Design Specification of Universal Groth16 for Ethereum virtual machines

Jehyuk Jang (Onther Inc.)

last updated on Jan. 25, 2023

Table of contents

1 Introduction

A SNARK stands for a succinct non-interactive argument of knowledge and is an argument protocol for verifiable computation. There are at least two parties in the protocol that are a prover and a verifier; sometimes they are called workers and clients, respectively. A verifier outsources the execution of a computationally intensive program to a prover. After a prover completes running the program, he/she returns the result to a verifier with proof of its correctness. Without reproducing the result, a verifier runs a verification algorithm to discriminate the validity of the claimed result.

To make verifiable computation meaningful, the verification algorithm should be succinct, meaning that it should be computationally much simpler than reproducing the outsourced result. Formally, an argument protocol is said to be correct if a proof constructed from a valid input and output instance always passes verification. Inversely, an argument protocol is said to be sound if a proof constructed from an invalid instance would not pass verification (with high probability). Practical SNARK protocols typically leave a negligible uncertainty to failure in the soundness check, which enables the verification of a proof to be computationally much simpler than reproducing the outsourced result. This is how they can achieve succinctness.

Many SNARKs make use of discrete elliptic curves defined over a finite field to hide or encrypt information, since they satisfy three properties: 1) reversing the results is computationally intractable; 2) it is partially homomorphic; 3) there is a utility function called *pairing* to compensate the partial homomorphism. A hiding map (function) is said to be (partially) homomorphic if there is a way to express the hide of (some) arithmetic operations (e.g., addition and/or multiplication) on two inputs in terms of the respective hides of the two inputs. Homomorphism is important because the proof is a set of compressions and encryptions of information produced while running an outsourced program. To verify the proof, the arithmetic relation between each piece of information determined by the program should be preserved and can be checked even after being encrypted. The elliptic curve encryption is partially homomorphic to additions but not to multiplications. However, fortunately, bilinear functions called *pairing* for elliptic curve cryptography enable us to check the multiplicative relation between each piece of information. For this reason, SNARKs using elliptic curves are called *pairing-based SNARKs*.

Pairing-based SNARKs require a trusted third party besides a prover and a verifier. Using elliptic curves, a trusted third party generates public proving and verifying keys using his/her private keys (so-called toxic wastes) during a setup of SNARK. A set of proof passes verification only if it was generated with proving keys. In addition, a set of proof passes verification only if the components hold predefined arithmetic relations with verifying keys and an instance. By analogy, three keys are needed to pass verification, which are valid proofs, verification keys, and an instance. This triple security system prevents malicious provers from hacking the proving algorithm to generate a forgery-proof that deceives a verification algorithm. However, this mechanism works only when the third party is trustable. If a third party leaks the private keys, all the security safeguards listed above will be broken.

According to the scope of the trusted setup, pairing-based SNARKs can be divided into two subclasses: universal (general-purpose) SNARKs and circuit-specific SNARKs. Proving and verifying keys generated by the setup of circuit-specific SNARKs include variables representing an outsourced program and their arithmetic relation. In other words, it is a commitment to a program notarized by a trusted third party so that a prover and a verifier argue about the same program, and the commitment prevents a malicious prover from constructing a false proof with a forgery program and instance to deceive the verification algorithm. On the contrary, proving and verifying keys generated by the setup of universal SNARKs do not include the commitment to a program. Instead, the prover and verifier commit to a program by themselves between the setup and proving. That is, a trusted third party does not participate in the commitment. As the result, circuit-specific SNARKs should redo the setup for every new outsourced program, whereas universal SNARKs do not. Each time the setup is re-run, the trusted third party must spend a lot of time and effort convincing the prover and verifier of their trust.

Although circuit-specific SNARKs suffer from the problem of convincing reliability, they still have the best performance. It has been known that Groth16, which is one of the circuit-specific SNARKs, has the lowest prover and verifier complexity and shortest proof length among all pairing-based SNARKs. It is inevitable for universal SNARKs to require higher prover and verifier complexity than circuit-specific SNARKs due to the

commitment to a program conducted by a prover and verifier. Hence, universality and performance are a trade-off.

Our goal in this article is to find a compromise between the universality and performance of pairing-based SNARKs. We relax the definition of universal setup from supporting any (NP) program in the world to supporting Ethereum virtual machine (EVM) applications. EVM is a stack machine with a small instruction set of opcodes. Examples of EVM applications are transactions in the Ethereum blockchain, each just a permutation of the opcodes. With the relaxed definition of universal setup, we design a pairing-based SNARK and universal to EVM applications with better performance than existing strictly universal SNARKs.

We propose UniGroth16 as a solution to the problem. The main idea of UniGroth16 is in a derive algorithm to derive circuit-specific public keys from EVM-specific public keys. A trusted third party, through the setup, generates EVM-specific public keys containing commitments to all arithmetic instructions of an EVM system. Then, through a derive algorithm, anyone including not only a prover and verifier but also untrusted helpers derive circuit-specific public commitments to an EVM application. The derived commitments can be applied to Groth16's prove and verify algorithms. As a result, UniGroth16 has two advantages: 1) No trusted third parties are needed to rerun the setup whenever outsourcing a new EVM application. 2) The complexity of the prover and verifier inherits the complexity of Groth16, which is known to be the most succinct.

2 Notations and Terminologies

2.1 General notations

- $[k] := \{0, \cdots, k-1\}$ for a positive integer k.
- For an n-tuple $\mathbf{a}$, the length is denoted by $|\mathbf{a}|$ and the projection to the i-th element is denoted by $\mathbf{a}_i$.
- We treat a tuple as a row vector, and the concatenation of two vectors (tuples) $\mathbf{a}, \mathbf{b}$ is denoted by $(\mathbf{a}, \mathbf{b})$.
- The concatenation of vectors or matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ to form a matrix $\mathbf{E}$ is denoted by

$$\mathbf{E} = (\mathbf{A}, \mathbf{B}; \mathbf{C}, \mathbf{D}) = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}.$$

- For a function $f : X \to Y$, the image X under f and the preimage of $y \in Y$ under f are denoted respectively by $img(f)$ and $f^{-1}[y]$.
- For a function $f : \prod_{i=0}^{m-1} X_i \to \prod_{i=0}^{n-1} Y_i$ with sets X_i and Y_i, the projection of the k-th output of f with an input tuple x is denoted by $f_k(x)$.

2.2 Cryptographic notations

- Let $\mathbf{b} = (\mathbf{1}^\lambda, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ be a bilinear group parameters generated from a security parameter $\mathbf{1}^\lambda$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order p with generators $G \in \mathbb{G}_1$ and $H \in \mathbb{G}_2$, $[x]_A := xA$ for a field element x and a generator A, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerative bilinear map holding that $e([a]_G, [b]_H) = e([1]_G, [1]_H)^{ab}$ and that $e([1]_G, [1]_H)$ generates $\mathbb{G}_T$.

3 Constraint System

Let $\mathcal{R}$ be a relation generator that given a security parameter λ in unary returns a polynomial time decidable binary relation R_S and a linear map $\mathbf{\Omega}$. Let $\mathcal{R}_\lambda$ and $\mathcal{W}_\lambda$ respectively be the set of possible relations R_S and maps $\mathbf{\Omega}$ that the relation generator may output given 1^λ. Given a relation $R_S \subseteq \mathbb{F}^L \times \mathbb{F}^{M-L}$ of sizes M, L with

$M > L$ and a linear map $\mathbf{\Omega}: \mathbb{F}^s \to \mathbb{F}^L$ with a positive integer $s < L$, we define a derived relation $R_{\mathbf{\Omega}} = \left\{ (\mathbf{f}', \mathbf{w}) : (\mathbf{\Omega}(\mathbf{f}'), \mathbf{w}) \in R_S \right\}$.

In this section we construct the relation R_S for EVM systems and the relation $R_{\mathbf{\Omega}}$ for EVM applications.

3.1 Quadratic arithmetic programs

Consider an NP statement with instance $c_1, \cdots, c_l \in \mathbb{F}$ and witness $c_0 = 1, c_{l+1}, \cdots, c_{m-1} \in \mathbb{F}$ claimed. We can generate an arithmetic circuit to verify the NP statement in a polynomial time by checking whether the pair of instance and witness holds all constraints given in the circuit or not. In short, an arithmetic circuit gives us a relation set R consisting of the pairs of instance and witness.

We consider rank-1 constraint systems for an arithmetic circuits, which consists of n constraints of equations in the form of

$$\left(\sum_{i=0}^{m-1} c_i U_{i,k} \right) \cdot \left(\sum_{i=0}^{m-1} c_i V_{i,k} \right) = \left(\sum_{i=0}^{m-1} c_i W_{i,k} \right), \tag{1}$$

for $k = 0, \cdots, n-1$, where $U_{i,k}, V_{i,k}, W_{i,k} \in \mathbb{F}$ are called wire constants, which are determined by an NP statement. Sometimes the constraints are referred to as multiplication gates. In the k-th multiplication gate, we say that the i-th wire is either an input wire, if $U_{i,k} \neq 0$ or $V_{i,k} \neq 0$, or an output wire, if $W_{i,k} \neq 0$. We assume circuit convertible rank-1 constraint systems, where the wire constants are convertible to an arithmetic circuit: Each constraints has exactly two inputs and one output; Each multiplication input wire can be a sum of more than one wires so that addition gates in a circuit are not solely regarded as constraints but are counted as a part of inputs of a multiplication gate; The same wire cannot be shared by output wires of different multiplication gates; In a gate, there is no short , i.e., the same wire cannot be shared by input and output wires. We say witness is circuit satisfiable for an NP statement with witness if and only if all equations in (1) hold.

Assuming a large $|\mathbb{F}|$, we can reformulate the equation constraints in (1) into a quadratic arithmetic program. To this end, we first pick any n distinct $x_0, \cdots, x_{n-1} \in \mathbb{F}$ and define a root polynomial $t(X) = \prod_{k=0}^{n-1} (X - x_k)$. Then, we interpolate wire polynomials $u_i(X), v_i(X), w_i(X)$ of degree $n-1$ using the wire constants in (1) such that $u_i(x_k) = U_{i,k}, v_i(x_k) = V_{i,k}, w_i(x_k) = W_{i,k}$, for $i = 0, \cdots, m-1$ and $k = 0, \cdots, n-1$. As a result, we define a QAP

$$Q = \left\{ t(X), \{u_i(X)\}_{i=0}^{m-1}, \{v_i(X)\}_{i=0}^{m-1}, \{w_i(X)\}_{i=0}^{m-1} \right\}. \tag{2}$$

We can define a relation R given by a circuit in terms of QAP. Combining the elements of Q with the instance and witness in (1), we define a testing polynomial

$$p(X) = \left(\sum_{i=0}^{m-1} c_i u_i(X) \right) \cdot \left(\sum_{i=0}^{m-1} c_i v_i(X) \right) - \left(\sum_{i=0}^{m-1} c_i w_i(X) \right),$$

where we can see that, for all $k = 0, \cdots, n-1$, $p(x_k) = 0$ if equations in (1) hold. Conversely, if one or more equations in (1) does not hold, say it is the q-th equation, $p(x_q) \neq 0$ except for negligible probability by the Schwartz-Zippel lemma. This implies $p(X)$ is divisible by $t(X)$, i.e., there exist $h(X)$ such that $p(X) = t(X)h(X)$. As a result, the relation R of a circuit is defined as

$$R = \left\{ (\mathbf{f}, \mathbf{w}) \left| \begin{array}{l} \mathbf{f} = (c_1, \cdots, c_l), \mathbf{w} = (c_{l+1}, \cdots, c_{m-1}), \exists h(X) \\ \left[\left(\sum_{i=0}^{m-1} c_i u_i(X) \right) \cdot \left(\sum_{i=0}^{m-1} c_i v_i(X) \right) - \left(\sum_{i=0}^{m-1} c_i w_i(X) \right) = t(X)h(X) \right] \end{array} \right. \right\}.$$

3.2 Quadratic arithmetic programs for an EVM

We define an EVM as a triple of instruction set $\mathcal{S}$, instruction executor H, and the maximum length $s_{\max}$ of instructions that an EVM application can have. The instruction set we consider includes instructions for arithmetic, bitwise, logical, and hashing operations, but does not include the other instructions for the control of system flow or data communication. However, in a real-world EVM, the executor may communicate with a read-only memory or an external permanent storage to get input values, using the data communication instructions. Thus, to make our definition practical, our instruction set include a virtual instruction called LOAD, which loads public input data from a read-only memory or an external permanent storage.

The instruction set having s_D instructions is denoted by $\mathcal{S}=\left\{s^{(k)}\right\}_{k\in[s_D]}$. The instruction executor is denoted by $H(\cdot)$ that takes two inputs such as an instruction and a vector of input values and returns a vector of output values.

For an EVM=$(\mathcal{S},H,s_{\max})$, we define an EVM QAP $Q_{\mathcal{S}}$ to construct the relation $R_{\mathcal{S}}$. Let $Q_{\mathcal{S}}:=\bigcup_{k\in[s_D]}Q^{(k)}$, where $Q^{(k)}$ is the QAP to check the circuit satisfiability of witness $\mathbf{w}^{(k)}\in\mathbb{F}^{m^{(k)}-l^{(k)}}$ for solely executing the k-th instruction $\mathbf{f}_{out}^{(k)}=H\left(\mathbf{f}_{in}^{(k)},s^{(k)}\right)$ with the instance $\mathbf{f}^{(k)}=\left(\mathbf{f}_{out}^{(k)},\mathbf{f}_{in}^{(k)}\right)\in\mathbb{F}^{l^{(k)}}$. We refer each circuit represented by $Q^{(k)}$ to as the k-th subcircuit. Let $n^{(k)}$, $m^{(k)}$, and $l^{(k)}$ respectively denote the number of multiplication gates, total wires, and input and output wires in the k-th subcircuit. Let $\mathcal{I}_V^{(k)}$ denote the set of indices for input and output wires in the k-th instruction subcircuit. Let $\mathcal{I}_P^{(k)}$ denote the set of indices for interior wires in the k-th instruction subcircuit. It holds that $\mathcal{I}_V^{(k)}\bigcup\mathcal{I}_P^{(k)}=\left[m^{(k)}\right]$.

When building wire polynomials for $Q_{\mathcal{S}}$, in addition to the original structure given in (2), we impose an indeterminate Y and multiply powers of it to the original wire polynomials in X. Multiplying the powers of Y enables the wire polynomials to be linearly combined to derive a QAP of an EVM application F. To this end, we build two root polynomials $t_X(X)$ having arbitrary roots $\{x_k\}_{k\in[n]}$ with $n=\max_{k\in[s_D]}n^{(k)}$ and $t_Y(Y)$ having roots $\left\{y_k=\omega^k\right\}_{k\in[s_{\max}]}$, where $\omega\in\mathbb{F}$ satisfies $\omega^{s_{\max}}=1$ and $\sum_{k\in[s_{\max}]}\omega^k=0$. We construct $Q^{(k)}$ for each $k\in\left[s_D\right]$ as follow:

$$Q^{(k)}=\left\{\begin{array}{l} t_X(X),t_Y(Y),\\ \left\{\hat{u}_{i,j}^{(k)}(X,Y)=u_i^{(k)}(X)Y^{ks_{\max}+j}\right\}_{i\in\left[m^{(k)}\right],j\in[s_{\max}]},\\ \left\{\hat{v}_{i,j}^{(k)}(X,Y)=v_i^{(k)}(X)Y^{ks_{\max}+j}\right\}_{i\in\left[m^{(k)}\right],j\in[s_{\max}]},\\ \left\{\hat{w}_{i,j}^{(k)}(X,Y)=w_i^{(k)}(X)Y^{ks_{\max}+j}\right\}_{i\in\left[m^{(k)}\right],j\in[s_{\max}]} \end{array}\right\}, \tag{3}$$

The degrees of all wire polynomials $u_i^{(k)}$, $v_i^{(k)}$, $w_i^{(k)}$ are fixed to n so that all $Q^{(k)}$ share the same root polynomial $t_X(X)$. The factors $Y^{ks_{\max}}$ multiplied to each wire polynomial are to prevent two different $Q^{(k)}$ and $Q^{(k')}$ from sharing the same wire polynomial or, more generally, to prevent a wire polynomial of each subcircuit from being regenerated by linearly combining wire polynomials of the other subcircuits.

By linearly combining the wire polynomials in $Q_{\mathcal{S}}$, we can derive a test polynomial for verifying at most $s_{\max}$ subcircuits. For example, let $\mathbf{g}=\left(\mathbf{g}_0,\cdots,\mathbf{g}_{s_F-1}\right)$ with $s_F\le s_{\max}$ be the indices of the instructions ($\mathbf{g}_k<s_D$) to be verified for the executions with respective instance and witness. In other words, we derive a test polynomial for

the subcircuits satisfiability of witness $\mathbf{w}^{(k)}$ for instance $\mathbf{f}_{out}^{(k)} = H\left(\mathbf{f}_{in}^{(k)}, s^{(k)}\right)$ for each $k \in [s_F]$.

To this end, we linearly combine the wire polynomials in Q_S to derive a complete wire polynomial in the form of, for example, $u_i^{(\mathbf{g}_k)}(X)L_k^{(\mathbf{g}_k)}(Y)$, where $L_k^{(\mathbf{g}_k)}(Y)$ is the Lagrange basis such that $L_k^{(\mathbf{g}_k)}(\omega^k) = 1$ and $L_k^{(\mathbf{g}_k)}(\omega^{k'}) = 0$ for all other $k' \not\equiv k \pmod{s_{\max}}$, which can be constructed by

$$L_k^{(\mathbf{g}_k)}(Y) := s_{\max}{}^{-1} Y^{\mathbf{g}_k s_{\max}} \sum_{j=0}^{s_{\max}-1} \omega^{-kj} Y^j. \tag{4}$$

With the instruction sequence $\mathbf{g}$ given above, we can construct the test polynomial as follow:

$$\begin{aligned} p(X,Y) = &\left(\sum_{k=0}^{s_F-1} \sum_{i=0}^{m^{(\mathbf{g}_k)}-1} c_i^{(k)} u_i^{(\mathbf{g}_k)}(X) L_k^{(\mathbf{g}_k)}(Y) \right) \cdot \left(\sum_{k=0}^{s_F-1} \sum_{i=0}^{m^{(\mathbf{g}_k)}-1} c_i^{(k)} v_i^{(\mathbf{g}_k)}(X) L_k^{(\mathbf{g}_k)}(Y) \right) \\ &- \left(\sum_{k=0}^{s_F-1} \sum_{i=0}^{m^{(\mathbf{g}_k)}-1} c_i^{(k)} w_i^{(\mathbf{g}_k)}(X) L_k^{(\mathbf{g}_k)}(Y) \right), \end{aligned} \tag{5}$$

where $\left(\mathbf{f}^{(k)}, \mathbf{w}^{(k)}\right) = \left(c_i^{(k)}\right)_{i \in \left[m^{(k)}\right]}$. It holds that $p(X,Y) = 0$ at every pair of the roots in $\{x_i\}_{i\in[n]} \times \{\omega^k\}_{k \in [s_{\max}]}$, if and only if (with allowing negligible false positive probability) every witnesses $\mathbf{w}^{(k)}$ is satisfiable to the corresponding subcircuit $\mathbf{g}_k$.

To construct the relation R_S representing an EVM system only specified by Q_S, but not specified by an instruction sequence $\mathbf{g}$, we generalize the test polynomial in (5) for possible permutations of $\mathbf{g}$ as follow:

$$\begin{aligned} p(X,Y) = &\left(\sum_{k=0}^{s_{\max}-1} \sum_{g=0}^{s_D-1} \sum_{i=0}^{m^{(g)}-1} c_{k,i,g} L_k^{(g)}(Y) u_i^{(g)}(X) \right) \cdot \left(\sum_{k=0}^{s_{\max}-1} \sum_{g=0}^{s_D-1} \sum_{i=0}^{m^{(g)}-1} c_{k,i,g} L_k^{(g)}(Y) v_i^{(g)}(X) \right) \\ &- \left(\sum_{k=0}^{s_{\max}-1} \sum_{g=0}^{s_D-1} \sum_{i=0}^{m^{(g)}-1} c_{k,i,g} L_k^{(g)}(Y) w_i^{(g)}(X) \right). \end{aligned} \tag{6}$$

Unlike the circuit satisfiability discussed in the previous subsection, the coefficients $c_{k,i,g}$ in (6) include not only instance and witness but also the permutation of instructions. That is, both instruction permutation and witness are reflected into a target polynomial as inner products with wire polynomials. Each coefficient $c_{k,i,g}$ can be either an element of $\left(\mathbf{f}^{(k)}, \mathbf{w}^{(k)}\right)$ or an instruction selector in $\{0,1\}$. To sum up, it holds that $p(X,Y) = 0$ at every pair of the roots in $\{x_i\}_{i\in[n]} \times \{\omega^k\}_{k\in[s_{\max}]}$, if and only if the coefficients $c_{k,i,g}$ are satisfiable to an EVM system. This is equivalent to finding $h_X(X,Y)$ and $h_Y(X,Y)$ such that

$$p(X,Y) = h_X(X,Y)t_X(X) + h_Y(X,Y)t_Y(Y).$$

We now construct the relation R_S. Let $M_S = \sum_{g\in[s_D]} m^{(g)}$ and $L_S = \sum_{g\in[s_D]} l^{(g)}$. Let $\hat{\mathbf{u}}(X,Y), \hat{\mathbf{v}}(X,Y), \hat{\mathbf{w}}(X,Y) \in \mathbb{F}^{M_S s_{\max}}[X,Y]$ respectively be the row vectors of wire polynomials $\hat{u}_{i,j}^{(g)}(X,Y), \hat{v}_{i,j}^{(g)}(X,Y), \hat{w}_{i,j}^{(g)}(X,Y)$ for all $g \in [s_D], i \in \left[m^{(g)}\right], j \in [s_{\max}]$. The relation R_S is constructed by

$$R_{\mathcal{S}}=\left\{(\mathbf{f},\mathbf{w})\left|\begin{array}{l}\mathbf{f}\in\mathbb{F}^{L_{\mathcal{S}}s_{\max}},\mathbf{w}\in\mathbb{F}^{(M_{\mathcal{S}}-L_{\mathcal{S}})s_{\max}},\mathbf{c}=(\mathbf{f},\mathbf{w}),\\ \exists\left(h_X(X,Y),h_Y(X,Y)\right)\left[\begin{array}{l}\hat{\mathbf{cu}}^T(X,Y)\cdot\hat{\mathbf{cv}}^T(X,Y)-\hat{\mathbf{cw}}^T(X,Y)\\ =t_X(X)h_X(X,Y)+t_Y(Y)h_Y(X,Y)\end{array}\right]\end{array}\right\}. \tag{7}$$

3.3 Quadratic arithmetic programs for EVM applications

We define an EVM application $F:=(\mathbf{g},\rho)$ as a tuple of an instruction indices sequence $\mathbf{g}$ and a wire map ρ. An instruction index sequence allows repetitions and indicates the elements of $\mathcal{S}$ composing an EVM application. A wire map contains information about the communication of data used for or made by the execution of instructions in $\mathbf{g}$. The circuit for the execution of EVM application can be derived by combining the subcircuits of instructions in $\mathbf{g}$, where the input and output wires of subcircuits are connected to each other according to the wire map. Analogously, the QAP and relation for a EVM application can be derived by linearly combining the wire polynomials in $Q_{\mathcal{S}}$, according to $\mathbf{g}$ and ρ.

Given F, we define a wire map $\rho:\mathcal{P}_F\to\mathcal{P}_F$, where $\mathcal{P}_F:=\bigcup_{k=0}^{s_F}\left(\{k\}\times\left[m^{(\mathbf{g}_k)}\right]\right)$. Let $c_i^{(k)}$ denote the value assigned to the i-th wire in the subcircuit for the k-th instruction of $\mathbf{g}$. The wire map is constructed as follow:

$$\rho(k,i)=\begin{cases}(k',i'), & \text{if the } (k,i)\text{-th wire is an input wire and connected to the } (k',i')\text{-th wire,}\\ (k,i), & \text{if the } (k,i)\text{-th wire is an output or internal wire.}\end{cases} \tag{8}$$

Then, for all $(k,i)\in\mathcal{P}_F$,

$$c_i^{(k)}=c_{\rho_2(k,i)}^{(\rho_1(k,i))}. \tag{9}$$

As every subcircuit is always connected to at least one other subcircuit, a map ρ is non-surjective and non-injective. A set of indices to input wires fed by the (k,i)-th output wire is given by $\rho^{-1}\left[(k,i)\right]$. If the (k,i)-th wire is an output wire feeding the other wires, $\left|\rho^{-1}\left[(k,i)\right]\right|>1$. If the (k,i)-th wire is an internal wire, $\rho^{-1}\left[(k,i)\right]=\{(k,i)\}$. If the (k,i)-th wire is an input wire, $\rho^{-1}\left[(k,i)\right]$ is not defined.

Using the wire polynomials in $Q_{\mathcal{S}}$ constructed for an EVM system, we derive wire polynomials specific to the EVM application $F=(\mathbf{g},\rho)$. We first use $\mathbf{g}$ to form the Lagrange bases $L_k^{(\mathbf{g}_k)}(Y)$ as done in the previous subsection. Then, we reflect the connectivity between subcircuits into the polynomials. When two wires indexed by (k,i) and (k',i') are connected, it means $c_i^{(k)}=c_{i'}^{(k')}$, which implies that it holds, for example, $c_i^{(k)}u_i^{(\mathbf{g}_k)}(X)L_k^{(\mathbf{g}_k)}(Y)+c_{i'}^{(k')}u_{i'}^{(\mathbf{g}_{k'})}(X)L_{k'}^{(\mathbf{g}_{k'})}(Y)=c_i^{(k)}\left(u_i^{(\mathbf{g}_k)}(X)L_k^{(\mathbf{g}_k)}(Y)+u_{i'}^{(\mathbf{g}_{k'})}(X)L_{k'}^{(\mathbf{g}_{k'})}(Y)\right)$. For this reason, we add all wire polynomials indexed by the elements in $\rho^{-1}\left[(k,i)\right]$ for every $(k,i)\in img(\rho)$. As the result, we construct a test polynomial $p(X,Y)$ as follow:

$$\begin{aligned}p(X,Y)=&\left(\sum_{(k,i)\in img(\rho)}c_i^{(k)}\sum_{(k',i')\in\rho^{-1}[(k,i)]}L_{k'}^{(\mathbf{g}_{k'})}(Y)u_{i'}^{(\mathbf{g}_{k'})}(X)\right)\cdot\left(\sum_{(k,i)\in img(\rho)}c_i^{(k)}\sum_{(k',i')\in\rho^{-1}[(k,i)]}L_{k'}^{(\mathbf{g}_{k'})}(Y)v_{i'}^{(\mathbf{g}_{k'})}(X)\right)\\ &-\left(\sum_{(k,i)\in img(\rho)}c_i^{(k)}\sum_{(k',i')\in\rho^{-1}[(k,i)]}L_{k'}^{(\mathbf{g}_{k'})}(Y)w_{i'}^{(\mathbf{g}_{k'})}(X)\right).\end{aligned} \tag{10}$$

To derive the relation $R_{\mathbf{\Omega}}$ for F, we define a derive matrix $\mathbf{\Omega} \in \mathbb{F}^{l' \times L_S s_{\max}}$ so that the test polynomial in (10) equals

$$\begin{aligned} p(X,Y) = &\left(\mathbf{f}'\mathbf{\Omega}\hat{\mathbf{u}}_{\mathcal{I}_V}^T(X,Y) + \mathbf{w}\hat{\mathbf{u}}_{\mathcal{I}_P}^T(X,Y)\right) \cdot \left(\mathbf{f}'\mathbf{\Omega}\hat{\mathbf{v}}_{\mathcal{I}_V}^T(X,Y) + \mathbf{w}\hat{\mathbf{v}}_{\mathcal{I}_P}^T(X,Y)\right) \\ &- \left(\mathbf{f}'\mathbf{\Omega}\hat{\mathbf{w}}_{\mathcal{I}_V}^T(X,Y) + \mathbf{w}\hat{\mathbf{w}}_{\mathcal{I}_P}^T(X,Y)\right), \end{aligned} \tag{11}$$

where $\hat{\mathbf{u}}_{\mathcal{I}_V}, \hat{\mathbf{v}}_{\mathcal{I}_V}, \hat{\mathbf{w}}_{\mathcal{I}_V}$ are vectors of the wire polynomials $\hat{u}_i^k(X,Y), \hat{v}_i^k(X,Y), \hat{w}_i^k(X,Y)$ for all $k \in [s_D]$, indexed by the input-output wire indices $i \in \mathcal{I}_V^{(k)}$, and $\hat{\mathbf{u}}_{\mathcal{I}_P}, \hat{\mathbf{v}}_{\mathcal{I}_P}, \hat{\mathbf{w}}_{\mathcal{I}_P}$ are those indexed by the internal wire indices $i \in \mathcal{I}_P^{(k)}$, and $\mathbf{f}' \in \mathbb{F}^{l'}$ with a length l' and $\mathbf{w} \in \mathbb{F}^{(M_S - L_S)s_{\max}}$ are respectively instance and witness of the circuit.

The instance $\mathbf{f}'$ includes the public inputs to the circuit and the output of every subcircuit for the instantiation of the connection between subcircuits. Let l denote the length of public inputs loaded by LOAD instruction. We can observe that the instance length $l' = l + s_F - 1$ for the following reasons: All l public inputs are fed into the LOAD subcircuit as inputs; EVM instructions always have one output, and so do their subcircuits, except for the LOAD subcircuit; Every input wire in all $s_F - 1$ subcircuits except for LOAD is always fed (connected) by the output wire of the other subcircuit including LOAD. Thus, the instance $\mathbf{f}'$ is of length $l + s_F - 1$.

Meanwhile, as there is no connection of wires between internal wires, we remain the witness $\mathbf{w}$ and polynomials for them in (11) unchanged from (7).

We now construct the derive matrix $\mathbf{\Omega} \in \mathbb{F}^{l' \times L_S s_{\max}}$. The derive matrix is a Kronecker product $\otimes$ of Lagrange coefficients and a selector vector for instructions and wires. In other words, the mapping by $\mathbf{\Omega}$ first selects and picks some of polynomials from $\hat{\mathbf{u}}_{\mathcal{I}_V}, \hat{\mathbf{v}}_{\mathcal{I}_V}, \hat{\mathbf{w}}_{\mathcal{I}_V}$ according to the wire map ρ and then combines them with appropriate Lagrange coefficients. For formal construction, let $m = |img(\rho)|$ denote the number of wires in the circuit of an EVM application and $\mathbf{m}$ denote a vector of the circuit wire indices, defined as

$$\mathbf{m} := \left((k,t)\right)_{(k,t) \in img(\rho)}. \tag{12}$$

Let the Lagrange coefficients $\mathbf{\omega}^{(k')} = \left(\omega^{-k'j}\right)_{j \in [s_{\max}]}$ for $k' \in [s_F]$. Let the instruction and wire selectors $\mathbf{\theta}_V^{(i,k)} = \left(\theta_t^{(i)}\right)_{t \in \mathcal{I}_V^{(k)}}$ for $k \in [s_D]$, where

$$\theta_t^{(i)} = \begin{cases} 1, & if \ \exists (k',t') \in \rho^{-1}[\mathbf{m}_i] : t = t', \\ 0, & otherwise. \end{cases}$$

Then, for $i \in [l']$, the i-th row of $\mathbf{\Omega}$ equals the concatenation of vectors $\mathbf{v}^{(i,k)}$ for all $k \in [s_D]$, where

$$\mathbf{v}^{(i,k)} = \begin{cases} \mathbf{\theta}_V^{(i,\mathbf{s}_{k'})} \otimes \mathbf{\omega}^{(k')}, & if \ \exists (k',t') \in \rho^{-1}[\mathbf{m}_i] : k = \mathbf{s}_{k'}, \\ \mathbf{0}^{\left|\mathcal{I}_V^{(k)}\right| s_{\max}}, & otherwise, \end{cases}$$

and $\mathbf{0}^x$ for a positive integer x denotes a zero vector of length x.

As the result, the relation $R_{\mathbf{\Omega}}$ for an EVM application is derived from Q_S as follow:

$$R_{\mathbf{\Omega}} = \left\{ (\mathbf{f}', \mathbf{w}) \left| \begin{array}{l} \mathbf{f}' \in \mathbb{F}^{l'}, \mathbf{w} \in \mathbb{F}^{(M_S - L_S)s_{\max}}, \exists \left(h_X(X,Y), h_Y(X,Y)\right) \\ \left[\begin{array}{l} \left(\mathbf{f}'\mathbf{\Omega}\hat{\mathbf{u}}_{\mathcal{I}_V}^T(X,Y) + \mathbf{w}\hat{\mathbf{u}}_{\mathcal{I}_P}^T(X,Y)\right) \cdot \left(\mathbf{f}'\mathbf{\Omega}\hat{\mathbf{v}}_{\mathcal{I}_V}^T(X,Y) + \mathbf{w}\hat{\mathbf{v}}_{\mathcal{I}_P}^T(X,Y)\right) \\ - \left(\mathbf{f}'\mathbf{\Omega}\hat{\mathbf{w}}_{\mathcal{I}_V}^T(X,Y) + \mathbf{w}\hat{\mathbf{w}}_{\mathcal{I}_P}^T(X,Y)\right) = t_X(X)h_X(X,Y) + t_Y(Y)h_Y(X,Y) \end{array} \right] \end{array} \right. \right\}. \tag{13}$$

Also, it will be shown in 4.4 that it is computationally efficient to construct $\mathbf{\Omega}_P \in \mathbb{F}^{(m-l')\times(M_S-L_S)s_{\max}}$ in the same manner as $\mathbf{\Omega}$ but using $\mathcal{I}_P^{(k)}$ instead of $\mathcal{I}_V^{(k)}$. When $\mathbf{\Omega}_P$ is provided, the witness $\mathbf{w}$ can be replaced by a much shorter one $\mathbf{w}'$.

4 Universal SNARK for EVM applications

We define arguments of knowledge for $\mathcal{R}_\lambda$ and $\mathcal{W}$, which consist of a quintuple of polynomial time algorithms $(\text{Setup}, \text{Derive}_V, \text{Prove}, \text{Verify})$ defined as follows:

$\text{Setup}(R_S) \mapsto (\texttt{urs}, \boldsymbol{\tau})$ that probabilistically produces a universal reference string $\texttt{urs} = (\boldsymbol{\sigma}_G, \boldsymbol{\sigma}_H, \mathbf{z}_G, \mathbf{a}_G)$ of length k and a simulation trapdoor $\boldsymbol{\tau}$,

$\text{Derive}_V(R_{\mathbf{\Omega}}, \texttt{urs}) \mapsto \texttt{crs}_V^{(\mathbf{\Omega})}$ that deterministically produces a circuit-specific reference string $\texttt{crs}_V^{(\mathbf{\Omega})} = \mathbf{\Omega}\mathbf{z}_G^T$,

$\text{Prove}(R_{\mathbf{\Omega}}, \texttt{urs}, \mathbf{f}, \mathbf{w}) \mapsto \boldsymbol{\pi}$ that takes as input a pair of instance $\mathbf{f}$ and witness $\mathbf{w}$ and probabilistically outputs a proof $\boldsymbol{\pi}$ which is a linear combination of $\texttt{urs}$, i.e., there exists a proof generator matrix $\mathbf{\Pi} \in \mathbb{F}^{3\times k}$ such that $\boldsymbol{\pi} = \mathbf{\Pi}\texttt{urs}$,

$\text{Verify}(R_{\mathbf{\Omega}}, \texttt{urs}, \texttt{crs}_V^{(\mathbf{\Omega})}, \mathbf{f}, \boldsymbol{\pi}) \mapsto 0/1$ that deterministically returns either reject or accept the input instance $\mathbf{f}$ and proof $\boldsymbol{\pi}$.

4.1 Universal and Circuit-Specific Reference Strings

We use two reference strings; one is universal reference string $\texttt{urs}$ and the other is circuit-specific reference string $\texttt{crs}$. A universal reference string $\texttt{urs}$ contains hierarchy information such as toxic wastes that are randomized parameters and not allowed to be open to public, and commitments to the wire polynomials for an EVM system. The contents of $\texttt{urs}$ are independent to EVM applications, and therefore $\texttt{urs}$ is universal under an EVM system. The circuit-specific reference string $\texttt{crs}$ contains commitments to the wire polynomials for an EVM application F. Using a derive matrix $\mathbf{\Omega}$ based on F, all commitments in $\texttt{crs}_V^{(\mathbf{\Omega})}$ can be derived from the commitments in $\texttt{urs}$.

First, the universal reference string $\texttt{urs}$ contains commitments to wire polynomials of the subcircuit for every instruction in $\mathcal{S}$. Each commitment is made in a tied form of three polynomials for two inputs and one output to preserve the input-output consistency of a constraint, as done in [Groth16]. Formally, given secret keys $\alpha_u, \alpha_v, \gamma_z, \gamma_a$, we tie the wire polynomials as follow,

$$\begin{cases} z_i^{(k)}(X) := \left(\alpha_u u_i^{(k)}(X) + \alpha_v v_i^{(k)}(X) + w_i^{(k)}(X)\right)\gamma_z^{-1}, \\ a_i^{(k)}(X) := \left(\alpha_u u_i^{(k)}(X) + \alpha_v v_i^{(k)}(X) + w_i^{(k)}(X)\right)\gamma_a^{-1}. \end{cases} \tag{14}$$

Given a set $\mathcal{Q}_S$ sub-QAPs defined in and secret keys $\tau = (x, y, \alpha_u, \alpha_v, \gamma_a, \gamma_z)$, $\texttt{urs}$ with encryption generators G and H has the form of

$$\texttt{urs} = (\boldsymbol{\sigma}_G, \boldsymbol{\sigma}_H, \mathbf{z}_G, \mathbf{a}_G),$$

where

$$\begin{cases} \boldsymbol{\sigma}_G = \left(\begin{array}{l} \left[\alpha_u\right]_G, \left[\alpha_v\right]_G, \left[\gamma_a\right]_G, \left\{\left[x^i y^j\right]_G\right\}_{i=0,j=0}^{n-1, s_D s_{\max}-1}, \\ \left\{\left[x^i y^j t_X(x) \gamma_a^{-1}\right]_G\right\}_{i=0,j=0}^{n-2, 2s_D s_{\max}-2}, \left\{\left[x^i y^j t_Y(y) \gamma_a^{-1}\right]_G\right\}_{i=0,j=0}^{n-1, s_D s_{\max}-2} \end{array} \right), \\ \boldsymbol{\sigma}_H = \left(\left[\alpha_u\right]_H, \left[\gamma_z\right]_H, \left[\gamma_a\right]_H, \left\{\left[x^i y^j\right]_H\right\}_{i=0,j=0}^{n-1, s_D s_{\max}-1} \right), \\ \mathbf{z}_G = \left(\left[s_{\max}^{-1} z_i^{(k)}(x) y^{k s_{\max}+j} \right]_G \right)_{j \in [s_{\max}], i \in \mathcal{I}_V^{(k)}, k \in [s_D]}, \\ \mathbf{a}_G = \left(\left[s_{\max}^{-1} a_i^{(k)}(x) y^{k s_{\max}+j} \right]_G \right)_{j \in [s_{\max}], i \in \mathcal{I}_P^{(k)}, k \in [s_D]}. \end{cases}$$

Next, the circuit-specific reference string $\mathtt{crs}_V^{(\boldsymbol{\Omega})}$ contains the commitments to the connectivity between input and output wire polynomials of subcircuits defined over an EVM application $F = (\mathbf{g}, \rho)$. Given the derive matrix $\boldsymbol{\Omega}$ constructed by using $\mathbf{g}$ and ρ, we obtain $\mathtt{crs}_V^{(\boldsymbol{\Omega})} = \boldsymbol{\Omega} \mathbf{z}_G^T$.

Let $u_i(X,Y), v_i(X,Y), w_i(X,Y)$ for $i \in [l']$ denote the output wire polynomials for the circuit of F, which are constructed as follows:

$$\begin{cases} u_i(X,Y) = \sum_{(k',i') \in \rho^{-1}[\mathbf{m}_i]} L_{k'}^{(\mathbf{g}_{k'})}(Y) u_{i'}^{(\mathbf{g}_{k'})}(X), \\ v_i(X,Y) = \sum_{(k',i') \in \rho^{-1}[\mathbf{m}_i]} L_{k'}^{(\mathbf{g}_{k'})}(Y) v_{i'}^{(\mathbf{g}_{k'})}(X), \\ w_i(X,Y) = \sum_{(k',i') \in \rho^{-1}[\mathbf{m}_i]} L_{k'}^{(\mathbf{g}_{k'})}(Y) w_{i'}^{(\mathbf{g}_{k'})}(X), \end{cases}$$

where $\mathbf{m}$ is the list of wire indices defined in (12), and $L_k^{(g)}(Y)$ is the Lagrange basis defined in (4).

Formally, $\mathtt{crs}_V^{(\boldsymbol{\Omega})}$ has the form of

$$\mathtt{crs}_V^{(\boldsymbol{\Omega})} = \left(\left[s_{\max}^{-1} z_i(x,y) \right]_G \right)_{i \in [s_F]}, \tag{15}$$

where

$$z_i(X,Y) = \left(\alpha_u u_i(X,Y) + \alpha_v v_i(X,Y) + w_i(X,Y) \right) \gamma_z^{-1}. \tag{16}$$

4.2 SNARK Protocol

We construct a SNARK protocol for QAP satisfiability for $\mathcal{R}_\lambda$. The algorithms are constructed as follows:

$\text{Setup}\left(R_S\right) \mapsto \left(\texttt{urs}, \boldsymbol{\tau}\right)$:

$\boldsymbol{\tau} = \left(x, y, \alpha_u, \alpha_v, \gamma_a, \gamma_z\right) \xleftarrow{\$} \mathbb{F}^*$;

$z_i^{(k)}(X) \leftarrow \left(\alpha_u u_i^{(k)}(X) + \alpha_v v_i^{(k)}(X) + w_i^{(k)}(X)\right) s_{\max}{}^{-1} \gamma_z^{-1}$ for all $k \in [s_D]$, $i \in \mathcal{I}_V^{(k)}$;

$a_i^{(k)}(X) \leftarrow \left(\alpha_u u_i^{(k)}(X) + \alpha_v v_i^{(k)}(X) + w_i^{(k)}(X)\right) s_{\max}{}^{-1} \gamma_z^{-1}$ for all $k \in [s_D]$, $i \in \mathcal{I}_P^{(k)}$;

$$\boldsymbol{\sigma}_G \leftarrow \left(\begin{array}{l} [\alpha_u]_G, [\alpha_v]_G, [\gamma_a]_G, [\alpha_v x]_G, [\alpha_v y]_G, \left\{ \left[x^i y^j \right]_G \right\}_{i=0,j=0}^{n-1, s_D s_{\max}-1}, \\ \left\{ \left[x^i y^j t_X(x) \gamma_a^{-1} \right]_G \right\}_{i=0,j=0}^{n-2, 2 s_D s_{\max}-2}, \left\{ \left[x^i y^j t_Y(y) \gamma_a^{-1} \right]_G \right\}_{i=0,j=0}^{n-1, s_D s_{\max}-2} \end{array} \right);$$

$$\boldsymbol{\sigma}_H \leftarrow \left([\alpha_u]_H, [\gamma_z]_H, [\gamma_a]_H, [\alpha_u x]_H, [\alpha_u y]_H, \left\{ \left[x^i y^j \right]_H \right\}_{i=0,j=0}^{n-1, s_D s_{\max}-1} \right);$$

$$\mathbf{z}_G \leftarrow \left(\left[s_{\max}{}^{-1} z_i^{(k)}(x) y^{k s_{\max}+j} \right]_G \right)_{j \in [s_{\max}], i \in \mathcal{I}_V^{(k)}, k \in [s_D]};$$

$$\mathbf{a}_G \leftarrow \left(\left[s_{\max}{}^{-1} a_i^{(k)}(x) y^{k s_{\max}+j} \right]_G \right)_{j \in [s_{\max}], i \in \mathcal{I}_P^{(k)}, k \in [s_D]};$$

$\texttt{urs} \leftarrow \left(\boldsymbol{\sigma}_G, \boldsymbol{\sigma}_H, \mathbf{z}_G, \mathbf{a}_G\right)$;

return $\left(\texttt{urs}, \tau\right)$.

$\text{Derive}_V\left(R_{\boldsymbol{\Omega}}, \texttt{urs}\right) \mapsto \texttt{crs}_V^{(\boldsymbol{\Omega})}$

parse $\mathbf{z}_G = \left(\left[s_{\max}{}^{-1} z_i^{(k)}(x) y^{k s_{\max}+j} \right]_G \right)_{j \in [s_{\max}], i \in \mathcal{I}_V^{(k)}, k \in [s_D]} \leftarrow \texttt{urs}$;

$\texttt{crs}_V^{(\boldsymbol{\Omega})} \leftarrow \boldsymbol{\Omega} \mathbf{z}_G^T$;

return $\texttt{crs}_V^{(\boldsymbol{\Omega})}$.

$\text{Prove}\left(R_{\boldsymbol{\Omega}}, \texttt{urs}, \mathbf{f}, \mathbf{w}\right) \mapsto \boldsymbol{\pi}$

$\left(r, s\right) \xleftarrow{\$} \mathbb{F}^*$;

$\boldsymbol{\Lambda} \leftarrow \left(\boldsymbol{\Omega}, \mathbf{0}; \mathbf{0}, \boldsymbol{\Omega}_P\right)$;

$$\text{parse} \left\{ \begin{array}{l} [\alpha_u]_G, [\alpha_v]_G, [\gamma_a]_G, [\gamma_a]_H, [\gamma_z]_H, \\ \left(\left[x^i y^j t_X(x) \gamma_a^{-1} \right]_G \right)_{i=0,j=0}^{n-2, 2 s_D s_{\max}-2}, \left(\left[x^i y^j t_Y(y) \gamma_a^{-1} \right]_G \right)_{i=0,j=0}^{n-1, s_D s_{\max}-2}, \\ \mathbf{a}_G = \left(\left[s_{\max}{}^{-1} a_i^{(k)}(x) y^j \right]_G \right)_{j \in [s_{\max}], i \in \mathcal{I}_P^{(k)}, k \in [s_D]}, \\ \mathbf{o}_G = \left(\left[x^i y^j \right]_G \right)_{i \in [n], j \in [s_D s_{\max}]}, \mathbf{o}_H = \left(\left[x^i y^j \right]_H \right)_{i \in [n], j \in [s_D s_{\max}]} \end{array} \right\} \leftarrow \texttt{urs};$$

$[A]_G \leftarrow [\alpha_v]_G + \left(\mathbf{f}, \mathbf{w}\right) \boldsymbol{\Lambda} \mathbf{U} \mathbf{o}_G^T + r [\gamma_a]_G$;

$[B]_G \leftarrow [\alpha_u]_G + \left(\mathbf{f}, \mathbf{w}\right) \boldsymbol{\Lambda} \mathbf{V} \mathbf{o}_G^T + s [\gamma_a]_G$;

$[B]_H \leftarrow [\alpha_u]_H + \left(\mathbf{f}, \mathbf{w}\right) \boldsymbol{\Lambda} \mathbf{V} \mathbf{o}_H^T + s [\gamma_a]_H$;

$[C]_G \leftarrow \mathbf{w} \boldsymbol{\Omega}_P \mathbf{a}_G^T + s [A]_G + r [B]_G - rs [\gamma_a]_G + \left[h_X(x,y) t_X(x) \gamma_a{}^{-1} \right]_G + \left[h_Y(x,y) t_Y(y) \gamma_a{}^{-1} \right]_G$;

return $\pi \leftarrow \left([A]_G, [B]_H, [C]_G \right)$.

$\mathrm{Verify}\left(R_{\mathbf{\Omega}}, \mathtt{urs}, \mathtt{crs}_V^{(\mathbf{\Omega})}, \mathbf{f}, \boldsymbol{\pi}\right) \mapsto 0/1$

parse $\left\{c_i\right\}_{i\in[l']} \leftarrow \mathbf{f}$;

parse $\left([\alpha_v]_G, [\alpha_u]_H, [\gamma_z]_H, [\gamma_a]_H\right) \leftarrow \mathtt{urs}$;

parse $\left(\left[s_{\max}{}^{-1} z_i(x,y)\right]_G\right)_{i\in\mathcal{I}_V} \leftarrow \mathtt{crs}_V^{(\mathbf{\Omega})}$;

parse $\left([A]_G, [B]_H, [C]_G\right) \leftarrow \pi$;

$[D]_G \leftarrow \sum_{i\in[l']} c_i \left[z_i(x,y)\right]_G$;

return 1, if $e\left([A]_G, [B]_H\right) = e\left([\alpha_v]_G, [\alpha_u]_H\right) e\left([D]_G, [\gamma_z]_H\right) e\left([C]_G, [\gamma_a]_H\right)$, return 0, otherwise.

4.3 Security under trusted Derive

In this subsection, we show that if Derive is trusted by Verify, the protocol constructed in subsection 4.2 holds the security properties below. In other words, for every given $R_S \in \mathcal{R}_\lambda$ and $\mathbf{\Omega} \in \mathcal{W}_\lambda$, we assume there is no possibility that the circuit-specific reference string used by Verify is derived by $\mathbf{\Omega}'$ other than $\mathbf{\Omega}$. In the next subsection, we will remove this assumption.

We follow the definitions of the security properties given in [Gro16] with modifications to make them compatible with our protocol.

Definition 2. An argument of knowledge $(\mathrm{Setup}, \mathrm{Derive}_V, \mathrm{Prove}, \mathrm{Verify}, \mathrm{Simulate})$ for $\mathcal{R}_\lambda$ is a non-interactive zero-knowledge if it has perfect completeness, perfect zero-knowledge and statistical knowledge soundness against affine prover strategies defined as follows:

Perfect completeness. The argument is perfect complete if a proof generated from a valid $(\mathbf{f}, \mathbf{w})$ is always accepted, i.e., for all $\lambda \in \mathbb{N}$, $R_S \in \mathcal{R}_\lambda$, $\mathbf{\Omega} \in \mathcal{W}_\lambda$, $(\mathbf{f}, \mathbf{w}) \in R_{\mathbf{\Omega}}$,

$$\Pr\left[\begin{array}{l}\left(\mathtt{urs}, \boldsymbol{\tau}\right) \leftarrow \mathrm{Setup}\left(R_S\right); \mathtt{crs}_V^{(\mathbf{\Omega})} \leftarrow \mathrm{Derive}_V\left(R_{\mathbf{\Omega}}, \mathtt{urs}\right); \boldsymbol{\pi} \leftarrow \mathrm{Prove}\left(R_{\mathbf{\Omega}}, \mathtt{urs}, \mathbf{f}, \mathbf{w}\right): \\ \mathrm{Verify}\left(R_{\mathbf{\Omega}}, \mathtt{urs}, \mathtt{crs}_V^{(\mathbf{\Omega})}, \mathbf{f}, \boldsymbol{\pi}\right) = 1\end{array}\right] = 1 \qquad (17)$$

Perfect zero-knowledge. The argument is zero-knowledge if a proof does not reveal any information besides the truth of the statement. Formally, the argument is perfect zero-knowledge, if, given $R_{\mathbf{\Omega}}$ and $\mathbf{f}$, any information about an acceptable $\boldsymbol{\pi}$ cannot be obtained by observing another acceptable proof $\boldsymbol{\pi}'$ simulated by the following algorithm:

$\mathrm{Simulate}\left(R_{\mathbf{\Omega}}, \boldsymbol{\tau}, \mathbf{f}\right) \mapsto \boldsymbol{\pi}' = \left([A]_G, [B]_H, [C]_G\right)$ that probabilistically outputs a simulated proof $\boldsymbol{\pi}'$ that, when inputted to Verify with $\mathbf{f} = \left(c_i\right)_{i\in\mathcal{I}_V}$, causes it to return 1: $(r,s) \overset{\$}{\leftarrow} \mathbb{F}^*$, $[A]_G = [r]_G$, $[B]_H = [s]_H$, and

$$[C]_G = \left[\frac{rs - \alpha_v\alpha_u - \sum_{i\in[l']} c_i\left(\alpha_u u_i(x,y) + \alpha_v v_i(x,y) + w_i(x,y)\right)}{\gamma_a}\right]_G.$$

In other words, for all $\lambda \in \mathbb{N}$, $R_S \in \mathcal{R}_\lambda$, $\mathbf{\Omega} \in \mathcal{W}_\lambda$, $(\mathbf{f}, \mathbf{w}) \in R_{\mathbf{\Omega}}$, the probability distributions of π and π' are identical.

Statistical knowledge soundness against affine prover strategies. Let $\mathcal{A}$ be an adversary that generates a pair

$(\mathbf{f},\mathbf{\Pi})$ of instance and proof generator matrix given $\mathtt{crs}_V^{(\mathbf{\Omega})}$. Suppose there exists a polynomial time extractor $\mathcal{X}_{\mathcal{A}}$ as an oracle machine that extracts a witness $\mathbf{w}$ from $\mathbf{\Pi}$. The argument is knowledge soundness against affine prover strategies if the probability that the extracted witness for every acceptable $(\mathbf{f},\boldsymbol{\pi})$ with $\boldsymbol{\pi}=\mathbf{\Pi}\mathtt{urs}$ generated by $\mathcal{A}$ is not in $R_{\mathbf{\Omega}}$ is negligible in λ, i.e., for all $R_{\mathcal{S}}\in\mathcal{R}_\lambda$, $\mathbf{\Omega}\in\mathcal{W}_\lambda$,

$$\Pr\left[\begin{array}{l}(\mathtt{urs},\tau)\leftarrow\mathrm{Setup}(\mathcal{R}_\lambda);\mathtt{crs}_V^{(\mathbf{\Omega})}\leftarrow\mathrm{Derive}_V(R_{\mathbf{\Omega}},\mathtt{urs},\mathbf{\Omega});\\ (\mathbf{f},\mathbf{\Pi})\leftarrow\mathcal{A}(R_{\mathbf{\Omega}},\mathtt{urs});\mathbf{w}\leftarrow\mathcal{X}_{\mathcal{A}}(R_{\mathbf{\Omega}},\mathbf{f},\mathbf{\Pi}):\\ \mathbf{\Pi}\in\mathbb{F}^{3\times k}\wedge\mathrm{Verify}\left(R_{\Omega},\mathtt{crs}_V^{(\mathbf{\Omega})},\mathbf{f},\mathbf{\Pi}\mathtt{urs}\right)=1\wedge(\mathbf{f},\mathbf{w})\notin R_{\mathbf{\Omega}}\end{array}\right]\approx 0. \tag{18}$$

Theorem 3. *The argument of knowledge* $(\mathrm{Setup},\mathrm{Derive},\mathrm{Prove},\mathrm{Verify},\mathrm{Simulate})$ *for* $\mathcal{R}_\lambda$ *constructed in subsection 4.2 is a non-interactive zero-knowledge argument with perfect completeness and perfect zero-knowledge. It also has statistical knowledge soundness against affine prover strategies in generic group model.*

Proof. It is straightforward to check the perfect completeness of protocol. To check the perfect zero-knowledge of protocol, we observe the probability distributions of proof π generated by Prove and simulated proof π'. In both proofs, the distributions of $[A]_G$ and $[B]_H$ are respectively identical to the uniform distributions of r and s, and given r,s, $[C]_G$ are deterministic. Thus, the protocol is perfect zero-knowledge.

To check the statistical knowledge soundness against affine prover strategies, we show that with high probability the valid witness can be extracted from an acceptable proof like that done for Theorem 2 in [Gro16].

In generic group model, the adversary generates proofs $\mathbf{\Pi}\mathtt{urs}=\left([A]_G,[B]_H,[C]_G\right)$ with Laurent polynomials A,B,C in indeterminates $x,y,\alpha_u,\alpha_v,\gamma_a,\gamma_z$ as follows:

$$\begin{aligned}A=&A_{\alpha_u}\alpha_u+A_{\alpha_v}\alpha_v+A_{\alpha_v x}\alpha_v x+A_{\alpha_v y}\alpha_v y+A_{\gamma_a}\gamma_a+A_{x,y}(x,y)\\&+\sum_{k\in[s_D]}\sum_{i\in\mathcal{I}_V^{(k)}}\sum_{j\in[s_{\max}]}A_{k,i,j}^{(z)}\frac{\alpha_u u_i^{(k)}(x)+\alpha_v v_i^{(k)}(x)+w_i^{(k)}(x)}{\gamma_z}y^{j+ks_{\max}}\\&+\sum_{k\in[s_D]}\sum_{i\in\mathcal{I}_P^{(k)}}\sum_{j\in[s_{\max}]}A_{k,i,j}^{(a)}\frac{\alpha_u u_i^{(k)}(x)+\alpha_v v_i^{(k)}(x)+w_i^{(k)}(x)}{\gamma_a}y^{j+ks_{\max}}\\&+\frac{A_{H^{(1)}}(x,y)t^{(1)}(x)}{\gamma_a}+\frac{A_{H^{(2)}}(x,y)t^{(2)}(y)}{\gamma_a},\end{aligned} \tag{19}$$

$$B=B_{\alpha_u}\alpha_u+B_{\alpha_u x}\alpha_u x+B_{\alpha_u y}\alpha_u y+B_{\gamma_z}\gamma_z+B_{\gamma_a}\gamma_a+B_{x,y}(x,y), \tag{20}$$

and the polynomial C can be constructed in the same manner with A, where $A_{x,y}(x,y)$, $B_{x,y}(x,y)$, and $C_{x,y}(x,y)$ have degree at most $(n-1,s_D s_{\max}-1)$, and $A_{H^{(1)}}(x,y)$, $A_{H^{(2)}}(x,y)$, $C_{H^{(1)}}(x,y)$, and $C_{H^{(2)}}(x,y)$ have degree at most $(n-2,2s_D s_{\max}-2)$ and $(n-1,s_D s_{\max}-2)$, respectively. Given $\mathtt{crs}_V^{(\mathbf{\Omega})}$, Verify combines A,B,C to check if the following verification polynomial V for $R_{\mathbf{\Omega}}$ equals 0:

$$V:=AB-\alpha_u\alpha_v-\sum_{i\in[s_F]}c_i\left(\alpha_u u_i(x,y)+\alpha_v v_i(x,y)+w_i(x,y)\right)-C\gamma_a. \tag{21}$$

If the coefficients of A,B,C determined are not related to $R_{\mathbf{\Omega}}$, in other words, if the verification polynomial V is not a zero polynomial, by the Schwartz-Zippel lemma, the probability that the evaluation is $V=0$ is negligible.

What remains to show is that given a zero polynomial V we can extract the valid witness. In other words, we show that V is a zero polynomial only if (with allowing negligible false positive probability) A, B, C is made of w such that $(\mathbf{f}, \mathbf{w}) \in R_{\Omega}$.

There should be no terms in α_u^2, which implies $A_{\alpha_u} = 0$ or $B_{\alpha_u} = 0$. Since the coefficient $B_{\alpha_u} A_{\alpha_v}$ of the term in $\alpha_u \alpha_v$ should equal 1, we have $A_{\alpha_u} = 0$. In addition, without loss of generality, we assume to rescale $A_{\alpha_v} = B_{\alpha_u} = 1$, which implies $B_{\gamma_z} = 0$, since there is no term in $\alpha_v \gamma_z$.

There should be no terms in $\alpha_u \alpha_v x^2$ and $\alpha_u \alpha_v y^2$, which respectively imply $A_{\alpha_v x} B_{\alpha_u x} = 0$ and $A_{\alpha_v y} B_{\alpha_u y} = 0$. That is, we can consider 5 possible combinations of zeros and non-zeros in $\{A_{\alpha_v x}, B_{\alpha_u x}, A_{\alpha_v y}, B_{\alpha_u y}\}$. First, assuming $A_{\alpha_v x} \neq 0$ and $B_{\alpha_u y} \neq 0$ implies the coefficient $A_{\alpha_v y} B_{\alpha_u x} + A_{\alpha_v x} B_{\alpha_u y}$ of term $\alpha_u \alpha_v xy$ was nonzero, which contradicts the zero polynomial F. Second, assuming $A_{\alpha_v y} \neq 0$ and $B_{\alpha_u x} \neq 0$ also implies a nonzero term in $\alpha_u \alpha_v xy$. Third, assuming $A_{\alpha_v x} \neq 0$ and $A_{\alpha_v y} \neq 0$ along with the fact that A_{α_v} and B_{α_u} are nonzero implies that the terms in $\alpha_u \alpha_v x$ and $\alpha_u \alpha_v y$ respectively having coefficients $A_{\alpha_v x} B_{\alpha_u} + A_{\alpha_v} B_{\alpha_u x}$ and $A_{\alpha_v y} B_{\alpha_u} + A_{\alpha_v} B_{\alpha_u y}$ were nonzero. Fourth, assuming $B_{\alpha_u x} \neq 0$ and $B_{\alpha_u y} \neq 0$ also implies that the terms in $\alpha_u \alpha_v x$ and $\alpha_u \alpha_v y$ were nonzero for the same reason. Thus, fifth and the last, we have $A_{\alpha_v x} = B_{\alpha_u x} = A_{\alpha_v y} = B_{\alpha_u y} = 0$, which are necessary for the zero polynomial F.

For all $i \in \mathcal{I}_p^{(k)}, j \in [s_{\max}]$, there should be no terms involving $\alpha_u \gamma_z^{-1} x^i y^j, \alpha_u \gamma_a^{-1} x^i y^j$, which implies $A_{k,i,j}^{(z)} = A_{k,i,j}^{(a)} = 0$ for all $k \in [s_D]$, given $B_{\alpha_u} = 1$, and $A_{H^{(1)}}(x, y) = A_{H^{(2)}}(x, y) = 0$.

Subsequently, the adversary polynomial A and B can be simplified as follows:

$$A = \alpha_v + A_{\gamma_a} \gamma_a + A_{x,y}(x, y), \tag{22}$$

$$B = \alpha_u + B_{\gamma_a} \gamma_a + B_{x,y}(x, y). \tag{23}$$

The verification equation in (21) holds only when the terms involving $\alpha_v x^i y^j$ and $\alpha_u x^i y^j$ for $i \in \mathcal{I}_p^{(k)}, j \in [s_{\max}]$ are respectively given

$$\alpha_v B_{x,y}(x, y) = \sum_{i \in [l']} c_i \alpha_v v_i(x, y) + \sum_{k \in [s_D]} \sum_{i \in \mathcal{I}_p^{(k)}} \sum_{j \in [s_{\max}]} C_{k,i,j}^{(a)} \alpha_v v_i^{(k)}(x) y^{j + k s_{\max}}, \tag{24}$$

$$\alpha_u A_{x,y}(x, y) = \sum_{i \in [l']} c_i \alpha_u u_i(x, y) + \sum_{k \in [s_D]} \sum_{i \in \mathcal{I}_p^{(k)}} \sum_{j \in [s_{\max}]} C_{k,i,j}^{(a)} \alpha_u u_i^{(k)}(x) y^{j + k s_{\max}}. \tag{25}$$

Given (24) and (25), the terms involving $x^i y^j$ in (21) are given

$$\begin{aligned} &\left(\sum_{i \in [l']} c_i u_i(x, y) + \sum_{k \in [s_D]} \sum_{i \in \mathcal{I}_p^{(k)}} \sum_{j \in [s_{\max}]} C_{k,i,j}^{(a)} u_i^{(k)}(x) y^{j + k s_{\max}} \right) \cdot \left(\sum_{i \in [l']} c_i v_i(x, y) + \sum_{k \in [s_D]} \sum_{i \in \mathcal{I}_p^{(k)}} \sum_{j \in [s_{\max}]} C_{k,i,j}^{(a)} v_i^{(k)}(x) y^{j + k s_{\max}} \right) \\ &- \left(\sum_{i \in [l']} c_i w_i(x, y) + \sum_{k \in [s_D]} \sum_{i \in \mathcal{I}_p^{(k)}} \sum_{j \in [s_{\max}]} C_{k,i,j}^{(a)} w_i^{(k)}(x) y^{j + k s_{\max}} \right) - C_{H^{(1)}}(x, y) t^{(1)}(x) - C_{H^{(2)}}(x, y) t^{(2)}(y). \end{aligned} \tag{26}$$

We let $\mathbf{f}$ denote a row vector of c_i for all $i \in [l']$ and $\mathbf{w}$ denote a row vector of $C_{k,i,j}^{(a)}$ for all

$k \in [s_D], i \in \mathcal{I}_p^{(k)}, j \in [s_{\max}]$. By the QAP in $R_{\boldsymbol{\Omega}}$, the terms in (26) equals zero only if $(\mathbf{f}, \mathbf{w}) \in R_{\boldsymbol{\Omega}}$. □

4.4 Efficient Verifier and Prover with Helped Derive

The protocol given in 4.2 holds the security properties under the assumption of trusted Derive. This assumption can only be realized in practice only when Derive_V is run by verifiers themselves. To reduce verifier complexity, we use a polynomial commitment scheme based on [Plonk] and [Kate], extending it from a commitment to a batch of univariate polynomials to a commitment to a batch of bivariate polynomials. With the commitment, any untrustworthy helpers can run Derive_V to provide a circuit-specific reference string $\texttt{crs}_V^{(\boldsymbol{\Omega})}$ such that with high probability verifier can be convinced that it is derived by $\boldsymbol{\Omega}$.

We construct a commitment scheme $\text{CM}_{\text{Derive}}$ for deriver and verifier as follow:

$\text{CM}_{\text{Derive}_V}\left(R_{\boldsymbol{\Omega}}, \texttt{urs}, \texttt{crs}_V^{(\boldsymbol{\Omega})}, \boldsymbol{\Omega}\right)$

1. Deriver sends $\texttt{crs}_V^{(\boldsymbol{\Omega})} = \left(\left[s_{\max}^{-1} z_i(x,y)\right]_G\right)_{i \in [l']}$.
2. Verifier sends random $\beta, \psi, \zeta \xleftarrow{\$} \mathbb{F}$.
3. Deriver and Verifier compute
$u_V(X,Y) \leftarrow \sum_{i \in [l']} \beta^i u_i(X,Y);\ v_V(X,Y) \leftarrow \sum_{i \in [l']} \beta^i v_i(X,Y);\ w_V(X,Y) \leftarrow \sum_{i \in [l']} \beta^i w_i(X,Y).$
4. Deriver computes the polynomials $H_1(X,Y), H_2(X,Y), H_3(X,Y), H_4(Y), H_5(Y), H_6(Y)$ such that
$$\begin{cases} u_V(X,Y) - u_V(\psi,\zeta) = H_1(X,Y)(X-\psi) + H_2(Y)(Y-\zeta), \\ v_V(X,Y) - v_V(\psi,\zeta) = H_3(X,Y)(X-\psi) + H_4(Y)(Y-\zeta), \\ w_V(X,Y) - w_V(\psi,\zeta) = H_5(X,Y)(X-\psi) + H_6(Y)(Y-\zeta). \end{cases}$$
5. Deriver sends the proof
$\pi_{\text{cm}} \leftarrow \left(\left[H_1(x,y)\right]_G, \left[H_2(y)\right]_G, \left[H_3(x,y)\right]_H, \left[H_4(y)\right]_H, \left[H_5(x,y)\right]_G, \left[H_6(y)\right]_G\right).$
6. Verifier checks if the equation holds:
$$e\left(\sum_{i \in [l']} \beta^i \left[z_i(x,y)\right]_G, [\gamma_z]_H\right) = e\left(u_V(\psi,\zeta)[\alpha_u]_G + v_V(\psi,\zeta)[\alpha_v]_G + w_V(\psi,\zeta)[1]_G, [1]_H\right)$$
$$e\left(\left[H_1(x,y)\right]_G, [\alpha_u x]_H - \psi[\alpha_u]_H\right) e\left(\left[H_2(y)\right]_G, [\alpha_u y]_H - \zeta[\alpha_u]_H\right)$$
$$e\left([\alpha_v x]_G - \psi[\alpha_v]_G, \left[H_3(x,y)\right]_H\right) e\left([\alpha_v y]_G - \zeta[\alpha_u]_G, \left[H_4(y)\right]_H\right)$$
$$e\left(\left[H_5(x,y)\right]_G, [x]_H - [\psi]_H\right) e\left(\left[H_6(y)\right]_G, [y]_H - [\zeta]_H\right).$$

A prover can also get helps from untrustworthy derivers. Helped provers can utilize Verify to check whether the derived results are correct or not, so unlike helped verifiers, they do not need to run a commitment scheme. We let denote $\mathbf{U}, \mathbf{V}, \mathbf{W}$ respectively be the matrices of monomial coefficients for the polynomial vectors $\hat{\mathbf{u}}(X,Y), \hat{\mathbf{v}}(X,Y), \hat{\mathbf{w}}(X,Y)$. We split the computations in Prove into two parts, Derive_P for helping prover and Prove_{helped} for helped prover, as follows:

$\text{Derive}_P\left(R_{\boldsymbol{\Omega}}, \texttt{urs}\right) \mapsto \texttt{crs}_P^{(\boldsymbol{\Omega})}$

$\boldsymbol{\Lambda} \leftarrow (\boldsymbol{\Omega}, \mathbf{0}; \mathbf{0}, \boldsymbol{\Omega}_P)$;

$$\text{parse}\begin{cases}\mathbf{a}_G=\left(\left[s_{\max}{}^{-1}a_i^{(k)}(x)y^j\right]_G\right)_{j\in[s_{\max}],i\in\mathcal{I}_P^{(k)},k\in[s_D]}\\ \mathbf{o}_G=\left(\left[x^iy^j\right]_G\right)_{i=0,j=0}^{n-1,s_Ds_{\max}-1}\\ \mathbf{o}_H=\left(\left[x^iy^j\right]_H\right)_{i=0,j=0}^{n-1,s_Ds_{\max}-1}\end{cases}\leftarrow \texttt{urs};$$

$$\left(\left[a_i(x,y)\right]_G\right)_{i\in[m]-[l']}\leftarrow\mathbf{\Omega}_P\mathbf{a}_G^T;$$

$$\left(\left[u_i(x,y)\right]_G\right)_{i\in[m]}\leftarrow\mathbf{\Lambda U o}_G^T;$$

$$\left(\left[v_i(x,y)\right]_G\right)_{i\in[m]}\leftarrow\mathbf{\Lambda V o}_G^T;$$

$$\left(\left[v_i(x,y)\right]_H\right)_{i\in[m]}\leftarrow\mathbf{\Lambda V o}_H^T;$$

$$\text{return } \texttt{crs}_P^{(\mathbf{\Omega})}\leftarrow\left(\left\{\left[a_i(x,y)\right]_G\right\}_{i\in[m]-[l']},\left\{\left[u_i(x,y)\right]_G\right\}_{i\in[m]},\left\{\left[v_i(x,y)\right]_G\right\}_{i\in[m]},\left\{\left[v_i(x,y)\right]_H\right\}_{i\in[m]}\right).$$

$$\text{Prove}_{helped}\left(R_{\mathbf{\Omega}_V},\texttt{urs},\texttt{crs}_P^{(\mathbf{\Omega})},\mathbf{f},\mathbf{w},\mathbf{\Omega}_V\right)\mapsto\pi$$

$$(r,s)\xleftarrow{\$}\mathbb{F}^*;$$

$$\text{parse}\begin{cases}\left[\alpha_u\right]_G,\left[\alpha_v\right]_G,\left[\gamma_a\right]_G,\left[\gamma_a\right]_H,\left[\gamma_z\right]_H,\\ \left(\left[x^iy^jt_X(x)\gamma_a^{-1}\right]_G\right)_{i=0,j=0}^{n-2,2s_Ds_{\max}-2},\left(\left[x^iy^jt_Y(y)\gamma_a^{-1}\right]_G\right)_{i=0,j=0}^{n-1,s_Ds_{\max}-2}\end{cases}\leftarrow\texttt{urs};$$

$$\text{parse}\begin{cases}\mathbf{a}_G=\left(\left[a_i(x,y)\right]_G\right)_{i\in[m]-[l']},\\ \mathbf{u}_G=\left(\left[u_i(x,y)\right]_G\right)_{i\in[m]},\\ \mathbf{v}_G=\left(\left[v_i(x,y)\right]_G\right)_{i\in[m]},\\ \mathbf{v}_H=\left(\left[v_i(x,y)\right]_H\right)_{i\in[m]}\end{cases}\leftarrow\texttt{crs}_P^{(\mathbf{\Omega})};$$

$$\left[A\right]_G\leftarrow\left[\alpha_v\right]_G+(\mathbf{f},\mathbf{w})\mathbf{u}_G^T+r\left[\gamma_a\right]_G;$$

$$\left[B\right]_G\leftarrow\left[\alpha_u\right]_G+(\mathbf{f},\mathbf{w})\mathbf{v}_G^T+s\left[\gamma_a\right]_G;$$

$$\left[B\right]_H\leftarrow\left[\alpha_u\right]_H+(\mathbf{f},\mathbf{w})\mathbf{v}_H^T+s\left[\gamma_a\right]_H;$$

$$\left[C\right]_G\leftarrow\mathbf{w}\mathbf{a}_G^T+s\left[A\right]_G+r\left[B\right]_G-rs\left[\gamma_a\right]_G+\left[h_X(x,y)t_X(x)\gamma_a{}^{-1}\right]_G+\left[h_Y(x,y)t_Y(y)\gamma_a{}^{-1}\right]_G;$$

$$\text{return } \pi\leftarrow\left(\left[A\right]_G,\left[B\right]_H,\left[C\right]_G\right).$$

As a result, Prover and Verifier have reduced computation complexity. Let M_F denote the number of wires selected by the instruction and wire selectors from Q_S. Similarly, let L_F denote the number of public wires selected. Let $\mathbf{\Omega}^*\in\mathbb{F}^{l'\times L_Fs_{\max}}$ and $\mathbf{\Omega}_P^*\in\mathbb{F}^{(m-l')\times(M_F-L_F)s_{\max}}$ respectively denote simpler representations of $\mathbf{\Omega}\in\mathbb{F}^{l'\times L_Ss_{\max}}$ and $\mathbf{\Omega}_P\in\mathbb{F}^{(m-l')\times(M_S-L_S)s_{\max}}$ removing zero columns. With $\text{CM}_{\text{Derive}}$, a verifier reduces the number of G_1 exponentiations from $L_Fs_{\max}$ to $2l'$, at the cost of the increased proof size by 4 G_1 elements and 2 G_2 elements and computing 8 more pairings for the verification of commitment proofs. Similarly, with helped Prove, given $\mathbf{\Omega}_P^*\in\mathbb{F}^{(m-l')\times(M_F-L_F)s_{\max}}$, a prover reduces the number of G_1 exponentiations from $5N+(M_F-L_F)s_{\max}$ to $3N+3m-l'+L_Fs_{\max}$, at the cost of computing 3 more pairings. We summarize the size and complexity of our SNARK protocol and compares those with PlonK [plonk] in Table 1.

Table 1. Comparison on the size and complexity of SNARK protocols, where $s_{\max}$ and s are respectively the maximum and actual lengths of an instruction sequence $\mathbf{s}$, $N = ns_{\max}$ is the maximum number of multiplication gates in a circuit, and A is the number of addition gates in a circuit (for PlonK), n_G is the number of multiplication gates in a circuit (for Groth16), M_F and L_F are the total number of wires and I/O wires, respectively, in the subcircuits uniquely indexed by $\mathbf{g}$, m is the number of wires in a circuit.

		Proposed (unhelped)	**Proposed (helped)**	PlonK	Groth16 (circuit-specific)
Circuit proof length ($\mathbb{G}_1, \mathbb{G}_2, \mathbb{F}$)		0	(4, 2, 0)	(9, 0, 8)	0
Witness proof length ($\mathbb{G}_1, \mathbb{G}_2, \mathbb{F}$)		(2, 1, 0)	(2, 1, 0)		(2, 1, 0)
Instance length ($\mathbb{F}$)		l'	l'	l	l
Prover complexity	FFT	1 Mult + 1 Div (order N)	1 Mult + 1 Div (order N)	12 Mults + 6 Divs (order $N + A$)	1 Mult+ 1 Div (order n_G)
	G1 exp	$5N + (M_F - L_F)s_{\max}$	$3N + 3m - s$	$9(N + A)$	$3n_G + m - l$
	G2 exp	N	m	-	n_G
Verifier complexity	G1 exp	$L_F s_{\max}$	$2l'$	18	l
	Pairing	3	11	2	3
Deriver complexity	FFT	-	3 Divs (order N)	-	-
	G1 exp	-	$(m - s)s_{\max} + L_F s_{\max} + 2N$	$7(N + A)$	-
	G2 exp	-	N	-	-