{y}'$ but this implies doubling the protocol communication. Instead, inspired on [ABST22], we propose the following assumption:

Definition 3. *Let $\mathcal{D}_k$ be a distribution over vectors $\mathbf{y} = (y_0, \ldots, y_{k-1}) \in \mathbb{F}^k$ and $\mathcal{D}_n$ be a distribution over vectors $\mathbf{h} = (h_1, \ldots, h_n) \in \mathbb{F}^n$. The $(\mathcal{D}_k, \mathcal{D}_n)$- Kernel Diffie-Hellman Assumption with Opening states that the following probability is negligible:*

$$\mathsf{Pr}\left[\begin{array}{c} (\mathbf{c}_1, \ldots, \mathbf{c}_k) \leftarrow \mathcal{A}\big(\mathsf{gk}, \{[\mathbf{y}]_1, [\mathbf{y}]_2\}\big) \\ \mathbf{c}_i \in \mathbb{F}^n,\ [w_i]_1 = \langle \mathbf{c}_i, [\mathbf{h}]_1 \rangle \end{array} : \sum_{i=0}^{k-1} e([w_i]_1, [y_i]_2) = [0]_T \right].$$

The assumption states that it is hard find a vector of group elements $\mathbf{w}$ in the kernel of $\mathbf{y}$ together with their openings, i.e. together with an algebraic representation of each w_i with respect to some other commitment key $\mathbf{h}$, even when $\mathbf{y}$ is given in both groups. It is trivially true in the Algebraic Group Model [FKL18] for most standard distributions $(\mathcal{D}_k, \mathcal{D}_n)$ in which elements $\mathbf{y}$, $\mathbf{h}$ are chosen independently. In our final construction $\mathbf{y}$ are the powers of some trapdoor element, and $\mathbf{h}$ are the Lagrange polynomial commitments evaluated at some other trapdoor element chosen independently. The assumption is useful in the context of two-tiered commitments since it implies that if it is possible to extract the openings of commitments in group $\mathbb{G}_1$ with respect to a commitment key $[\mathbf{h}]_1$, then the commitments with respect to key $[\mathbf{y}]_2$ will be binding even if $[\mathbf{y}]_1$ is available to the committer.

2.3 Commitment Schemes

Definition 4. *A commitment scheme is a tuple of algorithms* $(\mathsf{SetupCom}, \mathsf{Com}, \mathsf{VerCom})$ *that work as follows:*

- $\mathsf{SetupCom}\left(1^\lambda\right) \to \mathsf{ck}$ *takes as input the security parameter in unary and returns the commitment key* ck*, and descriptions of the input space $\mathcal{D}$, commitment space $\mathcal{C}$ and opening space $\mathcal{O}$.*
- $\mathsf{Com}(\mathsf{ck}, v) \to (c, o)$ *takes the commitment key* ck *and a value $v \in \mathcal{D}$, and outputs a commitment c and an opening o.*
- $\mathsf{VerCom}(\mathsf{ck}, c, v, o) \to b$ *takes as input a commitment c, a value v and an opening o, and accepts $(b = 1)$ or rejects $(b = 0)$.*

A commitment scheme should satisfy the correctness and binding properties, and often they also satisfies a hiding property.

Correctness. For all $\lambda \in \mathbb{N}$ and any input $v \in \mathcal{D}$ we have:

$$\mathsf{Pr}\left[\mathsf{ck} \leftarrow \mathsf{Setup}\left(1^{\lambda}\right), (c, o) \leftarrow \mathsf{Com}(\mathsf{ck}, v) : \mathsf{VerCom}(\mathsf{ck}, c, v, o) = 1\right] = 1$$

Binding. For every polynomial-time adversary $\mathcal{A}$, the following probability is negligible in the security parameter

$$\mathsf{Pr}\left[\begin{array}{rl} v \neq v' \ \wedge & \mathsf{VerCom}(\mathsf{ck}, c, v, o) = 1 \\ \mathsf{ck} \leftarrow \mathsf{Setup}\left(1^{\lambda}\right), & (c, v, o, v', o') \leftarrow \mathcal{A}(\mathsf{ck}) \end{array} ; \mathsf{VerCom}\left(\mathsf{ck}, c, v', o'\right) = 1\right].$$

Hiding. For $\mathsf{ck} \leftarrow \mathsf{Setup}\left(1^{\lambda}\right)$ and $\forall v, v' \in \mathcal{D}$, the following two distributions are statistically close:

$$\mathsf{Com}(\mathsf{ck}, v) \approx \mathsf{Com}\left(\mathsf{ck}, v'\right).$$

2.4 Non-Interactive Arguments

A universal relation $\mathcal{R}$ is a set of triples (R, ϕ, w) where R is a relation, $\phi \in \mathcal{D}_{\phi}$ is called the instance (or input), $w \in \mathcal{D}_w$ the witness, and $\mathcal{D}_{\phi}, \mathcal{D}_w$ are domains that may depend on R. $\mathcal{R}_N$ is the subset of triples (R, ϕ, w) in $\mathcal{R}$ such that R has size at most N, for some size bound $N \in \mathbb{N}$. On input $1^{\lambda}, N$, the relation generator $\mathcal{RG}$ returns a relation $R \in \mathcal{R}_N$ and some auxiliary input aux that will be given to the adversary.

A (publicly verifiable) non-interactive argument for $\mathcal{R}$ is a quadruple of probabilistic polynomial algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ such that:

- $(R, \mathsf{srs}, \tau) \leftarrow \mathsf{Setup}(1^{\lambda}, N)$: On input the security parameter in unary λ, the relation generator $\mathcal{RG}$ returns a relation $R \in \mathcal{R}_N$ and a structured reference string srs, together with a set of trapdoors τ and an auxiliary output aux. The verifier has access to a subset of the structured reference string, $\mathsf{srs}_{\mathsf{V}}$.
- $\pi \leftarrow \mathsf{Prove}(R, \mathsf{srs}, \phi, w)$: On input a common reference string srs and $(\phi, w) \in R$, the prover algorithm returns a proof π.
- $0/1 \leftarrow \mathsf{Verify}(R, \mathsf{srs}_{\mathsf{V}}, \phi, \pi)$: The verification algorithm takes as input a common reference string $\mathsf{srs}_{\mathsf{V}}$, a statement ϕ and an argument π and returns 0 (reject) or 1 (accept).
- $\pi \leftarrow \mathsf{Sim}(R, \tau, \phi)$: The simulator takes as input the simulation trapdoor and statement ϕ and returns an argument π.

A non-interactive argument of knowledge for $\mathcal{R}$ should satisfy perfect completeness, computational knowledge soundness.

Perfect completeness. For all $\lambda, N \in \mathbb{N}, R \in \mathcal{R}_N, (\phi, w) \in R$

$$\mathsf{Pr}[(R, \mathsf{srs}, \tau) \leftarrow \mathsf{Setup}(1^{\lambda}, N); \pi \leftarrow \mathsf{Prove}(R, \mathsf{srs}, \phi, w) \mid \mathsf{Verify}(R, \mathsf{srs}_V, \phi, \pi) = 1] = 1.$$

Computational knowledge soundness. We say $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ is an argument of knowledge if for all non-uniform polynomial time adversaries $\mathcal{A}$ there exists a non-uniform polynomial time extractor $\mathcal{X}_{\mathcal{A}}$, such that the following probability is negligible in the security parameter:

$$\mathsf{Pr}\left[\begin{array}{c}(R, \mathsf{aux}, \mathsf{srs}, \tau) \leftarrow \mathsf{Setup}(1^\lambda, N); \\ ((\phi, \pi); w) \leftarrow (\mathcal{A} \| \mathcal{X}_{\mathcal{A}})\,(R, \mathsf{aux}, \mathsf{srs});\end{array} \middle| \; (\phi, w) \notin R \text{ and } \mathsf{Verify}(R, \mathsf{srs}_{\mathsf{V}}, \phi, \pi) = 1\right]$$

For some of the arguments considered in this paper, we will also prove the property of zero-knowledge.

Perfect zero-knowledge. $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ is perfect zero-knowledge if for all $\lambda, N \in \mathbb{N}$ and all adversaries $\mathcal{A}$, $(\phi, w) \in R$,

$$\begin{aligned}&\mathsf{Pr}\left[\begin{array}{c}(R, \mathsf{aux}, \mathsf{srs}, \tau) \leftarrow \mathsf{Setup}(1^\lambda, N); \\ \pi \leftarrow \mathsf{Prove}(R, \mathsf{srs}, \phi, w);\end{array} \middle| \; \mathcal{A}(R, \mathsf{aux}, \mathsf{srs}, \tau, \pi) = 1\right] \\ = &\mathsf{Pr}\left[\begin{array}{c}(R, \mathsf{aux}, \mathsf{srs}, \tau) \leftarrow \mathsf{Setup}(1^\lambda, N); \\ \pi \leftarrow \mathsf{Sim}(R, \mathsf{srs}, \phi);\end{array} \middle| \; \mathcal{A}(R, \mathsf{aux}, \mathsf{srs}, \tau, \pi) = 1\right]\end{aligned}$$

A non-interactive argument for $\mathcal{R}$ is universal if the setup runs in two phases, one that defines the set of trapdoors τ and which is common to any relation $R \in \mathcal{R}_N$, and another one which does not use τ and which derives a structured reference string specific to R.

2.5 Folding Schemes

Given a number M of instance/witness pairs (ϕ_i, w_i) that satisfy some NP relation, a folding scheme [KST22] outputs a new instance/witness pair (ϕ, w) that also satisfies the NP relation, along with a proof π that the new instance ϕ is indeed an aggregated or "folded" statement derived from the statements ϕ_i. Folding schemes have also been termed as reductions of knowledge [KP23], in the sense that knowledge of a witness for u implies knowledge of the k-witnesses that are folded.

The definition is adapted from [KST22] by extending it to folding multiple statements. Since our folding constructions are for a single structure, i.e. a single type of statement, we omit any reference to the structure type.

Definition 5 (M-Folding scheme). *Consider a relation R over public parameters, structure, instance, and witness tuples. A folding scheme for R consists of a PPT generator algorithm $\mathcal{G}$, a deterministic encoder algorithm $\mathcal{K}$, and a pair of interactive PPT algorithms $\mathcal{P}$ and $\mathcal{V}$, denoting prover and verifier, respectively, that behave as follows:*

- $\mathcal{G}(1^\lambda, M) \to \mathsf{pp}$: *On input security parameter λ, and a bound $M \in \mathsf{poly}(\lambda)$ this algorithm samples public parameters* pp.
- $\mathcal{K}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{vk})$: *On input* pp, *this algorithm outputs a prover key* pk *and a verifier key* vk*;*

– *We write*

$$\langle \mathcal{P}, \mathcal{V} \rangle \left(\mathsf{pk}, \mathsf{vk}, (\phi_1, w_1), \ldots, (\phi_m, w_m)\right)$$

to express the interaction between the prover and the verifier, where, for some $m \leq M$*, the prover takes as input* m *instance-witness pairs* $(\mathsf{pp}, \phi_1, w_1) \in R, \ldots, (\mathsf{pp}, \phi_m, w_m) \in R$ *and the prover's key* pk*, and the verifier takes as input* m *instance pairs* $\phi_1, \ldots, \phi_m$ *and the verifier's key* vk*. The interaction is treated as a function of the inputs that returns a pair* (ϕ, w)*.*

A folding scheme should satisfy the following properties:

1. **Completeness**: for all algorithms $\mathcal{A}$,

$$\Pr\left[(\mathsf{pp}, \phi, w) \in R \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, M), \\ (\phi_1, w_1), \ldots, (\phi_m, w_m) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \forall i, i \in [m], (\mathsf{pp}, \phi_i, w_i) \in R,\ m \leq M \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}), \\ (\phi, w) \leftarrow \langle \mathcal{P}, \mathcal{V} \rangle \left(\mathsf{pk}, \mathsf{vk}, (\phi_1, w_1), \ldots, (\phi_m, w_m)\right) \end{array}\right] = 1.$$

2. **Knowledge soundness**: there exists a PPT extractor $\mathcal{E}$ such that for all expected polynomial time adversaries $\mathcal{P}^*$

$$\Pr\left[\begin{array}{c} \forall i, i \in [m] \\ (\mathsf{pp}, \phi_i, w_i) \in R \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, M), \\ (\phi_1, \ldots, \phi_m, \mathsf{st}) \leftarrow \mathcal{P}^*(\mathsf{pp}, \mathsf{r}), m \leq M \\ (w_1, \ldots, w_m) \leftarrow \mathcal{E}(\mathsf{pp}, \mathsf{r}) \end{array}\right] \geq$$
$$\Pr\left[(\mathsf{pp}, \phi, w) \in R \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, M), \\ (\phi_1, \ldots, \phi_m, \mathsf{st}) \leftarrow \mathcal{P}^*(\mathsf{pp}, \mathsf{r}), m \leq M \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}) \\ (\phi, w) \leftarrow \langle \mathcal{P}^*, \mathcal{V}^* \rangle (\phi_1, \ldots, \phi_m, \mathsf{st}) \end{array}\right] - \mathsf{negl}(\lambda),$$

where r denotes an arbitrary long random tape.

Our folding schemes will be presented in interactive form following the definition presented above, but they are all public coin protocols that can be compiled to a non-interactive variant through the Fiat-Shamir heuristic.

2.6 Folding R1CS Instances

Given three matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}^{n \times (m+1)}$ that define an R1CS relation, NOVA [KST22] introduces a generalization, called *relaxed R1CS*. The interest of this new constraint system is that it is compatible with folding.

The relaxed R1CS language parameterized by these matrices and is defined as follows:

$$\mathcal{L}^{\text{relaxed}}_{\mathbf{A},\mathbf{B},\mathbf{C}} = \big\{(u, \mathbf{x}, \mathbf{e}) \in \mathbb{F} \times \mathbb{F}^l \times \mathbb{F}^n \mid \exists \mathbf{w} \in \mathbb{F}^{m-l} \text{ s.t.}$$
$$\mathbf{z} = \begin{pmatrix} u \\ \mathbf{x} \\ \mathbf{w} \end{pmatrix} \ \wedge\ \mathbf{Az} \circ \mathbf{Bz} = u\mathbf{Cz} + \mathbf{e}\big\}$$

This corresponds to the standard R1CS language when $u = 1$ and $\mathbf{e} = \mathbf{0}$. As several relations are folded together, the values u and $\mathbf{e}$ are constructed from random linear combinations of some cross terms. They can be viewed as some kind of "error terms".

The authors of NOVA also modify the language to be compatible with commit and prove techniques and define *Committed Relaxed R1CS*. We define it in the source group $\mathbb{G}_1$ of some pairing friendly elliptic curve, as this is where we will be using this scheme. The language is the following:

$$\begin{aligned}\mathcal{L}^{\text{c-relaxed}}_{\mathsf{ck},\widetilde{\mathsf{ck}},\mathbf{A},\mathbf{B},\mathbf{C}} = \big\{(u, \mathbf{x}, [e]_1, [w]_1) \in \mathbb{F} \times \mathbb{F}^l \times \mathcal{C} \times \mathcal{C} \mid \exists(\mathbf{w}, \mathbf{e}) \in \mathbb{F}^{m-l} \times \mathbb{F}^n \text{ s.t.} \\ [w]_1 = \mathsf{Com}(\mathsf{ck}, \mathbf{w}) \wedge [e]_1 = \widetilde{\mathsf{Com}}(\widetilde{\mathsf{ck}}, \mathbf{e}) \wedge \\ ((u, \mathbf{x}, \mathbf{e}), \mathbf{w}) \in \mathcal{R}^{\text{relaxed}}_{\mathbf{A},\mathbf{B},\mathbf{C}}\big\},\end{aligned}$$

where we denote with $\mathcal{C}$, respectively , $\widetilde{\mathcal{C}}$ the commitment space for two additively homomorphic commitment schemes with, respectively, key ck and key $\widetilde{\mathsf{ck}}$.

For completeness, we recall NOVA's 2-folding scheme for the latter relation in Fig. 1. In Section 3, we will implicitly be running the 2-folding protocol through an Inner Pairing Product Argument.

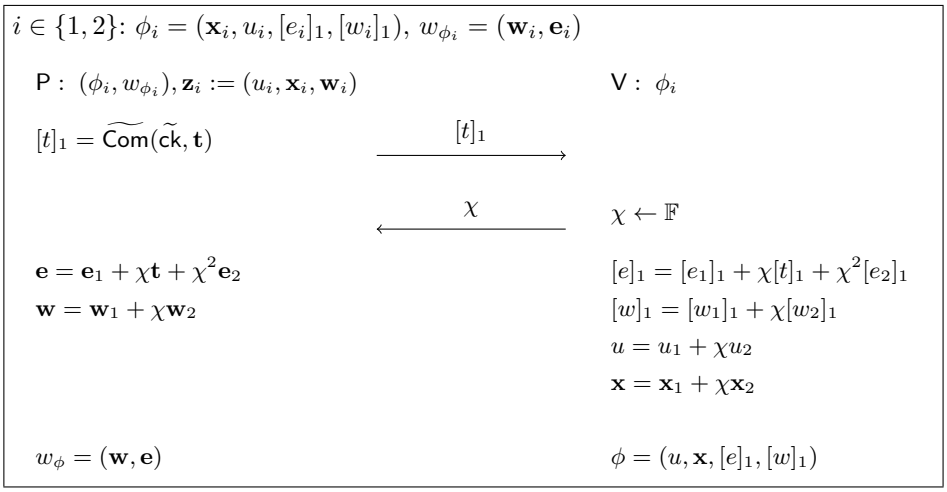

Fig. 1: Public coin protocol for folding committed relaxed R1CS statements [KST22], where $\mathbf{t} = \mathbf{A}\mathbf{z}_1 \circ \mathbf{B}\mathbf{z}_2 + \mathbf{A}\mathbf{z}_2 \circ \mathbf{B}\mathbf{z}_1 - u_1\mathbf{C}\mathbf{z}_2 - u_2\mathbf{C}\mathbf{z}_1$.

If commitments $[w]_1$ and $[e]_1$ are randomized, and the randomness space $\mathbb{F}^R$, $\mathbb{F}^{\widetilde{R}}$, then relation witness should include this randomness. That is, the relation should be defined as:

$$\begin{aligned}\mathcal{L}^{\text{c-relaxed}}_{\mathsf{ck},\widetilde{\mathsf{ck}},\mathbf{A},\mathbf{B},\mathbf{C}} = \big\{(u, \mathbf{x}, [e]_1, [w]_1) \in \mathbb{F} \times \mathbb{F}^l \times \mathcal{C} \times \widetilde{\mathcal{C}} \mid \exists(\mathbf{w}, \mathbf{e}, \mathbf{r}_w, \mathbf{r}_e) \in \mathbb{F}^{m-l} \times \mathbb{F}^n \times \mathbb{F}^R \times \mathbb{F}^{\widetilde{R}} \\ \text{s.t. } [w]_1 = \mathsf{Com}(\mathsf{ck}, \mathbf{w}; \mathbf{r}_w) \wedge [e]_1 = \widetilde{\mathsf{Com}}(\widetilde{\mathsf{ck}}, \mathbf{e}; \mathbf{r}_e) \wedge \\ (u, \mathbf{x}, \mathbf{e}, \mathbf{w}) \in \mathcal{R}^{\text{relaxed}}_{\mathbf{A},\mathbf{B},\mathbf{C}}\big\},\end{aligned}$$

but statements are folded analogously and the randomness necessary to open the folded commitments is the same linear combination of the randomness of the two statements as for the witness.

3 FLIP: Folding R1CS Instances via IPP

In this section we introduce FLIP, a new folding protocol that allows to aggregate k committed R1CS statements into a single committed relaxed R1CS statement that can later be proven very efficiently. Given k initial R1CS statements and their committed witnesses $(\mathbf{x}_i, [w_i]_1)_{0 \le i \le k-1}$, with $[w_i]_1 = \mathsf{Com}(\mathsf{ck}, \mathbf{w}_i)$, FLIP generates one final tuple $(u, \mathbf{x}, [e]_1, [w]_1) \in \mathcal{L}^{\text{c-relaxed}}_{\mathsf{ck},\widetilde{\mathsf{ck}},\mathbf{A},\mathbf{B},\mathbf{C}}$ for the corresponding committed relaxed R1CS language and a proof that the initial statements where folded correctly into the final statement.

In more detail, our setting is the following: a prover wants to prove k different R1CS statements expressed as committed relaxed R1CS statements, or equivalently, it wants to prove that, for $i = 0, \ldots, k-1$,

$$(u_i, \mathbf{x}_i, [e_i]_1, [w_i]_1) \in \mathcal{L}^{\text{c-relaxed}}_{\mathsf{ck},\widetilde{\mathsf{ck}},\mathbf{A},\mathbf{B},\mathbf{C}},$$

and $[e_i]_1 = [0]_1$, $u_i = 1$ for the same matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$.

We show how this can be done in $\log_2 k$ rounds and $O(\log_2 k)$ communication plus the cost of communicating the k public inputs by using Inner-Pairing Product (IPP) techniques [BMM+21], which build on Bulletproofs [BBB+18, BCC+16] and two-tiered commitments [AFG+10]. We assume without loss of generality that $k = 2^\mu$ for some $\mu \in \mathbb{N}$.

At the end of an inner product argument, given some committed vector of group elements $[\mathbf{v}]_1 \in \mathbb{G}_1^k$, the result is a single group element $[\hat{v}]_1 \in \mathbb{G}_1$ such that $\hat{v} = \langle \mathbf{v}, \otimes_{i=1}^{\log_2 k}(1, \alpha_i) \rangle$, where α_i are the verifier challenges. That is, it is a randomized linear combination of the coordinates of $\mathbf{v}$, where the randomness has tensor structure. We apply this observation and use the homomorphic commitments to end up with a single folded instance that is a R1CS relaxed instance where the randomness used to combine the statements has tensor structure. The verifier reads all the public inputs, but otherwise the communication is sublinear in k.

The proof has the usual recursive structure. In each round, ν statements

$$(u_i, \mathbf{x}_i, [e_i]_1, [w_i]_1) \in \mathcal{L}^{\text{c-relaxed}}_{\mathsf{ck},\widetilde{\mathsf{ck}},\mathbf{A},\mathbf{B},\mathbf{C}},$$

are reduced to new $\nu/2$ statements for the same language. At the beginning of the round prover and verifier both have commitments:

$$[W]_T = \sum_{i=0}^{\nu-1} e([w_i]_1, [q_i]_2),$$

and

$$[E]_T = \sum_{i=0}^{\nu-1} e([e_i]_1, [y_i]_2),$$

and field elements $u_0, \ldots, u_{\nu-1}$, and $\mathbf{x}_0, \ldots, \mathbf{x}_{\nu-1}$. The values $[W]_T, [E]_T$ are target group commitments under some keys $[\mathbf{q}]_2, [\mathbf{y}]_2$ which are part of the prover key for that round, pk_ν. In the first round, $\nu = k$ and the verifier should check that $[E]_T = [0]_T$, and $u_i = 1$, for all $i = 0, \ldots, k-1$, and the keys $[\mathbf{q}]_2, [\mathbf{y}]_2$ are fixed (and are not necessarily distinct).

Then, interactively, the statement is reduced to proving that, given new public inputs $\{u'_i, \mathbf{x}'_i\}_{i=0}^{\nu/2-1}$, and commitments

$$[W']_T = \sum_{i=0}^{\nu/2-1} e([w'_i]_1, [q'_i]_2)$$

$$[E']_T = \sum_{i=0}^{\nu/2-1} e([e'_i]_1, [y'_i]_2)$$

in the target group under keys $[\mathbf{q}']_2, [\mathbf{y}']_2$, it holds that for all $\{u'_i, \mathbf{x}'_i, [e'_i]_1, [w'_i]_1\}_{i=0}^{\nu/2-1}$ are satisfying instances of the same committed relaxed R1CS language.

In the last round, the prover opens the final $[W']_T$ and $[E']_T$ to some $[w]_1, [e]_1$, reducing the initial k claims about membership in R1CS to a single claim of membership in committed relaxed R1CS. We prove that our construction is a k-folding scheme, which means that, given as input the witness of the folded instance, i.e. an opening of $[w]_1, [e]_1$ with respect to $\mathsf{ck}, \widetilde{\mathsf{ck}}$, we can extract the witness of the original instances. In reality, the folding scheme will be used in combination with a proof system to prove the last step and the extractability of the witness can be derived from the knowledge soundness of the proof system.

Achieving Logarithmic Verifier. In each round, the verifier has to additionally check the correctness of the new halved commitment key. To avoid this linear overhead for the verifier, we can choose a structured commitment key and delegate the proof of the derived commitment keys to the prover, a technique due to Dory [Lee21], and also used in SnarkPack [GMN22a]. If the initial commitment key is $[\mathbf{y}^{(k)}]_2 = \{[y^i]_2\}_{i=0}^{k-1}$, for some $y \leftarrow \mathbb{Z}_p^\star$, then the final commitment key $[y^{(0)}]_2$ can be checked in the last round by the verifier in an additional constant-size proof. This is done by observing that $[y^{(0)}]_2$ is the commitment in the second source group under the original key $[\mathbf{y}^{(k)}]_2$ of a polynomial $g_\alpha(X)$ that has a logarithmic representation. Instead of calculating the commitment himself, the verifier computes a random opening of this polynomial and checks it against its supposed commitment $[y^{(0)}]_2$ and the evaluation proof sent by the prover. This process will be done for both $[y^{(0)}]_2$ and $[q^{(0)}]_2$.

Technical Overview of FLIP: The protocol is described in Fig. 3. We denote $[E_{LR}]_T$, the commitment to the left part of the vector of commitments under the right part of the key in the second group, i.e. $[E_{LR}]_T = \sum_{i=0}^{\nu/2-1} e([e_i]_1, [y_{i+\nu/2}]_2)$. Similarly for $[E_{RL}]_T, [W_{LR}]_T, [W_{RL}]_T$.

$\mathsf{pk} = \left(\mathsf{ck}, \widetilde{\mathsf{ck}}, \{[y^i]_2\}_{i=0}^{k-1}, [y]_1\right), \mathsf{vk} = ([y]_2)$

$\mathsf{P}:\ \{\alpha_i\}_{i=1}^{\mu}, [q^{(0)}, y^{(0)}]_2, [W]_T, [E]_T, \mathsf{pk}$		$\mathsf{V}:\ \{\alpha_i\}_{i=1}^{\mu}, \mathsf{vk}$
$g'_\alpha(X) = \prod_{i=1}^{\mu}\left(1 + \alpha_{\mu+1-i}^{-1} X^{2^{i-1}}\right)$		
$g_\alpha(X) = \prod_{i=1}^{\mu}\left(1 + \alpha_{\mu+1-i}^{-2} X^{2^{i-1}}\right)$		
$[q^{(0)}]_2 = [g'(q, \alpha_1, \ldots, \alpha_\mu)]_2$		
$[y^{(0)}]_2 = [g(y, \alpha_1, \ldots, \alpha_\mu)]_2$		
$[W]_T = e([w]_1, [q^{(0)}]_2)$		
$[E]_T = e([e]_1, [y^{(0)}]_2)$		
	$\xrightarrow{[w]_1, [e]_1, [q^{(0)}, y^{(0)}]_2}$	
		$r, \xi \leftarrow \mathbb{F}$,
		$u = g_\alpha(r), u' = g'_\alpha(r)$
	$\xleftarrow{r, \xi}$	
$u = g_\alpha(r), u' = g'_\alpha(r)$		
$\pi(X) = \frac{g_\alpha(X) - u}{X - r} + \xi \frac{g'_\alpha(X) - u'}{X - r}$		
$[\pi]_2 \coloneqq [\pi(y)]_2 = \sum_{i=1}^{k-1} \pi_i [y^i]_2$		
	$\xrightarrow{[\pi]_2}$	
		$u'' = u + \xi u'$
		$[\psi]_2 = [y^{(0)}] + \xi [q^{(0)}]$
		$e([1]_1, [\psi]_2 - [u'']_2)$
		$\stackrel{?}{=} e([y]_1 - [r]_1, [\pi]_2)$
		$e([w]_1, [q^{(0)}]_2) \stackrel{?}{=} [W]_T$
		$e([e]_1, [y^{(0)}]_2) \stackrel{?}{=} [E]_T$

Fig. 2: Last round of the FLIP protocol. It proves the statement $[y^{(0)}]_2 = [g_\alpha(y)]_2$, $[q^{(0)}]_2 = [g'_\alpha(y)]_2$ and that the openings of $[W]_T, [E]_T$ are correct. If $\{[y^i]_1\}_{i=0}^{k-1}$ is included in the prover key pk, the proof of correct opening can be computed in $\mathbb{G}_1$ to save prover work.

$\mathsf{pk}_\nu = (\widetilde{\mathsf{ck}}, [\mathbf{y}]_2 \in \mathbb{G}_2^\nu, [\mathbf{q}]_2 \in \mathbb{G}_2^\nu)$

$\Phi := ([W]_T, [E]_T, \{(u_i, \mathbf{x}_i)\}_{i=0}^{\nu-1}), w_\Phi := \{(([e_i]_1, [w_i]_1); (\mathbf{w}_i, \mathbf{e}_i, \mathbf{r}_{w_i}, \mathbf{r}_{e_i}))\}_{i=0}^{\nu-1}$ where $[W]_T = \sum_{i=0}^{\nu-1} e([w_i]_1, [q_i]_2)$, $[E]_T = \sum_{i=0}^{\nu-1} e([e_i]_1, [y_i]_2)$, $[w_i]_1 = \mathsf{Com}(\mathsf{ck}, \mathbf{w}_i; \mathbf{r}_{w_i})$ and $[e_i]_1 = \widetilde{\mathsf{Com}}(\widetilde{\mathsf{ck}}, \mathbf{e}_i; \mathbf{r}_{e_i})$.

$\mathsf{P}: (\Phi, w_\Phi), \mathsf{pk}_\nu$	$\mathsf{V}: \Phi, \mathsf{vk}_\nu$

$\mathbf{z}_i = (u_i, \mathbf{x}_i^\top, \mathbf{w}_i^\top)^\top$

$\mathbf{t}_i = \mathbf{A}\mathbf{z}_i \circ \mathbf{B}\mathbf{z}_{i+\nu/2} + \mathbf{A}\mathbf{z}_{i+\nu/2} \circ \mathbf{B}\mathbf{z}_i - u_i\mathbf{C}\mathbf{z}_{i+\nu/2} - u_{i+\nu/2}\mathbf{C}\mathbf{z}_i$

Compute:

$[t_i]_1 = \widetilde{\mathsf{Com}}(\widetilde{\mathsf{ck}}, \mathbf{t}_i; \mathbf{r}_{t_i}), \mathbf{r}_{t_i} \leftarrow \mathbb{F}^{\tilde{R}}$;

$[T_L]_T = \sum_{i=0}^{\nu/2-1} e([t_i]_1, [y_i]_2)$; $[T_R]_T = \sum_{i=0}^{\nu/2-1} e([t_{\nu/2+i}]_1, [y_{\nu/2+i}]_2)$

$[E_{LR}]_T = \sum_{i=0}^{\nu/2-1} e([e_i]_1, [y_{\nu/2+i}]_2)$; $[E_{RL}]_T = \sum_{i=0}^{\nu/2-1} e([e_{\nu/2+i}]_1, [y_i]_2)$

$[W_{LR}]_T = \sum_{i=0}^{\nu/2-1} e([w_i]_1, [q_{\nu/2+i}]_2)$; $[W_{RL}]_T = \sum_{i=0}^{\nu/2-1} e([w_{\nu/2+i}]_1, [q_i]_2)$

$\mathcal{C} = \{[T_L, T_R, E_{LR}, E_{RL}, W_{LR}, W_{RL}]_T\}$

$\xrightarrow{\mathcal{C}}$

$\xleftarrow{\alpha \leftarrow \mathbb{F}}$

Compute:

$[E']_T = [E]_T + \alpha^{-2}[E_{LR}]_T + \alpha[T_L]_T + \alpha^{-1}[T_R]_T + \alpha^2[E_{RL}]$

$[W']_T = [W]_T + \alpha^{-1}[W_{LR}]_T + \alpha[W_{RL}]_T$

$\mathbf{x}'_i = \mathbf{x}_i + \alpha\mathbf{x}_{i+\nu/2}$

$u'_i = u_i + \alpha u_{i+\nu/2}$

$\mathbf{w}'_i = \mathbf{w}_i + \alpha\mathbf{w}_{i+\nu/2}, \mathbf{r}_{w'_i} = \mathbf{r}_{w_i} + \alpha\mathbf{r}_{w_{i+\nu/2}}$

$\mathbf{e}'_i = \mathbf{e}_i + \alpha\mathbf{t}_i + \alpha^2\mathbf{e}_{i+\nu/2}, \mathbf{r}_{e'_i} = \mathbf{r}_{e_i} + \alpha\mathbf{r}_{t_i} + \alpha^2\mathbf{r}_{e_{i+\nu/2}}$

$[e'_i]_1 = [e_i]_1 + \alpha[t'_i]_1 + \alpha^2[e_{i+\nu/2}]_1$

$[w'_i]_1 = [w_i]_1 + \alpha[w_{i+\nu/2}]_1$

$[y'_i]_2 = [y_i]_2 + \alpha^{-2}[y_{i+\nu/2-1}]_2$

$[q'_i]_2 = [q_i]_2 + \alpha^{-1}[q_{i+\nu/2-1}]_2$

Set:

$\Phi' = ([W']_T, [E']_T, \{(u'_i, \mathbf{x}'_i)\}_{i=0}^{\nu/2-1})$

$w_{\Phi'} = \left\{(([e'_i]_1, [w'_i]_1); (\boldsymbol{w}'_i, \boldsymbol{e}'_i, \mathbf{r}_{w'_i}, \mathbf{r}_{e'_i}))\right\}$

$\mathsf{pk}_{\nu/2} = (\widetilde{\mathsf{ck}}, [\mathbf{y}']_2 \in \mathbb{G}_2^{\nu/2}, [\mathbf{q}']_2 \in \mathbb{G}_2^{\nu/2})$

Output:

$(\Phi', w_{\Phi'}), \mathsf{pk}_{\nu/2}$

Compute:

$[E']_T = [E]_T + \alpha^{-2}[E_{LR}]_T + \alpha[T_L]_T + \alpha^{-1}[T_R]_T + \alpha^2[E_{RL}]$

$[W']_T = [W]_T + \alpha^{-1}[W_{LR}]_T + \alpha[W_{RL}]_T$

$\mathbf{x}'_i = \mathbf{x}_i + \alpha\mathbf{x}_{i+\nu/2}$

$u'_i = u_i + \alpha u_{i+\nu/2}$

Set:

$\Phi' = ([W']_T, [E']_T \{(u'_i, \mathbf{x}'_i)\}_{i=0}^{\nu/2-1}$

Output:

Φ'

Fig. 3: FLIP: Public coin protocol for folding committed relaxed R1CS statements through inner pairing product.

We start from two copies of one initial key $[y^{(k)}]_2 = ([1]_2, [y]_2, [y^2]_2, \ldots, [y^{k-1}]_2)$ to commit to W_i on one hand, and E_i on the other. At each step of the protocol one of the copies is folded by linearly combining the two halves with $\alpha^{(-1)}$ for the W_i's and with $\alpha^{(-2)}$ for the $E_i's$. This choice is justified by the folding formulae detailed in Fig. 1, where e_i, w_i are combined differently

Denote the keys generated in the final round of folding $k = 2^\mu$ instances as $[q^{(0)}]_2, [y^{(0)}]_2$. Then it holds that $[q^{(0)}]_2 =< [y^{(k)}]_2, \mathbf{j} >$ and $[y^{(k)}]_2 =< [y^{(k)}]_2, \mathbf{s} >$, where

$$\mathbf{j} = \otimes_{j=1}^{k}(1, \alpha_i^{-1}).$$

$$\mathbf{s} = \otimes_{j=1}^{k}(1, \alpha_i^{-2}).$$

The last round of the protocol, which proves the correctness of the folded keys, is described in Fig. 2. Define the polynomials $g(X, \alpha_1, \ldots, \alpha_\mu) = g_\alpha(X) = \prod_{i=1}^{\mu}\left(1 + \alpha_{\mu+1-i}^{-2} X^{2^{i-1}}\right)$. The vector of coefficients of $g(X)$ is $\mathbf{s}$. Then the final key $[y^{(0)}]_2$ is the commitment to the $\mathbf{s}$ under the original key, i.e. $[y^{(0)}]_2 = \mathsf{Com}_{\mathsf{ck}_2}(\mathbf{s})$, for $\mathsf{ck}_2 = \{[y_0^{(k)}]_2, \ldots, [y_{k-1}^{(k)}]_2\}$. Because g can be evaluated at any point $r \in \mathbb{F}$ in logarithmic time by the verifier, the prover can send $[y^{(0)}]_2$, ask for an evaluation point and respond with a KZG evaluation proof $[\pi]_2$. The verifier evaluates $g(r, \alpha_1, \ldots, \alpha_\mu)$ and checks it against the evaluation proof and the commitment to get convinced of the correctness of $[y^{(0)}]_2$ in logarithmic time.

In more detail, write $g_\alpha(X) = g(X, \alpha_1, \ldots, \alpha_\mu)$, the prover sends $[g_\alpha(y)]_2 = [y^{(0)}]_2$ and the verifier responds with $r \in \mathbb{F}$. Both evaluate $[u]_2 = [g_\alpha(r)]_2$, and the prover demonstrates that $[y^{(0)}]_2$, the commitment to g_α, correctly opens to u. The same thing applies for the polynomial $g'(X, \alpha_1, \ldots, \alpha_\mu) = g'_\alpha(X) = \prod_{i=1}^{\mu}\left(1 + \alpha_{\mu+1-i}^{-1} X^{2^{i-1}}\right)$ and key $[q^{(0)}]_2$. Because the initial keys are powers of the same trapdoor, the KZG evaluation proofs can be batched.

Completeness: If we remove the second layer of commitments and observe only the commitments that the prover is making in group $\mathbb{G}_1$, the prover is actually folding k R1CS statements in parallel (with one challenge per level), building a tree of folded statements so that the final folded statement is the root of a tree. Completeness, therefore, follows from the completeness of the folding strategy in NOVA, and the fact that the commitments are homomorphic.

Knowledge Soundness: (Intuition) With the same analysis as IPP Arguments [BMM+21], which is similar to [BBB+18, BCC+16], we can show that if the commitment under the key $\mathsf{ck}_2 = [\mathbf{y}^{(k)}]_2$ is binding, then we can extract commitments to $[w_1]_1, \ldots, [w_k]_1$ such that the final value $[w]_1$,$[e]_1$,u, is a correct folding of these commitments according to the folding strategy for committed relaxed R1CS described in 2.6. Then it follows from the results of NOVA [KST22], that if $[w]_1$,$[e]_1$,u are a valid committed relaxed R1CS statement, then so are the original statements.

In practice, the key that we might want to use is the result of some already existing setup ceremony so that the same key is given in both source groups,

$\mathsf{ck}_2 = ([\mathbf{y}^{(k)}]_1, [\mathbf{y}^{(k)}]_2)$. Unfortunately, this implies that the key $[\mathbf{y}^{(k)}]_2$ is not binding, or in other words it is possible to find non-trivial relations. For example, it is easy to find $[a]_1, [b]_1$ such that $e([a]_1, [y_i^{(k)}]_2) = e([b]_1, [y_j^{(k)}]_2)$ choosing $[a]_1 = [y_j^{(k)}]_2, [b]_1 = [y_i^{(k)}]_2$. To solve this issue, SnarkPack proposes to uses a pair of independent commitment keys $\mathsf{ck}_2, \mathsf{ck}_2'$ in $\mathbb{G}_2$. This solution is relatively expensive, as essentially the protocol is executed twice. Instead, we use a single key, following an idea of aPlonk [ABST22]. More specifically, we can avoid using two keys because we are always committing in the target group to extractable commitments in $\mathbb{G}_1$ under keys $\mathsf{ck}, \widetilde{\mathsf{ck}}$. If $\mathsf{ck}, \widetilde{\mathsf{ck}}$ are independently chosen of $\mathbf{y}$, the target group commitment is still binding. The intuition is that the only way to find non-trivial pairing product relations involves creating group elements in $\mathbb{G}_1$ that are in the span of $[\mathbf{y}^{(k)}]_1$, but if, as part of the proof we prove that these commitments are extractable with respect to other, independently chosen keys, it is no longer possible to find these relations. This is captured by the Algebraic $(\mathcal{D}_k, \mathcal{D}_m)$- Kernel Matrix Diffie-Hellman Assumption.

Our protocol in this section assumes the commitments are algebraic, that is commitments where the keys are vectors of group elements in $\mathbb{G}_1$, the commitments and the randomness are in $\mathbb{F}$, and the commitment is of the form $\mathsf{Com}(\mathsf{ck}, \mathbf{w}; \mathbf{r}_w) = \langle (\mathbf{w}||\mathbf{r}_w), \mathsf{ck} \rangle$, $\widetilde{\mathsf{Com}}(\widetilde{\mathsf{ck}}, \mathbf{e}; \mathbf{r}_e) = \langle (\mathbf{e}||\mathbf{r}_e), \widetilde{\mathsf{ck}} \rangle$ (any Pedersen type commitment), for some distributions of the keys $\mathsf{ck} \leftarrow \mathcal{D}_{m-l+R}$, $\widetilde{\mathsf{ck}} \leftarrow \widetilde{\mathcal{D}}_{n+\widetilde{R}}$. Then, knowledge soundness holds under $(\mathcal{D}_k, \mathcal{D}_{m-l+R})$- Kernel Matrix Diffie-Hellman Assumption with Opening and the $(\mathcal{D}_k, \widetilde{\mathcal{D}}_{n+\widetilde{R}})$- Kernel Matrix Diffie-Hellman Assumption with Opening. For any $\mathsf{ck}, \widetilde{\mathsf{ck}}$ which satisfies this assumption the protocol is sound. However, our version of $\mathsf{Groth16}$ for committed relaxed instances (Section 4), requires specific distributions for ck and $\widetilde{\mathsf{ck}}$.

Theorem 1. *If $\mathsf{ck} \leftarrow \mathcal{D}_{m-l+R}, \widetilde{\mathsf{ck}} \leftarrow \widetilde{\mathcal{D}}_{n+\widetilde{R}}$, the protocol described in Fig. (2,3) is a k-folding scheme with perfect completeness and knowledge soundness under the $(\mathcal{D}_k, \mathcal{D}_{m-l+R})$- Kernel Matrix Diffie-Hellman Assumption with Opening and the $(\mathcal{D}_k, \widetilde{\mathcal{D}}_{n+\widetilde{R}})$- Kernel Matrix Diffie-Hellman Assumption with Opening.*

Proof. For simplicity we are going to prove the theorem for $k = 2$, but the generalization to arbitrary length is straightforward. Also, we will present separately the extraction of $[\boldsymbol{w}]_1$ and $[\boldsymbol{e}]_1$ values through rewinding, but in fact extraction can be done simultaneously by rewinding as much as the most demanding case requires.

Assume the verifier interacts with the prover and constructs two final target group commitments $[W']_T, [E']_T$ and that the final keys pass the verification, i.e.

$$[y^{(0)}]_2 = [y_1]_2 + \alpha^{-2}[y_2]_2, \quad [q^{(0)}]_2 = [q_1]_2 + \alpha^{-1}[q_2]_2 \tag{1}$$

We show first how to extract $[\boldsymbol{w}]_1$. By construction, we have that,

$$[W']_T = [W]_T + \alpha^{-1}[W_{LR}]_T + \alpha[W_{RL}]_T \tag{2}$$

The prover also proves that $[W']_T$ opens to $[w']_1$ under the correct final key $[q^{(0)}]_2$, i.e.,

$$e([w']_1, [q^{(0)}]_2) = [W']_T. \tag{3}$$

We will show that (1) $[W]_T = e([\delta_1]_1, [q_1]_2) + e([\delta_2]_1, [q_2]_2)$ and (2) $[w']_1 = [\delta_1]_1 + \alpha[\delta_2]_1$, that is, we will show that we can extract an opening of $[W]_T$ under the previous key and that the linear combination of the extracted opening gives $[w']_1$ thus completing the proof.

From equations (1),(2),(3) we get that:

$$e([w']_1, [q_1]_2) + \alpha^{-1} e([w']_1, [q_2]_2) = [W]_T + \alpha^{-1}[W_{LR}]_T + \alpha[W_{RL}]_T \tag{4}$$

Rewinding the prover 4 times we get $\alpha_i, [w'_i]_1$ that satisfy equation (4). We use 3 of them to find ν_i such that the following hold:

$$\sum_{i=1}^{3} \nu_i \alpha_i^{\gamma} = 0, \qquad \forall \gamma \in \{-1, 0, 1\}$$

Now, linearly combining the equations (4) with these ν_i we get that:

$$[W]_T = e([\beta_1]_1, [q_1]_2) + e([\beta_2]_1, [q_2]_2) \tag{5}$$

Similarly, by finding different ν_i, we get that:

$$[W_{LR}]_T = e([\beta_3]_1, [q_1]_2) + e([\beta_4]_1, [q_2]_2) \tag{6}$$

$$[W_{RL}]_T = e([\beta_5]_1, [q_1]_2) + e([\beta_6]_1, [q_2]_2) \tag{7}$$

Combining equations (4),(5),(6),(7) we get that the following holds for each $[w']_1$ and corresponding α:

$$\begin{aligned} &e([w']_1, [q_1]_2) + e(\alpha^{-1}[w']_1, [q_2]_2) = \\ &e([\beta_1]_1 + \alpha^{-1}[\beta_3]_1 + \alpha[\beta_5]_1, [q_1]_2) + \\ &\quad e([\beta_2]_1 + \alpha^{-1}[\beta_4]_1 + \alpha[\beta_6]_1, [q_2]_2) \end{aligned} \tag{8}$$

This means that the prover breaks the $(\mathcal{D}_k, \mathcal{D}_{m-l+R})$- Kernel Diffie-Hellman with Opening Assumption or the following equations hold for each $[w']_1$ and corresponding α:

$$[w']_1 = [\beta_1]_1 + \alpha^{-1}[\beta_3]_1 + \alpha[\beta_5]_1 \tag{9}$$

$$[w']_1 = \alpha[\beta_2]_1 + [\beta_4]_1 + \alpha^2[\beta_6]_1 \tag{10}$$

From equations (9), (10) we have:

$$[\beta_3]_1 + \alpha([\beta_1]_1 - [\beta_4]_1) + \alpha^2([\beta_5]_1 - [\beta_2]_1) + \alpha^3[\beta_6]_1 = 0 \tag{11}$$

Because equation (11) holds for 4 α_i, except with negligible probability, we have that:

$$[\beta_3]_1 = [\beta_6]_1 = 0, \qquad [\beta_1]_1 = [\beta_4]_1, \qquad [\beta_5]_1 = [\beta_2]_1 \tag{12}$$

Finally, from equations (5),(9), (12), we have that:

$$[W]_T = e([\beta_1]_1, [q_1]_2) + e([\beta_2]_1, [q_2]_2), \qquad [w']_1 = [\beta_1]_1 + \alpha[\beta_2]_1.$$

We move now to showing the case of $[\boldsymbol{e}]_1$.

Because of the construction, we have that,

$$[E']_T = [E]_T + \alpha^{-2}[E_{LR}]_T + \alpha[T_L]_T + \alpha^{-1}[T_R]_T + \alpha^2[E_{RL}], \tag{13}$$

where $[E]_T$ is the previous target group commitment. The prover also proves that $[E']_T$ opens to $[e']_1$ under the correct final key $[y']_2$, i.e.

$$e([e']_1, [y^{(0)}]_2) = [E']_T. \tag{14}$$

We will show that (1) $[E]_T = e([\delta_1]_1, [y_1]_2) + e([\delta_2]_1, [y_2]_2)$, (2) $[T_L]_T = e([\delta_3]_1, [y_1]_2)$, (3) $[T_R]_T = e([\delta_3]_1, [y_2]_2)$ and (4) $[e']_1 = [\delta_1]_1 + \alpha[\delta_3]_1 + \alpha^2[\delta_2]_1$, that is, we will show that we can extract an opening of $[E]_T$ under the previous key, the same opening for $[T_L]_T, [T_R]_T$ under each of the previous keys, and that the linear combination of the extracted openings gives $[e']_1$, thus completing the proof.

From equations (1),(13),(14) we get that:

$$e([e']_1, [y_1]_2) + \alpha^{-2}e([e']_1, [y_2]_2) = [E]_T + \alpha^{-2}[E_{LR}]_T + \alpha[T_L]_T + \alpha^{-1}[T_R]_T + \alpha^2[E_{RL}] \tag{15}$$

Rewinding the prover 7 times we get $\alpha_i, [e'_i]_1$ that satisfy equation (15). We use 5 of them to find ν_i such that the following hold:

$$\sum_{i=1}^{5} \nu_i = 1, \qquad \sum_{i=1}^{5} \nu_i \alpha_i^{\gamma} = 0, \qquad \forall \gamma \in \{-2, -1, 1, 2\}$$

Now, linearly combining the equations (15) with these ν_i we get that:

$$[E]_T = e([\beta_1]_1, [y_1]_2) + e([\beta_2]_1, [y_2]_2) \tag{16}$$

Similarly, by finding different ν_i, we get that:

$$[E_{LR}]_T = e([\beta_3]_1, [y_1]_2) + e([\beta_4]_1, [y_2]_2) \tag{17}$$

$$[T_L]_T = e([\beta_5]_1, [y_1]_2) + e([\beta_6]_1, [y_2]_2) \tag{18}$$

$$[T_R]_T = e([\beta_7]_1, [y_1]_2) + e([\beta_8]_1, [y_2]_2) \tag{19}$$

$$[E_{RL}]_T = e([\beta_9]_1, [y_1]_2) + e([\beta_{10}]_1, [y_2]_2) \tag{20}$$

Combining equations (15),(16),(17),(18),(19),(20), we get that the following holds for each $[e']_1$ and corresponding α:

$$\begin{aligned} e([e']_1, [y_1]_2) + e(\alpha^{-2}[e']_1, [y_2]_2) = \\ e([\beta_1]_1 + \alpha^{-2}[\beta_3]_1 + \alpha[\beta_5]_1 + \alpha^{-1}[\beta_7]_1 + \alpha^2[\beta_9]_1, [y_1]_2) + \\ e([\beta_2]_1 + \alpha^{-2}[\beta_4]_1 + \alpha[\beta_6]_1 + \alpha^{-1}[\beta_8]_1 + \alpha^2[\beta_{10}]_1, [y_2]_2) \end{aligned} \tag{21}$$

This means that either the prover breaks the $(\mathcal{D}_k, \widetilde{\mathcal{D}}_{n+\widetilde{R}})$- Kernel Diffie-Hellman with Opening Assumption or the following equations hold for each $[e']_1$ and corresponding α:

$$[e']_1 = [\beta_1]_1 + \alpha^{-2}[\beta_3]_1 + \alpha[\beta_5]_1 + \alpha^{-1}[\beta_7]_1 + \alpha^2[\beta_9]_1 \tag{22}$$

$$[e']_1 = \alpha^2[\beta_2]_1 + [\beta_4]_1 + \alpha^3[\beta_6]_1 + \alpha[\beta_8]_1 + \alpha^4[\beta_{10}]_1 \tag{23}$$

From equations (22), (23), we have:

$$\begin{aligned} [\beta_3]_1 + \alpha[\beta_7]_1 + \alpha^2([\beta_1]_1 - [\beta_4]_1) + \alpha^3([\beta_5]_1 - [\beta_8]_1) \\ + \alpha^4([\beta_9]_1 - [\beta_2]_1) - \alpha^5[\beta_6]_1 - \alpha^6[\beta_{10}]_1 = 0 \end{aligned} \tag{24}$$

Because equation (24) holds for 7 α_i, except with negligible probability, we have that:

$$\begin{aligned} [\beta_3]_1 = [\beta_6]_1 = [\beta_7]_1 = [\beta_{10}]_1 = 0, \qquad [\beta_1]_1 = [\beta_4]_1, \qquad [\beta_5]_1 = [\beta_8]_1, \\ [\beta_2]_1 = [\beta_9]_1 \end{aligned} \tag{25}$$

Finally, from equations (16),(18), (19),(25) we have that:

$$[E]_T = e([\beta_1]_1, [y_1]_2) + e([\beta_2]_1, [y_2]_2), \qquad [e']_1 = [\beta_1]_1 + \alpha[\beta_5]_1 + \alpha^2[\beta_2]_1$$

$$[T_L]_T = e([\beta_5]_1, [y_1]_2) \qquad [T_R] = e([\beta_5]_1, [y_2]_2)$$

thus concluding the proof.

4 Groth16 for Relaxed R1CS

Groth16 proves the satisfiability of a R1CS constraint system. It is well-known for its short proof size and verifier efficiency, but its structured reference string depends on the specific R1CS relation it proves. We propose a proof system to prove relaxed R1CS instances stemming from the same R1CS instance, i.e. with the same matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$. The key insight for our new scheme is based on the observation that the "non-universal" part of Groth16 are only the linear relations defined by $\mathbf{A}, \mathbf{B}, \mathbf{C}$.

We also show how to achieve zero-knowledge. We note that the FLIP protocol presented in last section does not reveal more about the witness than the commitments to $[e_i]_1, [w_i]_1$. If these are sufficiently randomized and hide the witness, using our generalized Groth16 proof system on top will not reveal more information, and the zero-knowledge property of the composition easily follows.

If $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathcal{P}_1, \mathcal{P}_2)$ is a bilinear group, and $\mathbb{F}$ a prime field of cardinal p, we aim to design a generalization of Groth16 that allows to prove membership in the following language:

$$\begin{aligned}\mathcal{L}^{\text{c-relaxed}}_{\mathsf{ck},\widetilde{\mathsf{ck}},\mathbf{A},\mathbf{B},\mathbf{C}} = \big\{&(u, \mathbf{x}, [e]_1, [w]_1) \in \mathbb{F} \times \mathbb{F}^l \times \mathbb{G}_1 \times \mathbb{G}_1 \mid \\ &\exists (\mathbf{w}, \mathbf{e}, r_w, r_e) \in \mathbb{F}^{m-l} \times \mathbb{F}^n \times \mathbb{F} \times \mathbb{F} \text{ s.t.} \\ &[w]_1 = \mathsf{Com}_{\mathsf{ck}}(\mathsf{pp}, \mathbf{w}; r_w) \wedge [e]_1 = \mathsf{Com}_{\widetilde{\mathsf{ck}}}(\mathsf{pp}, \mathbf{e}; r_e) \wedge \\ &((u, \mathbf{x}, \mathbf{e}), \mathbf{w}) \in \mathcal{R}^{\text{relaxed}}_{\mathbf{A},\mathbf{B},\mathbf{C}}\big\},\end{aligned}$$

for certain $\mathsf{ck}, \widetilde{\mathsf{ck}}$. In Sec. 3, the key ck could any binding commitment key, whereas here we choose:

$$\mathsf{ck} = ([\ell_{l+1}(x)/\rho]_1, \ldots, [\ell_m(x)/\rho]_1, [\delta/\rho]_1) \in \mathbb{G}_1^{m-l+1},$$

$$\widetilde{\mathsf{ck}} = ([\ell_0(x)/\psi]_1, \ldots, [\ell_{n-1}(x)/\psi]_1, [\delta/\psi]_1) \in \mathbb{G}_1^{n+1},$$

for some $x, \rho, \psi \leftarrow \mathbb{Z}_p^*$, and where $\ell_i(X)$ is the ith Lagrange basis polynomial associated to some set $\mathbb{H}$ of size n and $t(x)$ the corresponding vanishing polynomial. We assume $n \geq m+1$, and the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are of dimension $n \times (m+1)$. The language depends on the group key but we do not write this explicitly. The set $\mathbb{H}$ is also the set that is used to define the polynomials $\{u_j(x), v_j(x), w_j(x)\}_{j=0}^m$ in the setup of Groth16, which are the polynomials that interpolate the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ column-wise. For efficiency considerations, $\mathbb{H}$ is usually chosen to be a set of roots of unity.

The verification equation of our proof system is more complex than the original Groth16, so there are also more possibilities for the adversary to randomize the proof and our scheme falls short of proving membership in $\mathcal{L}^{\text{c-relaxed}}_{\mathsf{ck},\widetilde{\mathsf{ck}},\mathbf{A},\mathbf{B},\mathbf{C}}$ because $[w]_1, [e]_1$ may not be a valid commitment under, respectively, $\mathsf{ck}, \widetilde{\mathsf{ck}}$. In the honest case, for completeness, the prover will be proving membership in $\mathcal{L}^{\text{c-relaxed}}_{\mathsf{ck},\widetilde{\mathsf{ck}},\mathbf{A},\mathbf{B},\mathbf{C}}$, but for knowledge soundness we will argue that the adversary knows a witness for membership in the language:

$$\begin{aligned}\mathcal{L}^{\text{c-relaxed}}_{\mathsf{CK},\widetilde{\mathsf{CK}},\mathbf{A},\mathbf{B},\mathbf{C}} = \big\{&(u, \mathbf{x}, [e]_1, [w]_1) \in \mathbb{F} \times \mathbb{F}^l \times \mathbb{G}_1 \times \mathbb{G}_1 \mid \\ &\exists (\mathbf{w}, \mathbf{e}, \mathbf{R}_w, \mathbf{R}_e) \in \mathbb{F}^{m-l} \times \mathbb{F}^n \times \mathbb{F}^{D+4} \times \mathbb{F}^{D+4} \text{ s.t.} \\ &[w]_1 = \mathsf{Com}_{\mathsf{CK}}(\mathsf{pp}, \mathbf{w}; \mathbf{R}_w) \wedge [e]_1 = \mathsf{Com}_{\widetilde{\mathsf{CK}}}(\mathsf{pp}, \mathbf{e}; \mathbf{R}_e) \wedge \\ &((u, \mathbf{x}, \mathbf{e}), \mathbf{w}) \in \mathcal{R}^{\text{relaxed}}_{\mathbf{A},\mathbf{B},\mathbf{C}}\big\},\end{aligned}$$

where

$$\mathsf{CK} = (\mathsf{ck}, [\alpha]_1, [\delta]_1, \{[x^i]_1\}_{i=0}^D), \qquad \text{and} \qquad \widetilde{\mathsf{CK}} = (\widetilde{\mathsf{ck}}, [\alpha]_1, [\delta]_1, \{[x^i]_1\}_{i=0}^D),$$

for σ_j defined as specified in the structured reference string and where all trapdoors are uniformly chosen in $\mathbb{Z}_p^*$, and some bound D. The keys $\mathsf{CK}, \widetilde{\mathsf{CK}}$ can easily be proven to result in binding commitments, so we can still argue that given an adversary that computes a valid proof it is possible to extract a witness for the relaxed (non-committed) R1CS Relation.

Our scheme assumes that the choice of x, δ, ψ, ρ is done *before* the committed relaxed R1CS relation is defined, since the committed relation depends on $\mathsf{ck}, \widetilde{\mathsf{ck}}$ that depends on these variables. However, this choice means that the commitments $[w]_1, [e]_1$ are tailored to a specific circuit since the commitment keys are circuit-specific. In particular, for example, this means we cannot prove that this witness satisfies two different R1CS instances. We note that when $u = 1$ and $\mathbf{e} = \mathbf{0}$, the relaxed committed language is the language of commitments that open to a valid R1CS witness, which is quite similar in functionality to commit-and-prove (CaP) Groth16 as in LegoGroth16 [CFQ19]. However, the CaP formalism is quite different and has potentially stronger properties since it allows to prove many statements about a single commitment. This feature is not relevant in our application scenario, and this is why we do not follow the CaP formalism and we also find it acceptable to use circuit dependent keys.

We would like our new proof system to be combinable with the FLIP protocol. For this, we will assume that the relation generator also outputs auxiliary information $\mathsf{aux} = \left\{\{[y^i]_1, [y^i]_2\}_{i=1}^{k-1}\right\}$, which is given as input to the adversary, and show that this not compromise security. Also, looking ahead, we take into account how the structured reference string of our proof system would need to be generated in a setup ceremony. As in [BGM17], it would make sense to generate the setup in two phases, in the first phase sample $x, \alpha, \beta, \varphi$ and $\left\{\{[x^i]_1\}_{i=0}^{D}, \{[\alpha x^i]_1, [\beta x^i]_1, [\varphi x^i]_1\}_{i=0}^{n-1}\right\}$ and $\left\{\{[x^i]_2\}_{i=0}^{D}, [\beta]_2, [\varphi]_2\right\}$, for some $D \geq 2n-1$, and in the second phase sample δ, ρ, ψ and all related values. The value $D = 2n-1$ is enough to generate the setup, but we study security also when more powers of x are available. For this reason, although many of these values are not used by an honest prover and they should not necessarily be included in the SRS, these values are also given to the adversary.

The scheme is described in Fig. 4. The proof size is 3 group elements and non-interactive, as in the original Groth16 scheme, except now the statement includes two additional group elements $[w]_1, [e]_1$. Verification cost is 5 pairings, $l+1$ exponentiations in $\mathbb{G}_1$ and one exponentiation in $\mathbb{G}_T$. The latter can be substituted by 4 exponentiations and one product in $\mathbb{Z}_p$ by multiplying the verification equation by u^{-1}.

We first present the intuition for knowledge soundness. Groth16 is adding additional trapdoors to make sure the verifier checks simultaneously that:

- the public input is correct;
- linear relations: $[A]_1 = [\mathbf{Az}]_1, [B]_2 = [\mathbf{Bz}]_1, [C]_1 = [\mathbf{Cz}]_1$ (linear relations);
- *and* that $\mathbf{Az} \circ \mathbf{Bz} - \mathbf{Cz} = \mathbf{0}$ (Hadamard product relation).

The changes with respect to the standard Groth16 [Gro16] paper are: (a) the numerator of some terms involves an additional $\varphi \ell_j(x)$ term and (b) there are

Setup: The setup runs the relation generator that outputs a description of a R1CS instance R' associated to matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{Z}_p^{n \times m+1}$. Then it picks $x, \delta, \alpha, \beta, \psi, \varphi, \rho$ and it defines the committed R1CS relation with respect to the commitment keys:

$$\mathsf{ck} = ([\ell_{l+1}(x)/\rho]_1, \ldots, [\ell_m(x)/\rho]_1, [\delta/\rho]_1) \in \mathbb{G}_1^{m-l+1},$$

$$\widetilde{\mathsf{ck}} = ([\ell_0(x)/\psi]_1, \ldots, [\ell_{n-1}(x)/\psi]_1, [\delta/\psi]_1) \in \mathbb{G}_1^{n+1},$$

where $\ell_i(x)$ is the ith Lagrange basis polynomial of a set $\mathbb{H}$ of size n. Then it defines the polynomials $\boldsymbol{u}(x) = \boldsymbol{\ell}(x)^\top \mathbf{A}$, $\boldsymbol{v}(x) = \boldsymbol{\ell}(x)^\top \mathbf{B}$, and $\boldsymbol{w}(x) = \boldsymbol{\ell}(x)^\top \mathbf{C}$, and returns the srs defined as follows:

$$\mathsf{srs} := \left(\begin{array}{l} \Big[\mathsf{ck}, \widetilde{\mathsf{ck}}, \alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \{\sigma_j := u_j(x)\beta + v_j(x)\alpha + w_j(x)\}_{j=0}^{l} \\ \{x^i t(x)/\delta\}_{i=0}^{n-2}, \left\{\sigma_j := \dfrac{u_j(x)\beta + v_j(x)\alpha + w_j(x) + \varphi\ell_j(x)}{\delta}\right\}_{j=l+1}^{m} \Big]_1, \\ [\beta, \delta, [\varphi\rho]_2, [\psi]_2, \{x^i\}_{i=0}^{n-1}]_2, \ [\alpha\beta]_T \end{array} \right)$$

where

$$\mathsf{srs}_\mathsf{V} = \left(\{\sigma_j\}_{j=0}^{l}, [\delta]_2, [\varphi\rho]_2, [\psi]_2, [\alpha\beta]_T \right).$$

The setup also chooses $y \leftarrow \mathbb{Z}_p^*$ and outputs:

$$\mathsf{aux} = (\{[y^i]_1, [y^i]_2\}_{i=1}^{k}, [\varphi]_1, [\varphi]_2).$$

Prove, $\pi \leftarrow \mathsf{Prove}(\mathsf{srs}_\mathsf{P}, \phi = (\mathbf{z}_{[:l]}, u, [e]_1, [w]_1), W = (\mathbf{z}_{[l:]}, \mathbf{e}, r_w, r_e))$**:** assuming $z_0 = 1$, it acts as follows,

1. Let $A^\dagger(x) \leftarrow \sum_{j=0}^{m} z_j u_j(x)$, $B^\dagger(x) \leftarrow \sum_{j=0}^{m} z_j v_j(x)$, $C^\dagger(x) \leftarrow \sum_{j=0}^{m} z_j w_j(x)$,
2. Let $e^\dagger(x) = \sum_{i=0}^{n-1} e_i \ell_i(x)$. Set $h(x) = \sum_{i=0}^{n-2} h_i x^i \leftarrow (u^{-1} A^\dagger(x) B^\dagger(x) - C^\dagger(x) - e^\dagger(x) u^{-1})/t(x)$.
3. Set $[h(x)t(x)/\delta]_1 \leftarrow \sum_{i=0}^{n-2} h_i \left[x^i t(x)/\delta\right]_1$.
4. Set $r_a \leftarrow_r \mathbb{Z}_p$. Set $[A]_1 \leftarrow u^{-1} \sum_{j=0}^{m} z_j [u_j(x)]_1 + [\alpha]_1 + r_a [\delta]_1$.
5. Set $r_b \leftarrow_r \mathbb{Z}_p$. Set $[B]_2 \leftarrow [u\beta]_2 + \sum_{j=0}^{m} z_j [v_j(x)]_2 + r_b [\delta]_2$.
6. Set $[C]_1 \leftarrow r_b [A]_1 + r_a [B]_1 - r_a r_b [\delta]_1 + r_w[\varphi]_1 - u^{-1} r_e [1]_1 + \sum_{j=l+1}^{m} z_j [\sigma_j]_1 + [h(x)t(x)/\delta]_1$,
7. Return $\pi := ([A, C]_1, [B]_2)$.

Verify, $\{1, 0\} \leftarrow \mathsf{Verify}(\mathsf{srs}_\mathsf{V}, \phi, \pi = ([A, C]_1, [B]_2))$**:** assuming $z_0 = 1$, checks:

$$e([A]_1, [B]_2) - e([C]_1, [\delta]_2) - e(\sum_{j=0}^{l} z_j [\sigma_j]_1, [1]_2) - e(u^{-1} [e]_1, [\psi]_2) + e([w]_1, [\varphi\rho]_2) = u [\alpha\beta]_T.$$

Fig. 4: Groth16 for committed relaxed R1CS, where $\mathbf{z}_{[:l]} = (z_0, \ldots, z_l)$ and $\mathbf{z}_{[l:]} = (z_{l+1}, \ldots, z_m)$.

Simulate, $\pi \leftarrow \mathcal{S}(R, \phi, \mathsf{srs_V}, \mathsf{ts})$**:** Given the simulation trapdoors $\mathsf{ts} := (\beta, \delta, \psi, \varphi, \rho)$ and statement $\phi = (\mathbf{z}_{[:l]}, u, [e]_1, [w]_1)$ (1) chooses $A, B \leftarrow_r \mathbb{Z}_p$ (2) computes

$$[C]_1 = \frac{1}{\delta}\left([A \cdot B]_1 - \sum_{j=0}^{l} z_j [\sigma_j]_1 - u\beta [\alpha]_1 - \psi u^{-1} [e]_1 - \varphi\rho [w]_1\right)$$

Fig. 5: Simulator of Groth16 for committed relaxed R1CS.

additional terms in the original verification equation which involve $[\varphi\rho]_2, [\psi]_2$ and $[w]_1, [e]_1, u$.

Modification (a), applies the same technique used in Groth16 for proving linear relations, except the change makes sure the same witness for $[A]_1, [B]_2, [C]_1$ and also $[w]_1$ is used. Modification (b) accounts for the fact that now the verification equation is checking a more complex Hadamard relation that includes error terms, and the additional trapdoors are there to ensure that $[e]_1, [w]_1$ are in the right space.

We prove the following theorem:

Theorem 2. *Groth16 for relaxed committed R1CS in Fig. (4,5)has the following properties:*

- *Perfect Completeness: given* $(u, \mathbf{x}, [e]_1, [w]_1) \in \mathcal{L}^{c\text{-}relaxed}_{\mathsf{ck},\widetilde{\mathsf{ck}},\mathbf{A},\mathbf{B},\mathbf{C}}$*, the verifier will accept with probability* 1 *any proof computed by ah honest prover;*
- *Computational Soundness in the asymmetric bilinear group model: any adversary that computes a valid proof for* $(u, \mathbf{x}, [e]_1, [w]_1) \notin \mathcal{L}^{c\text{-}relaxed}_{\mathsf{CK},\widetilde{\mathsf{CK}},\mathbf{A},\mathbf{B},\mathbf{C}}$*, and receives as auxiliary input*

$$\mathsf{aux} = \left\{\{[y^i]_1, [y^i]_2\}_{i=0}^{k-1}, \{[x^i]_1, [x^i]_2\}_{i=n}^{D}, \{[\alpha x^i]_1, [\beta x^i]_1, [\varphi x^i]_1\}_{i=0}^{n-1}\right\},$$

for some $D \geq 2n-1$*, implies an adversary against the symmetric q-DLOG assumption with* $q = max(2n, D, k-1)$*.*

Proof (Theorem 2):

Completeness: It can easily be verified that all randomization terms (those involving r_a, r_b, r_e, r_w) are the same on both sides of the verification equation. For the rest of the proofs, we will ignore these terms for simplicity, i.e. we assume $r_a = r_b = r_e = r_w = 0$. Let $\boldsymbol{\ell}(\boldsymbol{x})^\top = (\ell_0(x), \ldots, \ell_{n-1}(x))$. Then,

$$(1)\ [w]_1 = \mathsf{Com}_{\mathsf{ck}}(\mathbf{z}_{[l:]}; 0) = \sum_{j=l+1}^{m} z_j \frac{\ell_j(x)}{\rho}, \quad (2)\ [e]_1 = \mathsf{Com}_{\widetilde{\mathsf{ck}}}(\mathbf{e}; 0) = \frac{1}{\psi}\boldsymbol{\ell}(\boldsymbol{x})^\top \mathbf{e}$$

$$(3)\ A^\dagger(x) = \boldsymbol{u}(\boldsymbol{x})^\top \mathbf{z} = \boldsymbol{\ell}(\boldsymbol{x})^\top(\mathbf{Az}), \qquad (4)\ B^\dagger(x) = \boldsymbol{v}(\boldsymbol{x})^\top \mathbf{z} = \boldsymbol{\ell}(\boldsymbol{x})^\top(\mathbf{Bz})$$

There exists a polynomial $h(x)$ of degree at most $n-2$ such that:

$$A^\dagger(x)B^\dagger(x) = (\boldsymbol{\ell}(\boldsymbol{x})^\top(\mathbf{Az})) \cdot (\boldsymbol{\ell}(\boldsymbol{x})^\top(\mathbf{Bz})) = \boldsymbol{\ell}(\boldsymbol{x})^\top((\mathbf{Az}) \circ (\mathbf{Bz})) + h(x)t(x),$$

since $A^\dagger(x)B^\dagger(x)$ agrees with $\boldsymbol{\ell(x)}^\top((\mathbf{Az})\circ(\mathbf{Bz}))$ on all of $\mathbb{H}$. On the other hand, the equation $\mathbf{Az}\circ\mathbf{Bz} = u\mathbf{Cz}+\mathbf{e}$ is equivalent to $u^{-1}\mathbf{Az}\circ\mathbf{Bz}-\mathbf{Cz}-u^{-1}\mathbf{e}=\mathbf{0}$. Therefore,

$$\begin{aligned} A^\dagger(x)B^\dagger(x) - C^\dagger(x) - e^\dagger(x)u^{-1} = \\ \boldsymbol{\ell(x)}^\top(u^{-1}\mathbf{Az}))\cdot(\boldsymbol{\ell(x)}^\top(\mathbf{Bz})) - \boldsymbol{\ell(x)}^\top(\mathbf{Cz}) - \boldsymbol{\ell(x)}^\top(u^{-1}\mathbf{e}) = \\ \boldsymbol{\ell(x)}^\top(u^{-1}\mathbf{Az}\circ\mathbf{Bz}-\mathbf{Cz}-u^{-1}\mathbf{e}) + h(x)t(x) \end{aligned}$$

so the polynomial $h(x)$ defined in Step 2 of the prover algorithm exists because the relaxed R1CS is satisfied. The verification is satisfied since we have,

$$e([A]_1,[B]_2) = u[\alpha\beta]_T + \left[\sum_{j=0}^{m} z_j(u_j(x)\beta + v_j(x)\alpha)\right]_T + u^{-1}[A^\dagger(x)B^\dagger(x)]_T, \quad (26)$$

and on the other hand,

$$\begin{aligned} &e([C]_1,[\delta]_2) + e(\left[\sum_{j=0}^{l} z_j(u_j(x)\beta + v_j(x)\alpha + w_j(x))\right]_1, [1]_2) = \\ &[C^\dagger(x)]_T + [h(x)t(x)]_T + [\sum_{j=0}^{m} z_j(u_j(x)\beta + v_j(x)\alpha)]_T + [\sum_{j=l+1}^{m} z_j\varphi\ell_j(x)]_T. \quad (27) \end{aligned}$$

Subtracting equations (26) and (27),

$$\begin{aligned} &e([A]_1,[B]_2) - e([C]_1,[\delta]_2) - e(\left[\sum_{j=0}^{l} z_j(u_j(x)\beta + v_j(x)\alpha + w_j(x))\right]_1, [1]_2) = \\ &u[\alpha\beta]_T + [u^{-1}A^\dagger(x)B^\dagger(x) - C^\dagger(x) - h(x)t(x)]_T + [\sum_{j=l+1}^{m} z_j\varphi\ell_j(x)]_T = \\ &u[\alpha\beta]_T + e(u^{-1}[e]_1,[\psi]_2) - e([w]_1,[\varphi\rho]_2), \end{aligned}$$

which is equivalent to the verification equation.

Knowledge Soundness. A significant part of the proof is taken almost verbatim from Groth16, accounting for the modifications, the auxiliary input that the adversary receives and some simplifications that follow from considering security only in the asymmetric bilinear group model. For simplicity, the vector of commitment keys ck is indexed in the natural way from $l+1$ to n.

Since the prover is assumed to be an algebraic adversary, there exist known field elements $A_\delta, A_{\sigma_j}, A_{\mathsf{ck}_j}, A_{\widetilde{\mathsf{ck}}_j}$ and polynomials $A(X)$ of degree D, $A_h(X)$ of degree $n-2$, $A_y(X)$ of degree $k-1$, and $A_\alpha(X), A_\beta(X), A_\varphi(X)$ of degree $n-1$

such that:

$$A = A_\alpha(x)\alpha + A_\beta(x)\beta + A_\varphi(x)\varphi + A_\delta\delta + \sum_{j=0}^{l} A_{\sigma_j}\sigma_j + \sum_{j=l+1}^{m} A_{\sigma_j}\sigma_j + \sum_{j=l+1}^{m+1} A_{\mathsf{ck}_j}\mathsf{ck}_j + \sum_{j=0}^{n} A_{\widetilde{\mathsf{ck}}_j}\widetilde{\mathsf{ck}}_j + A(x) + A_h(x)\frac{t(x)}{\delta} + A_y(y),$$

where σ_j are defined as explained in the protocol description, and $\mathsf{ck}_j, \widetilde{\mathsf{ck}}_j$ are the jth elements in the commitment keys (indexed, respectively, from $l+1$ to $m+1$ and from 0 to n).

We note that:

$$\sum_{j=0}^{l} A_{\sigma_j}\sigma_j = \alpha\sum_{j=0}^{l} A_{\sigma_j}v_j(x) + \beta\sum_{j=0}^{l} A_{\sigma_j}u_j(x) + \sum_{j=0}^{l} A_{\sigma_j}w_j(x) = \\ = \alpha A'_\alpha(x) + \beta A'_\beta(x) + A'(x),$$

for polynomials $A'_\beta(x), A'_\alpha(x), A'(x)$, of degree at most $n-1$. That is, any linear combination of σ_j, $j = 0, \ldots, l$, can be rewritten as a linear combination of other terms seen by the adversary, so from now on, we assume without loss of generality that $A_{\sigma_j} = 0$ for $j = 0, \ldots, l$.

In the knowledge soundness definition, the statement is chosen by the prover. In the AGM, the values $[w]_1, [e]_1$ have been computed by the prover as a linear combination of the values in the structured reference string or in the auxiliary string. Therefore, we can write out C, w, e in a similar fashion, and we call T_s, $s \in \{\delta, \sigma_j, \mathsf{ck}_j, \widetilde{\mathsf{ck}}_j\}$ the coefficients of term T, $T \in \{C, w, e\}$ in the corresponding variables, and similarly we call $T(X), T_y(X), T_h(X), T_\alpha(X), T_\beta(X), T_\varphi(X)$ the corresponding polynomials. Again, without loss of generality we will assume that $T_{\sigma_j} = 0$ for $j = 0, \ldots, l$.

On the other hand, B is in group $\mathbb{G}_2$, so it can be written as:

$$B = B_\beta\beta + B_\varphi\varphi + B_{\varphi\rho}\varphi\rho + B_\delta\delta + B(x) + B_y(y),$$

for field elements $B_\beta, B_\varphi, B_{\varphi\rho}, B_\delta$ and polynomials $B(X)$ of degree D and $B_y(x)$ of degree $k-1$.

The core of the proof consists in writing the verification equation as an equality of multivariate Laurent polynomials. The verification equation should hold when A, B, C, w, e are treated as formal polynomials in indeterminates $\alpha, \beta, \gamma, \varphi, \delta, x, y$. Else, using the same technique used in the proof of Groth16 due to Fuchsbauer et al. (Th. 7.2, [FKL18]) which consists of embedding a q-DLOG challenge in the trapdoor variables so that the challenge is information theoretically hidden, the adversary breaks a q-DLOG challenge (with $q = max(2n, D, k-1)$, which is the degree of the values seen by the adversary in the challenge variable).

(1) In the first part of the proof, Groth exploits the fact that the only quadratic term in the equation in which both elements are adversarially chosen is AB. This implies that

$$A_\beta(x) = 0.$$

The justification is that the terms with indeterminate β^2 are $A_\beta(x)B_\beta = 0$, which means $A_\beta(x) = 0$ or $B_\beta(x) = 0$. Since $[\alpha\beta]_1$ is not in the srs, and α is only given to the adversary in $\mathbb{G}_1$, the terms with indeterminate $\alpha\beta$ give us $A_\alpha(x)B_\beta = u$, so $A_\beta(x) = 0$. Additionally, without loss of generality, we can assume $A_\alpha(x) = 1$, $B_\beta = u$ (rescaling if necessary).

(2) Considering the terms involving $\frac{1}{\delta}$ in A, using the definition of σ_j, we have

$$\left(\sum_{j=l+1}^{m} A_{\sigma_j} \left(\beta u_j(x) + \alpha v_j(x) + w_j(x) + \varphi \ell_j(x)\right) + A_h(x)t(x) \right) \quad .$$

This term must be 0, since otherwise there is a term with β/δ in AB that cannot be canceled with any other term in the verification equation.

(3) The same reasoning as in Step (2) allows us to conclude that $A_{\mathsf{ck}_j}, A_{\widetilde{\mathsf{ck}}_j}$ are all zero, as otherwise there will be terms with β/ρ, β/ψ that would only appear in AB and in no other term of the verification equation.

(4) Since A is of the form $A = \alpha + ...$ and B of the form $B = u\beta + ...$, neither A nor B can have terms with y (with non-zero coefficients), i.e. $A_y(y) = 0$ and $B_y(y) = 0$, else there would be cross terms involving α and y or β and y that cannot be canceled otherwise. Similarly, C, w, e cannot have any y terms since there would be cross terms involving δ and y, $\varphi\rho$ and y, or ψ and y.

(5) We can also conclude that $A_\varphi(x) = B_\varphi = 0$, since if $A_\varphi(x) \neq 0$, then there is a non-zero term $\varphi\beta$ in AB, and if $B_\varphi \neq 0$, then there is a non-zero term $\varphi\alpha$ in AB, and none of them can appear anywhere else in the verification equation.

(6) As a result of all previous steps we conclude that:

$$A = \alpha + A(x) + A_\delta \delta$$

$$B = u\beta + B(x) + B_\delta \delta + B_\psi \psi + B_{\varphi\rho}\varphi\rho$$

(7) Since in the verification equation w is paired with $\varphi\rho$, we conclude that

$$w = w_\alpha \alpha + w_\delta \delta + w(x) + \sum_{j=l+1}^{m} w_{\mathsf{ck}_j} \mathsf{ck}_j.$$

This is because the rest of the terms $\{w_\beta(x), w_\varphi(x), \{w_{\sigma_j}\}_{j=l+1}^m, \{\widetilde{\mathsf{ck}}_j\}_{j=0}^n, w_h(x), w_y(y)\}$ must all be 0 since otherwise there would be terms with $\varphi\rho$ and other variables that cannot be canceled in the verification equation. On the other hand, the terms in the verification with $\alpha\varphi\rho$ are:

$$B_{\varphi\rho} + w_\alpha(x) = 0,$$

therefore the polynomial $w_\alpha(x)$ is actually a constant $w_\alpha = -B_{\varphi\rho} \in \mathbb{F}$. This shows that $[w]_1$ is a valid commitment to some vector $\mathbf{w}$ under the key CK and an opening can be extracted from the algebraic adversary. Define $\mathbf{z}_{[l:]} = (w_{\mathsf{ck}_{l+1}}, \ldots, w_{\mathsf{ck}_m})$.

(8) Similarly,

$$e = e_\alpha \alpha + e_\delta \delta + e(x) + \sum_{j=0}^{n} e_{\widetilde{\mathsf{ck}}_j} \widetilde{\mathsf{ck}}_j,$$

since the rest of the terms $\{e_\beta(x), e_\varphi(x), \{e_{\sigma_j}\}_{j=l+1}^m, \{\mathsf{ck}_j\}_{j=l+1}^m, e_h(x), e_y(y)\}$, and the only term with $\alpha\psi$ should be a constant (concretely, $u^{-1}e_\alpha = B_\psi$). This shows that $[e]_1$ is a valid commitment to some vector $\mathbf{e}$ under the key $\widetilde{\mathsf{CK}}$ and an opening can be extracted from the algebraic adversary. Define $\mathbf{e} = (e_{\widetilde{\mathsf{ck}}_0}, \ldots, e_{\widetilde{\mathsf{ck}}_{n-1}})$ as the extracted value.

(9) To complete the soundness we make the following argument. The terms that have only αx^i, and no other trapdoors are:

$$\alpha B(x) - \alpha \sum_{j=l+1}^{m} C_{\sigma_j} v_j(x) - \alpha \sum_{j=0}^{l} z_l v_j(x)$$

and the terms that have only βx^i, and no other trapdoors are:

$$u\beta A(x) - \beta \sum_{j=l+1}^{m} C_{\sigma_j} u_j(x) - \beta \sum_{j=0}^{l} z_j u_j(x)$$

and the terms that have only φx^i, and no other trapdoors are:

$$-\varphi \sum_{j=l+1}^{m} C_{\sigma_j} \ell_j(x) + \varphi \sum_{j=l+1}^{m} w_{\mathsf{ck}_j} \ell_j(x) = -\varphi \sum_{j=l+1}^{m} C_{\sigma_j} \ell_j(x) + \varphi \sum_{j=l+1}^{m} z_j \ell_j(x)$$

From the last equations, we conclude that $\mathbf{z}_{[l:]} = (C_{\sigma_{l+1}}, \ldots, C_{\sigma_n})$, and that

$$A(x) = u^{-1} \sum_{j=0}^{m} z_j u_j(x), \qquad B(x) = \sum_{j=0}^{m} z_j v_j(x).$$

The terms that depend only of x and no other trapdoor value are:

$$A(x)B(x) - \sum_{j=l+1}^{m} C_{\sigma_j} w_j(x) - \sum_{j=0}^{l} z_j w_j(x) - u^{-1} \sum_{j=0}^{n-1} e_{\widetilde{\mathsf{ck}}_j} \ell_j(x) - C_h(x)t(x)$$

Putting all facts together, we conclude that the last equation is equivalent to:

$$(u^{-1}\mathbf{Az}) \circ (\mathbf{Bz}) = \mathbf{Cz} + u^{-1}\mathbf{e}.$$

5 Conclusion

The FLIP protocol introduced in this paper, which does folding via inner pairing product, is a simpler way to use folding compared to doing recursive proof composition. Similar strategies can be used to fold other relations without cycles of elliptic curves.

The generalization of Groth16 is optimized for low communication complexity. Other alternatives with a simpler setup are possible. If, for example, $[e]_1, [w]_1$ are treated as polynomial commitments, extractability can be shown by opening in a random point and fewer trapdoors are necessary. Lower communication complexity might be achieved by using recent techniques such as Polymath [Lip24]. We leave for future work to explore this option (which requires additional changes in arithmetization to use some "square" span program) or to combine it with custom gates or lookups.

Our generalization can be useful in scenarios other than those envisaged in this paper. Specifically, one could use our proof system to prove the last step of recursive proof composition based on Nova, an interesting option to explore particularly as more constructions of embedded curves become available [SH23, Gui24] .

References

ABST22. Miguel Ambrona, Marc Beunardeau, Anne-Laure Schmitt, and Raphaël R. Toledo. aPlonK : Aggregated PlonK from multi-polynomial commitment schemes. Cryptology ePrint Archive, Report 2022/1352, 2022.

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, August 2010.

ark19. arkwork. Arkworks for zksnark programming, 2019. `https://github.com/arkworks-rs`.

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*, pages 315–334. IEEE Computer Society Press, May 2018.

BC23. Benedikt Bünz and Binyi Chen. Protostar: Generic efficient accumulation/folding for special sound protocols. Cryptology ePrint Archive, Paper 2023/620, 2023. `https://eprint.iacr.org/2023/620`.

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, May 2016.

BGH19. Sean Bowe, Jack Grigg, and Daira Hopwood. Halo: Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Paper 2019/1021, 2019. `https://eprint.iacr.org/2019/1021`.

BGM17. Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017.

BMM+21. Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. Proofs for inner pairing products and applications. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 65–97, December 2021.

CFQ19. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019.

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi 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, May 2020.

EG23. Liam Eagen and Ariel Gabizon. Protogalaxy: Efficient protostar-style folding of multiple instances. Cryptology ePrint Archive, Paper 2023/1106, 2023. `https://eprint.iacr.org/2023/1106`.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62, August 2018.

GMN22a. Nicolas Gailly, Mary Maller, and Anca Nitulescu. SnarkPack: Practical SNARK aggregation. LNCS, pages 203–229, 2022.

GMN22b. Nicolas Gailly, Mary Maller, and Anca Nitulescu. Snarkpack: Practical snark aggregation. In *Financial Cryptography, FC 2022*, page 203–229, Berlin, Heidelberg, 2022. Springer-Verlag.

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, May 2016.

Gui24. Aurore Guillevic. More embedded curves for SNARK-pairing-friendly curves. Cryptology ePrint Archive, Paper 2024/752, 2024.

HBHW21. Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification, 2021. `https://zips.z.cash/protocol/protocol.pdf`.

KP23. Abhiram Kothapalli and Bryan Parno. Algebraic reductions of knowledge. In *CRYPTO 2023, Part IV*, LNCS, pages 669–701, August 2023.

KS22. Abhiram Kothapalli and Srinath Setty. Supernova: Proving universal machine executions without universal circuits. Cryptology ePrint Archive, Paper 2022/1758, 2022. `https://eprint.iacr.org/2022/1758`.

KS23. Abhiram Kothapalli and Srinath Setty. Hypernova: Recursive arguments for customizable constraint systems. Cryptology ePrint Archive, Paper 2023/573, 2023. `https://eprint.iacr.org/2023/573`.

KST22. Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 359–388, August 2022.

Lab18. Protocol Labs. Filecoin, 2018. `https://filecoin.io/filecoin.pdf`.

Lee21. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 1–34, November 2021.

LGZX23. Xun Liu, Shang Gao, Tianyu Zheng, and Bin Xiao. Snarkfold: Efficient snark proof aggregation from split incrementally verifiable computation. 2023. https://eprint.iacr.org/2023/1946.

Lip24. Helger Lipmaa. Polymath: Groth16 is not the limit. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X*, volume 14929 of *Lecture Notes in Computer Science*, pages 170–206. Springer, 2024.

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.

LPS23. Helger Lipmaa, Roberto Parisella, and Janno Siim. Algebraic group model with oblivious sampling. LNCS, pages 363–392, November 2023.

MRV16. Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758, December 2016.

Set20. Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737, August 2020.

SH23. Antonio Sanso and Youssef El Housni. Families of prime-order endomorphism-equipped embedded curves on pairing-friendly curves. Cryptology ePrint Archive, Paper 2023/1662, 2023.