Simulation-Extractable zk-SNARK with a Single Verification

Jihye Kim[1], Jiwon Lee[2], and Hyunok Oh[2]

[1] Kookmin University, Seoul, Korea,
`jihyek@kookmin.ac.kr`
[2] Hanyang University, Seoul, Korea,
`{jiwonlee,hoh}@hanyang.ac.kr`

Abstract. The simulation-extractable zk-SNARK (SE-SNARK) introduces a security notion of non-malleability. The existing pairing-based zk-SNARKs designed from linear encoding are known to be vulnerable to algebraic manipulation of the proof. The latest SE-SNARKs check the proof consistency by increasing the proof size and the verification cost. In particular, the number of pairings increases almost doubles due to further verification.

In this paper, we propose two novel SE-SNARK constructions with a single verification. The consistency check is subsumed in a single verification through employing a hash function. The proof size and verification time of the proposed SE-SNARK schemes are minimal in that it is the same as the state-of-the-art zk-SNARK without non-malleability.

The proof in our SE-SNARK constructions comprises only three group elements (type III) in the QAP-based scheme and two group elements (type I) in the SAP-based scheme. The verification time in both requires only 3 pairings. The soundness of the proposed schemes is proven under the hash-algebraic knowledge (HAK) assumption and the (linear) collision-resistant hash assumption.

Keywords: pairing-based zk-SNARK, simulation-extractability, quadratic arithmetic program, square arithmetic program

1 Introduction

The zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK) is an effective zero-knowledge proof system to prove a statement without revealing the witness, where the proof size and the verification cost are succinct. In particular, the pairing-based zk-SNARKs [Gro16, PHGR13] are well-known for their constant-sized proof and constant-time verification, which make them a suitable choice for various applications including blockchain [BCG+14, DFKP13]. Especially, the Groth's protocol [Gro16] is accepted as a current standard for pairing-based SNARKs, which has a minimal proof size of 3 group elements and requires 3 pairings in verification.

One main concern in the pairing-based zk-SNARKs is that the proofs are vulnerable to the algebraic manipulation; since the proof elements possess an algebraic structure of linear encoding, it is possible to create a new proof from arbitrary proofs without knowing the witness. For instance, in Groth's protocol [Gro16] where the simplified version of the proof consists of three elements (G^a, H^b, G^c) satisfying $a \cdot b = c$, an adversary can forge a new proof by using a random r while preserving the algebraic relation as $(G^{ar}, H^{br^{-1}}, G^c)$ or (G^a, H^{b+r}, G^{c+ar}).

In order to prevent the malleability, Groth and Maller [GM17] introduced a *simulation-extractability*, a security notion for non-malleability of proofs. They defined a simulation-extractable zk-SNARK (SE-SNARK), and proposed a construction based on the Groth's zk-SNARK [Gro16] to maintain the proof size as 3 group elements. However, their construction relies on the representation of square arithmetic program (SAP), instead of quadratic arithmetic program (QAP) as in common zk-SNARKs; compared to the QAP, the SAP roughly *doubles* the circuit size which leads to doubling the common reference string (CRS) size and proving time. In short, Groth and Maller construction [GM17] sacrifices the CRS size and proving time to gain simulation-extractability.

To avoid this inefficiency, Bowe and Gabizon [BG18] restored the QAP representation in the SE-SNARK by applying an elliptic curve hashing [Ica09] to the Groth's protocol [Gro16]. However, they had to pay the price of proof size as 5 elements; 2 additional elements are required to check the consistency of hashed elements. The proof size can be a crucial cost for size-sensitive blockchains such as Zcash [BCG$^+$14] where each transaction requires a proof.

Most recently, Lipmaa [Lip19] improved the result further, by proposing an SE-SNARK for QAP which has a proof size of 4 elements. His construction adds a special tag and a trapdoor for the simulation-extractability, and compresses them into a single additional argument which cannot be algebraically manipulated without the knowledge of witnesses. The result of 4 elements is close enough to the 3 elements in Groth's protocol [Gro16], but it is still paying the price of one additional proof element.

Another crucial price is that all the SE-SNARKs, including the schemes above [GM17, BG18, Lip19], require an additional check in the verification. In the linear nature of pairing-based zk-SNARKs, the original check for the relation (i.e. $a \cdot b = c$ in QAP or $a^2 = c$ in SAP) is unable to detect algebraic modifications. It is formally proved in [GM17] that SNARKs from linear encoding require at least 2 verifications to be simulation-extractable, which is reduced to the hard-decisional NP problem. Hence, the SE-SNARK verifications suffer from additional pairings; [GM17], [BG18], and [Lip19] all require 2 additional pairings along with the original relation check which consists of 3 pairings. It almost doubles the cost of the verification, which is not desirable for applications where verification occurs frequently in the resource-limited clients.

In this paper, we propose SE-SNARKs with a *single verification*, by applying the hash function to overcome the boundaries of existing SE-SNARKs. The idea is from the fact that blending the hash function into the encodings can provide a unique connection between proof elements; it eliminates the requirement for additional check for algebraic modifications. In [BG18] which also applies a hash function, an additional verification is still required since the hash output is an independent element which should be checked afterwards. On the other hand, if we combine the hash output into the encoding itself (i.e. secret exponents), the additional check is unnecessary since proof elements are already determined as a unique tuple. Specifically, in the simplified proof (G^a, H^b, G^c) of Groth's protocol [Gro16], let c include the hash values of each input G^a and H^b; then G^c is determined as a unique element tightly connected to G^a and H^b. In this case, when G^a or H^b is (algebraically) modified, c should be also modified accordingly to satisfy the original relation[3,4].

We construct two versions of SE-SNARK: a QAP-based construction and an SAP-based construction, both with a single verification which reduces the verification time from 5 pairings to to 3 pairings compared to the existing SE-SNARKs [GM17, BG18, Lip19]. Our QAP-based construction achieves a proof size of 3 elements, which does not require any additional element as in [BG18, Lip19] or sacrifice CRS size as in [GM17]. Our SAP-based construction achieves a proof size of 2 elements, which surpasses the proof boundary of 3 elements in [GM17]. Both of our constructions accomplish simulation-extractability with a *minimal* proof size and verification time among the existing SE-SNARKs [GM17, BG18, Lip19]. The security of our SE-SNARKs is based on the hash-algebraic knowledge (HAK) assumption from [Lip19] and the existence of the (linear) collision-resistant hash function; the SAP-based scheme requires a collision-resistant function (CR), while the QAP-based scheme requires a linear collision-resistant hash function (LCR), a variant of the collision-resistant hash function. Both CR and LCR can be implemented from the standard hash function such as SHA2. In the security viewpoint, the existence of LCR is at least weaker than the discrete log assumption in the random oracle model. The complete version of the constructions is more complicated than the intuition and described in section 5.

Table 1 compares the size and computation performance of SNARKs, including Groth's zk-SNARK [Gro16] (without simulation-extractability) and various SE-SNARKs. Our QAP-based SE-SNARK achieves 3 proof elements (type III); it does not sacrifice any price for simulation-extractability from Groth's protocol [Gro16]. Also, our SAP-based SE-SNARK achieves 2 proof elements (type I), which is more efficient (one less proof element and two less pairings in verification) than Groth and Maller's SAP-based SE-SNARK [GM17].

The rest of this paper proceeds as follows. Section 2 organizes related works on zk-SNARKs. Section 3 introduces some preliminary backgrounds, and sec-

[3] Since the hash is applied before the encoding, we can adopt any standard hash (e.g. SHA-256) unlike [BG18] which requires a hash function to map an input into an elliptic curve.

[4] Notice that the boundary of 2 verifications from [GM17] is not applicable to our construction; the hash output in c prevents the construction from being included in SNARKs from linear encodings.

Table 1: The comparison of SE-SNARKs, based on arithmetic circuit satisfiability with l element instances, m wires, and n multiplication gates. Since SAP uses squaring gates, $2n$ squaring gates and $2m$ wires are considered instead of n multiplication gates and m wires; Units: $\mathbb{G}$ stands for group elements, E stands for exponentiations and P stands for pairings.

	Circuit	$\lvert CRS \rvert$	$\lvert \pi \rvert$	**P time**	**V time**	**Eqs.**	**Security**
[Gro16]	QAP	$(m+2n)\mathbb{G}_1 + n\mathbb{G}_2$	$2\mathbb{G}_1 + \mathbb{G}_2$	$(m+3n)E_1 + nE_2$	$lE_1 + 3P$	1	GGM
[GM17]	SAP	$(2m+4n)\mathbb{G}_1 + 2n\mathbb{G}_2$	$2\mathbb{G}_1 + \mathbb{G}_2$	$(2m+4n)E_1 + 2nE_2$	$lE_1 + 5P$	2	XPKE
[BG18]	QAP	$(m+5n)\mathbb{G}_1 + n\mathbb{G}_2$	$3\mathbb{G}_1 + 2\mathbb{G}_2$	$(m+3n)E_1 + nE_2$	$lE_1 + 5P$	2	ROM
[Lip19]	QAP	$(m+3n)\mathbb{G}_1 + n\mathbb{G}_2$	$3\mathbb{G}_1 + \mathbb{G}_2$	$(m+4n)E_1 + nE_2$	$lE_1 + 5P$	2	HAK
Ours	QAP	$(m+3n)\mathbb{G}_1 + n\mathbb{G}_2$	$2\mathbb{G}_1 + \mathbb{G}_2$	$(m+4n)E_1 + nE_2$	$lE_1 + 3P$	1	HAK,LCR
Ours	SAP	$(2m+6n)\mathbb{G}$	$2\mathbb{G}$	$(2m+6n)E$	$lE + 3P$	1	HAK,CR

tion 4 introduces security assumptions. In section 5, we propose a QAP-based SE-SNARK construction. In section 6, we propose an SAP-based SE-SNARK construction. In section 7, we conclude.

2 Related Work

In the history of proof systems and verifiable computations, there are various NIZK arguments with different types which do not leverage QSP (Quadratic Span Program) or QAP (Quadratic Arithmetic Program) circuits [GKR08, CMT12, WJB+17, WTTW18, BBB+18, ZGK+18, BSCTV14]. A well-known branch comes from the sum-check protocol [GKR08], which gains a sublinear proof from the fiat-shamir transform [FS86]. Nonetheless, they do not support the constant time verification; the verification time is sublinear to the size of the circuits.

Since Gennaro et al. [GGPR13] introduced the Quadratic Span Program(QSP) and Quadratic Arithmetic Program(QAP), zk-SNARK gained a constant proof size and verification. In 2013, Parno et al. [PHGR13] proposed a zk-SNARK scheme called Pinocchio and provided a first practical implementation of zk-SNARK. After Pinocchio, many works added and enhanced some functionalities, such as multiple-function control, additional anonymity for the I/O, or proof scalability [CFH+15, DLFKP16, KPP+14, FFG+16, BBFR15, BSCTV17].

Later, Groth [Gro16] proposed a more efficient zk-SNARK scheme. Compared with Pinocchio [PHGR13], the proof size was reduced from 8 group elements to 3 group elements. Also the number of pairing operations required to verify the proof was reduced from 11 to 3. Recently these SNARK protocols are implemented as an open source [KPS18, BSCG+13] to be used in real applications. By exploiting the short proof sizes and the short verification times, zk-SNARK can be used as a key component in various cryptographic applications such as anonymous cryptocurrencies [BCG+14, KMS+16, GGM16].

Zerocash [BCG$^+$14], one of the anonymous cryptocurrencies based on blockchain technology, utilized a zk-SNARK to hide transaction information and to provide an efficient verification process. However, since zk-SNARKs [Gro16, PHGR13] do not provide simulation-extractability, zerocash has to add extra cryptographic primitives such as one-time signatures to avoid malleability attacks.

The SE-SNARK scheme [GM17] defines and provides the simulation-extractable SNARK (SE-SNARK), with a similar notion to the Signatures of knowledge [CL06]. While maintaining an efficient proof size of [Gro16], it can prevent the malleability attacks due to the simulation-extractability.

Recently, Bowe and Gabizon [BG18] put an effort to make Groth's scheme [Gro16] simulation-extractable by utilizing random oracle model, with additional hash in proofs and verification. However, the proof size and verification equations in their scheme is 5 group elements and 2 equations which is inefficient compared to [GM17]. And the security is proven in random oracle model. Lipmaa proposes a simulation-extractable SNARK scheme without using random oracle model [Lip19]. The security of the proposed scheme is proven under a new security assumption called subversion algebraic knowledge (SAK) assumption in which if an adversary $\mathcal{A}$ outputs a group element then $\mathcal{A}$ should know each exponent of known group elements or randomly generated group elements to build the group element. In the proposed scheme, the proof size is reduced to 4 group elements and 2 verification equations are required while QAP is supported.

3 Preliminaries

3.1 Notation

We denote the security parameter with $\lambda \in \mathbb{N}$. For functions $f, g : \mathbb{N} \to [0; 1]$ we write $f(\lambda) \approx g(\lambda)$ if $|f(\lambda) - g(\lambda)| = \lambda^{-\omega(1)}$. A function f is negligible if $f(\lambda) \approx 0$. We implicitly assume that the security parameter is available to all participants and the adversary. If S is a set, $x \overset{\$}{\leftarrow} S$ denotes the process of selecting x uniformly at random in S. If $\mathcal{A}$ is a probabilistic algorithm, $x \leftarrow \mathcal{A}(\cdot)$ denotes the process of running $\mathcal{A}$ on some proper input and returning output x.

We define that $\mathsf{trans}_{\mathcal{A}}$ includes all of $\mathcal{A}$'s inputs and outputs, including random coins for an algorithm $\mathcal{A}$. We use games in security definitions and proofs. A game $\mathcal{G}$ has a main procedure whose output is the output of the game. The notation $\Pr[\mathcal{G}]$ denotes the probability that the output is 1.

3.2 Relations

Given a security parameter 1^λ, a relation generator $\mathcal{R}$ returns a polynomial time decidable relation $R \leftarrow \mathcal{R}(1^\lambda)$. For $(\boldsymbol{\phi}, \boldsymbol{w}) \in R$ we say that $\boldsymbol{w}$ is a witness to the instance $\boldsymbol{\phi}$ being in the relation. We denote with $\mathcal{R}_\lambda$ the set of possible relations that $\mathcal{R}(1^\lambda)$ might output.

3.3 Zero-Knowledge Succinct Non-interactive Arguments of Knowledge

Definition 1. *A zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARK) for* $\mathcal{R}$ *is a set of four algorithms* $Arg = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vfy}, \mathsf{SimProve})$ *working as follows:*

- $(\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R)$*: the setup algorithm is a PPT algorithm which receives a relation* $R \in \mathcal{R}_\lambda$ *as input and outputs a common reference string* $\mathbf{crs}$ *and a simulation trapdoor* $\boldsymbol{\tau}$*.*
- $\boldsymbol{\pi} \leftarrow \mathsf{Prove}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{w})$*: the prover algorithm is a PPT algorithm which receives a common reference string* $\mathbf{crs}$ *as input for a relation* R *and* $(\boldsymbol{\phi}, \boldsymbol{w}) \in R$ *and outputs a proof* $\boldsymbol{\pi}$*.*
- $0/1 \leftarrow \mathsf{Vfy}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{\pi})$*: the verifier algorithm is a deterministic polynomial time algorithm which receives a common reference string* $\mathbf{crs}$*, an instance* $\boldsymbol{\phi}$ *and a proof* $\boldsymbol{\pi}$ *as input and outputs 0 (reject) or 1 (accept).*
- $\boldsymbol{\pi} \leftarrow \mathsf{SimProve}(\mathbf{crs}, \boldsymbol{\tau}, \boldsymbol{\phi})$*: the simulator is a PPT algorithm which receives a common reference string* $\mathbf{crs}$*, a simulation trapdoor* $\boldsymbol{\tau}$ *and an instance* $\boldsymbol{\phi}$ *as input and outputs a proof* $\boldsymbol{\pi}$*.*

It satisfies completeness, knowledge soundness, zero-knowledge, and succinctness as following:

Perfect Completeness: Perfect completeness states that a prover with a witness can convince the verifier for a given true instance. For all $\lambda \in \mathbb{N}$, for all $R \in \mathcal{R}_\lambda$ and for all $(\boldsymbol{\phi}, \boldsymbol{w}) \in R : Pr[(\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R); \boldsymbol{\pi} \leftarrow \mathsf{Prove}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{w}) : \mathsf{Vfy}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{\pi}) = 1] = 1$.

Computational Knowledge Soundness: Computational knowledge soundness says that the prover must know a witness and the witness can be efficiently extracted from the prover by a knowledge extractor. Proof of knowledge requires that there must exist an extract $\chi_{\mathcal{A}}$ given the same input of $\mathcal{A}$ outputs a valid witness for every adversarial prover $\mathcal{A}$ generating an accepting proof. Formally, we define $\mathbf{Adv}^{sound}_{Arg,\mathcal{A},\chi_{\mathcal{A}}}(\lambda) = Pr[\mathcal{G}^{sound}_{Arg,\mathcal{A},\chi_{\mathcal{A}}}(\lambda)]$ where the game $\mathcal{G}^{sound}_{Arg,\mathcal{A},\chi_{\mathcal{A}}}$ is defined as follows.

```
$\underline{\text{MAIN } \mathcal{G}^{sound}_{Arg,\mathcal{A},\chi_{\mathcal{A}}}(\lambda)}$
$R \leftarrow \mathcal{R}(1^\lambda)$
$(crs, \tau) \leftarrow \mathsf{Setup}(R)$
$(\phi, \pi) \leftarrow \mathcal{A}(crs)$
$\omega \leftarrow \chi_{\mathcal{A}}(trans_{\mathcal{A}})$
$assert\ (\phi, \omega) \notin R$
$return\ \mathsf{Vfy}(crs, \phi, \pi)$
```

An argument system Arg is computationally considered as knowledge sound if there exists a PPT extractor $\chi_{\mathcal{A}}$ for any PPT adversary $\mathcal{A}$, such that $\mathbf{Adv}^{sound}_{Arg,\mathcal{A},\chi_{\mathcal{A}}}(\lambda) \approx 0$.

Perfect Zero-Knowledge: Perfect zero-knowledge states that the system does not reveal any information except the truth of the instance. This is modelled by a simulator which can generate simulated proofs using some trapdoor information without knowing the witness. Formally, we define $\mathbf{Adv}^{zk}_{Arg,\mathcal{A}}(\lambda) = 2Pr[\mathcal{G}^{zk}_{Arg,\mathcal{A}(\lambda)}]-1$ where the game $\mathcal{G}^{zk}_{Arg,\mathcal{A}}$ is defined as follows:

```
MAIN $\mathcal{G}^{zk}_{Arg,\mathcal{A}}(\lambda)$
$R \leftarrow \mathcal{R}(1^\lambda)$
$(crs, \tau) \leftarrow \mathsf{Setup}(R)$
$b \leftarrow \{0,1\}$
$b' \leftarrow \mathcal{A}^{P^b_{crs,\tau}}(crs)$
$return\ 1\ if\ b = b'\ and$
$return\ 0\ otherwise$

$P^b_{crs,\tau}(\phi_i, w_i)$
$assert(\phi_i, w_i) \in R$
$\pi_i \leftarrow \mathsf{Prove}(crs, \phi, w)\ if\ b = 0$
$\pi_i \leftarrow \mathsf{SimProve}(crs, \tau, \phi)\ if\ b = 1$
$return\ \pi_i$
```

The argument system is perfectly zero-knowledge if for all PPT adversaries $\mathcal{A}$, $\mathbf{Adv}^{zk}_{Arg,\mathcal{A}}(\lambda) = 0$.

Succinctness: Succinctness states that the argument generates the proof of which size is polynomial in the security parameter, and of which the verifier's computation time is polynomial in the security parameter and in the instance size.

Definition 2. *A simulation-extractable SNARK system (SE-SNARK) for $\mathcal{R}$ is a zk-SNARK system* (Setup*,* Prove*,* Vfy*,* SimProve) *with simulation-extractability as following:*

Simulation-Extractability [GM17]: Simulation-extractability states that for any adversary $\mathcal{A}$ that sees a simulated proof for a false instance cannot modify the proof into another proof for a false instance. Non-malleability of proofs prevents cheating in the presence of simulated proofs. Formally, we define $\mathbf{Adv}^{proof-ext}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda) = Pr[\mathcal{G}^{proof-ext}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda)]$ where the game $\mathcal{G}^{proof-ext}_{Arg,\mathcal{A},\chi_\mathcal{A}}$ is defined as follows:

```
MAIN $\mathcal{G}^{proof-ext}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda)$
$R \leftarrow \mathcal{R}(1^\lambda); Q = \emptyset$
$(crs, \tau) \leftarrow \mathsf{Setup}(R)$
$(\phi, \pi) \leftarrow \mathcal{A}^{\mathsf{SimProve}_{\mathsf{crs},\tau}}(crs)$
$\omega \leftarrow \chi_\mathcal{A}(trans_\mathcal{A})$
$assert\ (\phi, \pi) \notin Q$
$assert\ (\phi, \omega) \notin R$
$return\ \mathsf{Vfy}(crs, \phi, \pi)$

$\mathsf{SimProve}_{crs,\tau}(\phi_i)$
$\pi_i \leftarrow \mathsf{SimProve}(crs, \tau, \phi_i)$
$Q = Q \cup \{(\phi_i, \pi_i)\}$
$return\ \pi_i$
```

An argument is simulation-extractable if for any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\chi_\mathcal{A}$ such that $\mathbf{Adv}^{proof-ext}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda) \approx 0$.

We note that simulation-extractability implies knowledge soundness, since simulation-extractability corresponds to knowledge soundness where the adversary is allowed to use the simulation oracle $\mathsf{SimProve}$.

When knowledge soundness and simulation-extractability are applied for a succinct argument, extractors are inherently non-black-box. As in [GM17] we assume the relationship generator is benign[5], such that the relation (including the potential auxiliary inputs) is distributed in such a way that the SNARK can be simulation-extractable.

4 Bilinear Groups and Assumptions

A bilinear group generator $\mathcal{BG}$ receives a security parameter as input and outputs a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$. $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are groups of prime order p with generator $G \in \mathbb{G}_1$, $H \in \mathbb{G}_2$, and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerative bilinear map (i.e. $e(G^a, H^b) = e(G, H)^{ab}$ and $e(G, H)$ generates $\mathbb{G}_T$).

4.1 Power Knowledge of Exponent Assumption

We define q-power knowledge of exponent assumption.

Definition 3 (q-PKE assumption). *[Gro10] The q-power knowledge of exponent assumption holds for $\mathbb{G}_1$, $\mathbb{G}_2$ if for all $\mathcal{A}$ there exists a non-uniform PPT extractor $\chi_{\mathcal{A}}$ such that*

$$Pr\left[\begin{array}{c}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H) \leftarrow \mathcal{BG}(1^\lambda); x \xleftarrow{\$} \mathbb{Z}_p;\\ \sigma \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, \{G^{x^i}\}_{i=1}^q, H, \{H^{x^i}\}_{i=1}^q);\\ (G^a, H^b) \leftarrow \mathcal{A}(\sigma); (a_0, \ldots, a_q) \leftarrow \chi_{\mathcal{A}}(trans_{\mathcal{A}}) :\\ a = b \wedge b \neq \sum_{i=0}^q a_i x^i\end{array}\right] \approx 0.$$

4.2 Hash-Algebraic Knowledge Assumption

Lipmaa proposes a new knowledge assumption called hash-algebraic knowledge (HAK) assumption [Lip19], which simply gives an adversary an additional ability to *hash* any element from the algebraic group model. In algebraic knowledge assumption, one assumes that each PPT algorithm is algebraic in the following sense. Assume that there are unknown exponents. Let x_i be a polynomial using the unknown exponents. Let G^{x} be a vector of G^{x_i}. Similarly, let G^{y} be a vector

[5] The non-falsifiable *knowledge of exponent* assumption is a necessary ingredient in building a SNARK with witness extraction. In Bitansky's analysis [BCI+13, BCPR16], there are some counter examples and observations; auxiliary inputs may affect the extraction of the witness in extractable one-way functions. However they also observe that the extractability still holds with respect to common auxiliary input that is taken from specific distributions that may be conjectured to be "benign", e.g. the uniform distribution.

of G^{y_i} where y_i is a polynomial using the unknown exponents. If the adversary $\mathcal{A}$'s input includes $G^{\mathbf{x}}$ and no other elements from the group $\mathbb{G}_1$ and $\mathcal{A}$ outputs group elements $G^{\mathbf{y}}$, then $\mathcal{A}$ knows matrices $\mathbf{N}$, such that $G^{\mathbf{y}} = G^{\mathbf{N}\mathbf{x}}$. Formally, a PPT algorithm $\mathcal{A}$ is algebraic (in $\mathbb{G}_1$) if there exists an efficient extractor $\chi_{\mathcal{A}}$, such that for any PPT sampleable distribution $\mathcal{D}$, $\mathbf{Adv}^{ak}_{\mathbb{G}_1,\mathcal{D},\mathcal{A}}(\lambda) \approx 0$, where $\mathbf{Adv}^{ak}_{\mathbb{G}_1,\mathcal{D},\mathcal{A}}(\lambda) := Pr[G^{\mathbf{x}} \overset{\$}{\leftarrow} \mathcal{D}; G^{\mathbf{y}} \leftarrow \mathcal{A}(G^{\mathbf{x}}); \mathbf{N} \leftarrow \chi_{\mathcal{A}}(trans_{\mathcal{A}}) : \mathbf{y} \neq \mathbf{N}\mathbf{x}]$. A group $\mathbb{G}_1$ is algebraic if every PPT algorithm $\mathcal{A}$ that obtains inputs from $\mathbb{G}_1$ and outputs elements in $\mathbb{G}_1$ is algebraic.

Furthermore, Lipmaa pointed out that the restriction that adversaries are algebraic is not valid in situations where the adversary can create new random group elements by say using elliptic curve hashing [Ica09]. So he models this capability by allowing the adversary to create additional group elements $G^{\mathbf{q}}$ for which she does not know discrete logarithms of exponent q_i or vector $\mathbf{q}$. It is required that $G^{\mathbf{q}}$ (but not necessarily $\mathbf{q}$) can be extracted from the adversary, such that $\mathbf{y} = \mathbf{N} \cdot \begin{pmatrix} \mathbf{x} \\ \mathbf{q} \end{pmatrix}$. In addition, $G^{\mathbf{q}}$ must be sampled from a public distribution $\mathcal{D}'$.

A PPT algorithm $\mathcal{A}$ is called as hash-algebraic (in $\mathbb{G}_1$) if there exists a PPT extractor $\chi_{\mathcal{A}}$, s.t. for any PPT sampleable distribution $\mathcal{D}$ and any distribution $\mathcal{D}'$ with min-entropy $\omega(\log \lambda)$, $\mathbf{Adv}^{hak}_{\mathbb{G}_1,\mathcal{D},\mathcal{D}',\mathcal{A}}(\lambda) :=$

$$Pr\left[\begin{array}{c} G^{\mathbf{x}} \overset{\$}{\leftarrow} \mathcal{D}; G^{\mathbf{y}} \leftarrow \mathcal{A}(G^{\mathbf{x}}); \\ (\mathbf{N}, G^{\mathbf{q}}) \leftarrow \chi_{\mathcal{A}}(trans_{\mathcal{A}}) : \mathbf{y} \neq \mathbf{N} \begin{pmatrix} \mathbf{x} \\ \mathbf{q} \end{pmatrix} \wedge (G^{\mathbf{q}} \sim \mathcal{D}') \end{array}\right] \approx 0.$$

Finally, we define the following $\mathcal{D} - HAK$ assumption in $\mathbb{G}_1$:

Definition 4 ($\mathcal{D} - HAK$ assumption in $\mathbb{G}_1$ [Lip19]). *For each PPT $\mathcal{A}$ that obtains inputs, distributed according to the distribution $\mathcal{D}$, there exists an extractor that outputs $G^{\mathbf{q}}$ and $\mathbf{N}$ such that $G^{\mathbf{q}} \sim \mathcal{D}'$ for some distribution $\mathcal{D}'$ of high min-entropy. More precisely, $\mathbf{Adv}^{hak}_{\mathbb{G}_\iota,\mathcal{D},\mathcal{D}',\mathcal{A}}(\lambda) \approx 0$ for each PPT adversary $\mathcal{A}$ and each distribution $\mathcal{D}'$ of min-entropy $\omega(\log \lambda)$.*

4.3 Linear Collision-Resistant Hash Function

We define collision-resistance and linear collision-resistance of a hash function.

Definition 5 (collision-resistance). *$\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ is a collision-resistant hash function if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}^{CR}_{\mathcal{H}}(\mathcal{A}) :=$*

$$Pr[(x, x') \leftarrow \mathcal{A}(\mathcal{X}, \mathcal{H}) : (x \neq x') \wedge (\mathcal{H}(x) = \mathcal{H}(x'))] \approx 0$$

Furthermore, it is difficult to find any collision for various equations in many collision-resistant hash functions like SHA2. Specifically for our purpose, we define a variant of a collision-resistant hash function called a *linear collision-resistant* hash function where it is hard to find non-trivial $x, x' \in \mathbb{Z}_p$ for given $G_1, G_2 \in \mathbb{G}$ where $\mathbb{G}$ is a cyclic group of prime order p such that $\mathcal{H}(G_1^x G_2^{x'}) = x + x'\mathcal{H}(G_2)$ for $\mathcal{H} : \mathbb{G} \to \mathbb{Z}_p$. Formally, it is defined as follows:

Definition 6 (Linear collision-resistance). *$\mathcal{H} : \mathbb{G} \to \mathbb{Z}_p$ is a linear collision-resistant hash function if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{H}}^{LCR}(\mathcal{A}) :=$*

$$Pr\left[\begin{matrix}(G_1, G_2) \xleftarrow{\$} \mathbb{G}, (x, x') \leftarrow \mathcal{A}(\mathbb{G}, \mathcal{H}, G_1, G_2) : \\ (x \neq 0) \wedge (x' \neq 1)) \wedge (\mathcal{H}(G_1^x G_2^{x'}) = x + x'\mathcal{H}(G_2))\end{matrix}\right] \approx 0$$

Any cryptographical hash functions such as SHA2, and Ajtai hash [Ajt96] which are used as a collision-resistant hash function can be also adopted as a linear collision-resistant hash function by treating input and output elements as strings if the output string is remapped in $\mathbb{Z}_p$.

It is difficult to analyze the exact security level of the linear collision-resistance. At least, we prove that the linear collision-resistance is as hard as the discrete log problem in the idealized hash function. In other words, given a random oracle $\mathcal{H}$, if we can find linear collision then we can break a discrete log problem.

Lemma 1. *For given $(\mathcal{H}, G_1, G_2)$ where $\mathcal{H}$ is a random oracle, if there is a PPT $\mathcal{A}$ such that $\mathbf{Adv}_{\mathcal{H}}^{LCR}(\mathcal{A})$ is non-negligible then there is a PPT algorithm $\mathcal{B}$ to compute $Dlog_{G_1}(G_2)$ with non-negligible probability.*

Proof. Assume that $\mathcal{A}$ finds (x, x') such that $\mathcal{H}(G_1^x G_2^{x'}) = x + x'\mathcal{H}(G_2)$. Let $v = G_1^x G_2^{x'}$ and $w = \mathcal{H}(v)$. By rewinding the hash query for v, we provide a different hash value of w' for v to $\mathcal{A}$. Then $\mathcal{A}$ outputs a valid (y, y') where $w' = \mathcal{H}'(v) = \mathcal{H}'(G_1^y G_2^{y'}) = y + y'\mathcal{H}(G_2)$ and $v = G_1^x G_2^{x'} = G_1^y G_2^{y'}$. Since $\frac{G_1^x}{G_1^y} = \frac{G_2^{y'}}{G_2^{x'}}$, $G_1^{\frac{x-y}{y'-x'}} = G_2$. Since we know x, x', y, y', we can compute $Dlog_{G_1}(G_2)$ which is $\frac{x-y}{y'-x'}$. □

5 QAP-based SE-SNARK Scheme

In this section, we propose our first SE-SNARK construction based on the quadratic arithmetic program (QAP) representation, which achieves a proof size of 3 elements and a single verification. Before presenting the formal construction, we briefly explain the main idea behind the scheme to achieve simulation-extractability without an additional check in section 5.1. Then we introduce the formal definition of QAP in section 5.2, and present the formal construction in section 5.3.

5.1 Main Idea

As an example of how standard zk-SNARK can be modified, suppose for an instance ϕ that (A, B, C) $(= (G^a, H^b, G^c))$ are three group elements in a proof that satisfies the verification equations of Groth's zk-SNARK [Gro16]. Then

$$e(A, B) = e(G^\alpha, H^\beta) e(G^{\frac{f(\phi)}{\gamma}}, H^\gamma) e(C, H^\delta) \quad (1)$$

for a known polynomial f in ϕ and some secret $\alpha, \beta, \gamma, \delta$.

There are two methods to generically randomize a proof A, B, C that satisfies (1). An adversary can set either

$$A' = A^r; B' = B^{\frac{1}{r}}; C' = C \tag{2}$$

or

$$A' = A; B' = BH^{r\delta}; C' = A^r C. \tag{3}$$

In the proposed approach, we devise a new way to neutralize the two attacks using the hash of A and B in C. The verification equation is required to detect the changes of A and B. We insert multiplications of a and hash of A, and b and hash of B in c. Hence, an adversary should know a and b to change A and B in the revised proof.

The left pairing function in (1) changes to $e(AG^{\delta\mathcal{H}(A)}, BH^{\mathcal{H}(B)})$, and C is revised to satisfy (1) as following:

$$C' = C \cdot G^{\frac{a\mathcal{H}(B)}{\delta} + b\mathcal{H}(A) + \mathcal{H}(A)\mathcal{H}(B)}$$

where $A = G^a$, $B = H^b$, and $\mathcal{H}$ is a linear collision-resistant hash function like SHA.

According to the revised C', the verification is revised by adding proper additional terms to A and B as follows:

$$e(A \cdot G^{\delta\mathcal{H}(A)}, B \cdot H^{\mathcal{H}(B)}) = e(G^\alpha, H^\beta)e(G^{\frac{f(\phi)}{\gamma}}, H^\gamma)e(C', H^\delta)$$

If A, B change to A', B' then C' should be revised to $C' \cdot G^{\frac{a(\mathcal{H}(B') - \mathcal{H}(B))}{\delta} + b(\mathcal{H}(A') - \mathcal{H}(A)) + \mathcal{H}(A')\mathcal{H}(B') - \mathcal{H}(A)\mathcal{H}(B)}$. However, since only G^a and H^b are available in the original proof, and $G^{\frac{a}{\delta}}$ and G^b are only computable if a witness is known, an adversary cannot forge the proof.

5.2 Quadratic Arithmetic Programs

In our SE-SNARK, we will formally adopt the quadratic arithmetic programs (QAP) [GGPR13, Gro16] in a relation R, which is as follows:

$$R = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, l, \{u_i(X), v_i(X), w_i(X)\}_{i=0}^m, t(X))$$

The bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ defines the finite field $\mathbb{Z}_p$, $1 \leq l \leq m$, and the polynomials $u_i(X), v_i(X), w_i(X)$ represent each linearly independent polynomial set in the QAP with the definition below:

$$\sum_{i=0}^{m} s_i u_i(X) \cdot \sum_{i=0}^{m} s_i v_i(X) \equiv \sum_{i=0}^{m} s_i w_i(X) + h(X)t(X)$$

where $u_i(X), v_i(X), w_i(X)$ have a strictly lower degree than n, which is the degree of $t(X)$. By defining s_0 as 1, the following definition describes the relation R.

$$R = \left\{ (\phi, w) \middle| \begin{array}{l} \phi = (s_1, \cdots, s_l) \in \mathbb{Z}_p^l \\ w = (s_{l+1}, \cdots, s_m) \in \mathbb{Z}_p^{m-l} \\ \\ \exists h(X) \in \mathbb{Z}_p[X], deg(h) \leq n-2 : \\ \qquad \sum_{i=0}^{m} s_i u_i(X) \cdot \sum_{i=0}^{m} s_i v_i(X) \equiv \sum_{i=0}^{m} s_i w_i(X) + h(X)t(X) \end{array} \right\}$$

We say $\mathcal{R}$ is a relation generator for the QAP, given the relation R with field size larger than $2^{\lambda-1}$.

5.3 Construction

– $(crs, \tau) \leftarrow \mathsf{Setup}(R)$: Select generators $G \xleftarrow{\$} \mathbb{G}_1, H \xleftarrow{\$} \mathbb{G}_2$, hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$ and parameters $\alpha, \beta, \gamma, \delta, x \xleftarrow{\$} \mathbb{Z}_p$, such that $t(x) \neq 0$, and set

$$\tau = (G, H, \alpha, \beta, \gamma, \delta, x)$$
$$crs = \begin{pmatrix} R, \mathcal{H}, G, G^{\alpha}, G^{\beta}, G^{\delta}, G^{\alpha\delta}, H, H^{\beta}, H^{\delta} \\ \{G^{\gamma x^i}, H^{\gamma x^i}, G^{\gamma^2 t(x) x^i}, G^{\gamma\delta x^i}\}_{i=0}^{n-1}, \{G^{\gamma w_i(x) + \beta u_i(x) + \alpha v_i(x)}\}_{i=0}^{l}, \\ \{G^{\gamma^2 w_i(x) + \beta\gamma u_i(x) + \alpha\gamma v_i(x)}\}_{i=l+1}^{m} \end{pmatrix}$$

– $\pi \leftarrow \mathsf{Prove}(crs, \phi, w)$: Set $s_0 = 1$ and parse ϕ as $(s_1, \ldots, s_l) \in \mathbb{Z}_p^l$ and w as $(s_{l+1}, \ldots, s_m) \in \mathbb{Z}_p^{m-l}$. Use the witness to compute $h(X)$ from the QAP, choose $r, s \xleftarrow{\$} \mathbb{Z}_p$ and compute $\pi = (A, B, C) = (G^a, H^b, G^c)$ such that

$$\begin{aligned} a =& \alpha + \gamma \sum_{i=0}^{m} s_i u_i(x) + r \\ b =& \beta + \gamma \sum_{i=0}^{m} s_i v_i(x) + s \\ c =& \sum_{i=l+1}^{m} s_i(\gamma^2 w_i(x) + \beta\gamma u_i(x) + \alpha\gamma v_i(x)) + \gamma^2 t(x)h(x) + sa + rb - rs \\ & + \delta a \mathcal{H}(B) + b\mathcal{H}(A) + \delta \mathcal{H}(A)\mathcal{H}(B) \end{aligned}$$

.

– $0/1 \leftarrow \mathsf{Vfy}(crs, \phi, \pi)$: Parse ϕ as $(s_1, \ldots, s_l) \in \mathbb{Z}_p^l$ and π as $(A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$. Set $s_0 = 1$ and accept the proof if and only if the following equation is satisfied:

$$e(AG^{\mathcal{H}(A)}, BH^{\delta\mathcal{H}(B)}) = e(G^{\alpha}, H^{\beta}) e(G^{\sum_{i=0}^{l} s_i(\gamma w_i(x) + \beta u_i(x) + \alpha v_i(x))}, H^{\gamma}) e(C, H)$$

– $\pi \leftarrow \mathsf{SimProve}(crs, \tau, \phi)$: Choose $\mu, \nu \xleftarrow{\$} \mathbb{Z}_p$ and compute $\pi = (A, B, C)$ such that

$$A = G^{\mu}, B = H^{\nu},$$
$$C = G^{\mu\nu - \alpha\beta + h_2\delta\mu + h_1\nu + h_1 h_2\delta - \gamma \sum_{i=0}^{l} s_i(\gamma w_i(x) + \beta u_i(x) + \alpha v_i(x))} \tag{4}$$

where $h_1 = \mathcal{H}(A)$ and $h_2 = \mathcal{H}(B)$.

5.4 Security Proof

Theorem 1. *The QAP-based SE-SNARK protocol is a non-interactive zero-knowledge argument of knowledge with perfect completeness and perfect zero-knowledge. It is simulation-extractable (implying it also has knowledge soundness) provided that the HAK (hash-algebraic knowledge) assumption holds, and a linear collision-resistant hash exists.*

Proof. Perfect Completeness: We demonstrate that the prover can compute the proof (A, B, C) as described from the common reference string. Let $h_1 = \mathcal{H}(A)$ and $h_2 = \mathcal{H}(B)$. The prover can compute the coefficients of

$$h(X) = \frac{(\sum_{i=0}^{m} s_i u_i(X))(\sum_{i=0}^{m} s_i v_i(X)) - (\sum_{i=0}^{m} s_i w_i(X))}{t(X)} = \sum_{j=0}^{n-2} \tilde{h}_j X^j.$$

Now, the proof elements can be computed as follows:

$$A = G^{\alpha} \prod_{j=0}^{n-1} (G^{\gamma x^j})^{u_j} \cdot G^r$$
$$B = H^{\beta} \prod_{j=0}^{n-1} (H^{\gamma x^j})^{v_j} \cdot H^s$$
$$C = \prod_{i=l+1}^{m} G^{s_i(\gamma^2 w_i(x) + \beta\gamma u_i(x) + \alpha\gamma v_i(x))} \cdot A^s A'^{h_2} B'^{(r+h_1)} \cdot G^{-rs} \cdot G^{\delta h_1 h_2} \cdot \prod_{j=0}^{n-1} (G^{\gamma^2 t(x) x^j})^{\tilde{h}_j}$$

where $A' = A^{\delta} = G^{\alpha\delta} \prod_{j=0}^{n-1} (G^{\delta\gamma x^j})^{u_j} \cdot G^{\delta r}$ and $B' = G^{\beta} \prod_{j=0}^{n-1} (G^{\gamma x^j})^{v_j} \cdot G^s$.

This computation provides us the proof elements specified in the construction

$$A = G^{\alpha + \gamma \sum_{i=0}^{m} s_i u_i(x) + r}$$
$$B = H^{\beta + \gamma \sum_{i=0}^{m} s_i v_i(x) + s}$$
$$C = G^{\sum_{i=l+1}^{m} s_i(\gamma^2 w_i(x) + \beta\gamma u_i(x) + \alpha\gamma v_i(x)) + \gamma^2 t(x) h(x) + sa + rb - rs + \delta a h_2 + b h_1 + \delta h_1 h_2}.$$

Here we show that the verification equation holds.

$$e(AG^{h_1}, BH^{\delta h_2}) = e(G^{\alpha}, H^{\beta}) e(G^{\sum_{i=0}^{l} s_i(\gamma w_i(x) + \beta u_i(x) + \alpha v_i(x))}, H^{\gamma}) e(C, H)$$

Taking discrete logarithms, checking the verification equation is equivalent to showing that

$$\begin{aligned}
&(a+h_1)\cdot(b+\delta h_2)\\
&\quad=(\alpha+\gamma\sum_{i=0}^{m}s_iu_i(x)+r)\cdot(\beta+\gamma\sum_{i=0}^{m}s_iv_i(x)+s)+\delta ah_2+bh_1+\delta h_1h_2\\
&\quad=\alpha\beta+\gamma^2(\sum_{i=0}^{m}s_iu_i(x))(\sum_{i=0}^{m}s_iv_i(x))+\sum_{i=0}^{m}s_i(\beta\gamma u_i(x)+\alpha\gamma v_i(x))\\
&\qquad+rb+sa-rs+\delta ah_2+bh_1+\delta h_1h_2\\
&\quad=\alpha\beta+\sum_{i=0}^{m}s_i(\gamma^2w_i(x)+\beta\gamma u_i(x)+\alpha\gamma v_i(x))+\gamma^2t(x)h(x)\\
&\qquad+rb+sa-rs+\delta ah_2+bh_1+\delta h_1h_2\\
&\quad=\alpha\beta+\gamma\sum_{i=0}^{l}s_i(\gamma w_i(x)+\beta u_i(x)+\alpha v_i(x))+\sum_{i=l+1}^{m}s_i(\gamma^2w_i(x)+\beta\gamma u_i(x)+\alpha\gamma v_i(x))\\
&\qquad+\gamma^2t(x)h(x)+rb+sa-rs+\delta ah_2+bh_1+\delta h_1h_2\\
&\quad=\alpha\beta+\gamma\sum_{i=0}^{l}s_i(\gamma w_i(x)+\beta u_i(x)+\alpha v_i(x))+c
\end{aligned}$$

where $A=G^a$, $B=H^b$ and $C=G^c$.

Note that since the vector $(s_{l+1},\ldots,s_m)$ is a valid witness for the instance $(s_1,\ldots,s_l)$, $(\sum_{i=0}^{m}s_iu_i(X))(\sum_{i=0}^{m}s_iv_i(X))=\sum_{i=0}^{m}s_iw_i(X)+h(X)t(X)$ for all $X\in\mathbb{Z}_p$.

ZERO-KNOWLEDGE: For the zero-knowledge, notice that the construction already provides the simulation SimProve which always produces verifying proofs. It can be observed that we obtain the same distribution over the real proof and the simulated proof, with the choice of random r,s in real proofs and the choice of random μ,ν in simulated proofs.

SIMULATION-EXTRACTABILITY: Assume that adversary $\mathcal{A}$ succeeds to forge a proof (A,B,C).

Our common reference string consists of group generators G, H raised to exponents that are polynomials in X_α, X_β, X_γ, X_δ, X_x evaluated on secret values $\alpha,\beta,\gamma,\delta,x$. Moreover, whenever $\mathcal{A}$ queries the simulation oracle, it gets back a simulated proof of $(A_i,B_i,C_i)_{i=1}^{q}$, which is a set of three group elements that can be computed by raising G,H to polynomials in indeterminates X_α, X_β, X_γ, X_δ, $X_x,X_{\mu_1},X_{\nu_1},\ldots,X_{\mu_q},X_{\nu_q}$ where we plug in randomly generated $\mu_1,\nu_1,\ldots,\mu_q,\nu_q$ for the latter ones.

By $D-HAK$, given a proof $\pi=(G^a,H^b,G^c)$, we can extract $a(\mathbf{X})$, $b(\mathbf{X})$, and $c(\mathbf{X})$ where $\mathbf{X}$ is an indeterminates vector. Note that X_{λ_j} (X_{ρ_j}) denotes an indeterminate to obtain G^{λ_j} (H^{ρ_j}) which is a randomly created group element by an adversary in $\mathbb{G}_1$ ($\mathbb{G}_2$) where λ_j (ρ_j) is unknown. Then the possible $a(\mathbf{X}), b(\mathbf{X})$, and $c(\mathbf{X})$ are as follows:

$$\begin{aligned}
&a(\mathbf{X}) = a_0 + a_\alpha X_\alpha + a_\beta X_\beta + a_\delta X_\delta + a_{\alpha\delta} X_\alpha X_\delta + \textstyle\sum_{i=0}^{n-1} a_{\gamma x^i} X_\gamma X_x^i \\
&+ \textstyle\sum_{i=0}^{n-1} a_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \displaystyle\sum_{i=0}^{n-1} a_{\gamma\delta x^i} X_\gamma X_\delta X_x^i \\
&+ \sum_{i=0}^{l} a_{s_i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) \\
&+ \sum_{i=l+1}^{m} a_{s_i} (X_\gamma^2 w_i(X_x) + X_\beta X_\gamma u_i(X_x) + X_\alpha X_\gamma v_i(X_x)) + \sum_{j=1}^{q} a_{\lambda_j} X_{\lambda_j} + \sum_{j=1}^{q} a_{A_j} X_{\mu_j} \\
&+ \sum_{j=1}^{q} a_{C_j} (X_{\mu_j} X_{\nu_j} - X_\alpha X_\beta - X_\gamma \sum_{i=0}^{l} s_{j,i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) \\
&+ h_2 X_\delta X_{\mu_j} + h_1 X_{\nu_1} + h_1 h_2 X_\delta)
\end{aligned}$$

$$\begin{aligned}
&b(\mathbf{X}) = b_0 + b_\beta X_\beta + b_\delta X_\delta + \textstyle\sum_{i=0}^{n-1} b_{\gamma x^i} X_\gamma X_x^i + \displaystyle\sum_{j=1}^{q} b_{\rho_j} X_{\rho_j} + \sum_{j=1}^{q} b_{B_j} X_{\nu_j} \\
&c(\mathbf{X}) = c_0 + c_\alpha X_\alpha + c_\beta X_\beta + c_\delta X_\delta + c_{\alpha\delta} X_\alpha X_\delta + \textstyle\sum_{i=0}^{n-1} c_{\gamma x^i} X_\gamma X_x^i \\
&+ \textstyle\sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \displaystyle\sum_{i=0}^{n-1} c_{\gamma\delta x^i} X_\gamma X_\delta X_x^i \\
&+ \sum_{i=0}^{l} c_{s_i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) \\
&+ \sum_{i=l+1}^{m} c_{s_i} (X_\gamma^2 w_i(X_x) + X_\beta X_\gamma u_i(X_x) + X_\alpha X_\gamma v_i(X_x)) + \sum_{j=1}^{q} c_{\lambda_j} X_{\lambda_j} + \sum_{j=1}^{q} c_{A_j} X_{\mu_j} \\
&+ \sum_{j=1}^{q} c_{C_j} (X_{\mu_j} X_{\nu_j} - X_\alpha X_\beta - X_\gamma \sum_{i=0}^{l} s_{j,i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) \\
&+ h_2 X_\delta X_{\mu_j} + h_1 X_{\nu_j} + h_1 h_2 X_\delta)
\end{aligned}$$

$a(\mathbf{X}), b(\mathbf{X})$, and $c(\mathbf{X})$ should satisfy the following verification equation.

$$\begin{aligned}
&(a(\mathbf{X}) + h_1)(b(\mathbf{X}) + h_2 X_\delta) \\
&= X_\alpha X_\beta + X_\gamma \sum_{i=0}^{l} a_{s_i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) + c(\mathbf{X})
\end{aligned} \tag{5}$$

We will now show that in order to satisfy the formal polynomials equations above, either the adversary must recycle an instance and a proof, or alternatively $\chi_{\mathcal{A}}$ manages to extract a witness.

First, suppose we have some $a_{A_k} \neq 0$. Since there is no $X_\beta X_{\mu_k}$ in the right form, $b_\beta = 0$. Moreover, since there is no $X_\gamma X_{\mu_k}$ or $X_{\rho_j} X_{\mu_k}$ in the right form, $b_{\gamma x^i} = 0$ and $b_{\rho_j} = 0$. Consequently, $b(\mathbf{X}) = b_0 + b_\delta X_\delta + b_{B_k} X_{v_k}$. If b_{B_k}=0 then $c_{C_k} = 0$ due to no $X_{\mu_k} X_\nu$, and there is $X_\alpha X_\beta$ in the right form. However since there is no $X_\alpha X_\beta$ in the left form, $b_{B_k} \neq 0$.

Since there is no $X_\alpha X_{\nu_k}$ in the right form, $a_\alpha = 0$. Since there are only $X_\alpha X_{\nu_k}$, X_{ν_k}, and $X_{\mu_k} X_{\nu_k}$ related with X_{ν_k} in the right form, $a(\mathbf{X}) = a_0 + a_{A_k} X_{\mu_k}$.

Plugging this into (5) gives us,

$$
\begin{aligned}
&(a_0 + a_{A_k} X_{\mu_k} + h_1')(b_0 + b_\delta X_\delta + b_{B_k} X_{v_k} + h_2' X_\delta) \\
&= X_\alpha X_\beta + X_\gamma \sum_{i=0}^{l} a_{s_i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) + c(\mathbf{X})
\end{aligned}
$$

where $h_1' = \mathcal{H}(G^{a_0 + a_{A_k}\mu_k}) = \mathcal{H}(G^{a_0} A_k^{a_{A_k}})$ and $h_2' = \mathcal{H}(H^{b_0 + b_\delta \delta + b_{B_k}\nu_k}) = \mathcal{H}(H^{b_0} H^{\delta b_\delta} B_k^{b_{B_k}})$.

The only way this is possible is by setting

$$
\begin{aligned}
c(\mathbf{X}) = c_0 + c_{A_k} X_{\mu_k} + c_{C_k} (&X_{\mu_k} X_{\nu_k} - X_\alpha X_\beta + h_2 X_\delta X_{\mu_k} + h_1 X_{\nu_k} + h_1 h_2 X_\delta \\
&- X_\gamma \sum_{i=0}^{l} s_{k,i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x))
\end{aligned}
$$

Since there is no $X_\alpha X_\beta$ in the left form, $c_{C_k} = 1$.

Finally, we obtain the following equation.

$$
\begin{aligned}
&(a_0 + a_{A_k} X_{\mu_k} + h_1')(b_0 + b_\delta X_\delta + b_{B_k} X_{\nu_k} + h_2' X_\delta) \\
&= c_0 + c_{A_k} X_{\mu_k} + X_{\mu_k} X_{\nu_k} + h_2 X_\delta X_{\mu_k} + h_1 X_{\nu_k} + h_1 h_2 X_\delta
\end{aligned}
$$

Since $a_{A_k} b_{B_k} = 1$, there is $(a_0 + h_1') b_{B_k} X_{\nu_k}$ in the left form, and there is $h_1 X_{\nu_k}$ in the right form, $(a_0 + h_1') b_{B_k} = h_1$, $h_1' = -a_0 + a_{A_k} h_1$, and $\mathcal{H}(G^{-(-a_0)} A_k^{a_{A_k}}) = -a_0 + a_{A_k} \mathcal{H}(A_k)$. Since $\mathcal{H}$ is linear collision-resistant, it is hard to find non trivial $-a_0$ and a_{A_k}. Hence $a_0 = 0$, and $a_{A_k} = 1$. Similarly, Since there is $(b_\delta + h_2') a_{A_k} X_\delta X_{\mu_k}$ in the left form, and there is $h_2 X_\delta X_{\mu_k}$ in the right form, $(b_\delta + h_2') a_{A_k} = h_2$, $h_2' = -b_\delta + b_{B_k} h_2$, and $\mathcal{H}(H^{b_0} H^{-(-b_\delta)} B_k^{b_{B_k}}) = -b_\delta + b_{B_k} \mathcal{H}(B_k)$. Since $\mathcal{H}$ is collision-resistant, it is hard to find non trivial b_0 such that $\mathcal{H}(H^{b_0} H^{b_\delta} B_k^{b_{B_k}}) = \mathcal{H}(H^{b_\delta} B_k^{b_{B_k}})$. Hence $b_0 = 0$. In addition, since $\mathcal{H}$ is linear collision-resistant, it is hard to find non trivial $-b_\delta$ and b_{B_k} satisfying $\mathcal{H}(H^{-(-b_\delta)} B_k^{b_{B_k}}) = -b_\delta + b_{B_k} \mathcal{H}(B_k)$. Hence $b_\delta = 0$, and $b_{B_k} = 1$.

Consequently, $a(\mathbf{X}) = X_{\mu_k}$ and $b(\mathbf{X}) = X_{\nu_k}$. Since $u_i(X_x)_{i=1}^{l}$ are linearly independent, we see for $i = 1, \ldots, l$ that $s_i = s_{k,i}$. In other words, the adversary has recycled the k-th instance $\pi = \pi_k$ and the proof $(A, B, C) = (A_k, B_k, C_k)$. The same conclusion is obtained if $b_{B_k} \neq 0$.

Next, suppose for all $j = 1, \ldots, q$ that $a_{A_j} = b_{B_j} = 0$. Then $c_{C_j} = c_{A_j} = 0$ since there is no X_{μ_j} in the left form. Since there is $X_\alpha X_\beta$ in the right form, $a_\alpha b_\beta = 1$.

In the right form of (5), there are only X_β, $X_\beta X_\gamma$, $X_\beta X_\alpha$, and $X_\beta u_i(X_x)$ related with X_β, $a(\mathbf{X}) = a_0 + a_\alpha X_\alpha + \sum_{i=0}^{n-1} a_{\gamma x^i} X_\gamma X_x^i$. $b(\mathbf{X}) = b_0 + b_\beta X_\beta + b_\delta X_\delta + \sum_{i=0}^{n-1} b_{\gamma x^i} X_\gamma X_x^i$ since there is no $X_\alpha X_{\rho_j}$ in the right form. We are now left with

$$c(\mathbf{X}) = c_0 + c_\alpha X_\alpha + c_\beta X_\beta + c_\delta X_\delta + c_{\alpha\delta} X_\alpha X_\delta + \sum_{i=0}^{n-1} c_{\gamma x^i} X_\gamma X_x^i$$
$$+ \sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} c_{\gamma\delta x^i} X_\gamma X_\delta X_x^i$$
$$+ \sum_{i=0}^{l} c_{s_i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x))$$
$$+ \sum_{i=l+1}^{m} c_{s_i} (X_\gamma^2 w_i(X_x) + X_\beta X_\gamma u_i(X_x) + X_\alpha X_\gamma v_i(X_x))$$

In (5),

$$(a_\alpha X_\alpha + a_0 + \sum_{i=0}^{n-1} a_{\gamma x^i} X_\gamma X_x^i + h_1')(b_\beta X_\beta + b_0 + \sum_{i=0}^{n-1} b_{\gamma x^i} X_\gamma X_x^i + (b_\delta + h_2') X_\delta)$$
$$= X_\alpha X_\beta + X_\gamma \sum_{i=0}^{l} a_{s_i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x))$$
$$+ c_0 + c_\alpha X_\alpha + c_\beta X_\beta + c_\delta X_\delta + c_{\alpha\delta} X_\alpha X_\delta + \sum_{i=0}^{n-1} c_{\gamma x^i} X_\gamma X_x^i$$
$$+ \sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} c_{\gamma\delta x^i} X_\gamma X_\delta X_x^i$$
$$+ \sum_{i=0}^{l} c_{s_i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x))$$
$$+ \sum_{i=l+1}^{m} c_{s_i} (X_\gamma^2 w_i(X_x) + X_\beta X_\gamma u_i(X_x) + X_\alpha X_\gamma v_i(X_x))$$

Define for $i = l+1, \ldots, m$ that $s_i = c_{s_i}$. The terms involving $X_\beta X_\gamma X_x^i$ now give us $b_\beta \sum_{i=0}^{n-1} a_{\gamma x^i} X_x^i = \sum_{i=0}^{m} s_i u_i(X_x)$ in the left form. In addition, the terms involving $X_\alpha X_\gamma X_x^i$ provide $a_\alpha \sum_{i=0}^{n-1} b_{\gamma x^i} X_x^i = \sum_{i=0}^{m} s_i v_i(X_x)$ in the left form. The terms involving X_γ^2 produce

$$X_\gamma \sum_{i=0}^{m} s_i u_i(X_x) \cdot X_\gamma \sum_{i=0}^{m} s_i v_i(X_x) = X_\gamma^2 a_\alpha b_\beta (\sum_{i=0}^{n-1} a_{\gamma x^i} X_x^i)(\sum_{i=0}^{n-1} b_{\gamma x^i} X_x^i)$$
$$= X_\gamma^2 (\sum_{i=0}^{m} s_i w_i(X_x) + t(X_x) \sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_x^i)$$

Defining $h(X_x) = \sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_x^i$ we see that this means $(s_{l+1}, \ldots, s_m)$ is a witness for the instance $(s_1, \ldots, s_l)$ (the extracted witness may be one of many possible valid witnesses).

$\square$

6 SAP-based SE-SNARK Scheme

In the previous section, we propose an efficient SE-SNARK scheme with three group elements as a proof. Now it is interesting to observe whether it is possible to build a similar SE-SNARK scheme with two group elements if adopting Type I pairing instead of Type III pairing. Since each multiplication gate $a \cdot b = c$ can be transformed to $(a+b)^2 - (a-b)^2 = 4c$ as a square arithmetic program (SAP), it is possible to get a 2-element for boolean circuit satisfiability by changing a multiplication gate to two squaring gates.

6.1 Square Arithmetic Programs

In the SE-SNARK with two group elements, we will work with square arithmetic programs (SAP) R, with the definitions adopted from [GM17].

$$R = (p, \mathbb{G}, \mathbb{G}_T, e, l, \{u_i(X), w_i(X)\}_{i=0}^{m}, t(X))$$

The bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e)$ defines the finite field $\mathbb{Z}_p$, $1 \le l \le m$, and the polynomials $u_i(X), w_i(X)$ represent each linearly independent polynomial set in the SAP with the definition below:

$$(\sum_{i=0}^{m} s_i u_i(X))^2 \equiv \sum_{i=0}^{m} s_i w_i(X) + h(X)t(X)$$

where $u_i(X), w_i(X)$ have a strictly lower degree than n, which is the degree of $t(X)$. By defining s_0 as 1, the following definition describes the relation R.

$$R = \left\{ (\phi, w) \middle| \begin{array}{l} \phi = (s_1, \cdots, s_l) \in \mathbb{Z}_p^l \\ w = (s_{l+1}, \cdots, s_m) \in \mathbb{Z}_p^{m-l} \\ \\ \exists h(X) \in \mathbb{Z}_p[X], deg(h) \le n-2 : \\ \qquad (\sum_{i=0}^{m} s_i u_i(X))^2 \equiv \sum_{i=0}^{m} s_i w_i(X) + h(X)t(X) \end{array} \right\}$$

We say $\mathcal{R}$ is a relation generator for the SAP, given the relation R with a field size larger than $2^{\lambda-1}$.

6.2 Construction

In this section, we propose a scheme with two group elements as a proof in a symmetric group using SAP.

- $(crs, \tau) \leftarrow \mathsf{Setup}(R)$: Select a generator $G \overset{\$}{\leftarrow} \mathbb{G}$, hash functions $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$, and parameters $\alpha, \gamma, \delta, x \overset{\$}{\leftarrow} \mathbb{Z}_p$, such that $t(x) \ne 0$, and set

$$\tau = (G, \alpha, \gamma, \delta, x)$$

$$crs = \begin{pmatrix} R, H, G, G^{\alpha}, G^{\delta}, G^{\alpha\delta}, \\ \{G^{\gamma x^i}, G^{\gamma^2 t(x) x^i}, G^{\gamma\delta x^i}\}_{i=0}^{n-1}, \{G^{\gamma w_i(x)+2\alpha u_i(x)}\}_{i=0}^{l}, \\ \{G^{\gamma^2 w_i(x)+2\alpha\gamma u_i(x)}\}_{i=l+1}^{m} \end{pmatrix}$$

- $\pi \leftarrow \mathsf{Prove}(crs, \phi, w)$: Set $s_0 = 1$ and parse ϕ as $(s_1, \ldots, s_l) \in \mathbb{Z}_p^l$ and w as $(s_{l+1}, \ldots, s_m) \in \mathbb{Z}_p^{m-l}$. Use the witness to compute $h(X)$ from the SAP, pick $r \xleftarrow{\$} \mathbb{Z}_p$ and compute $\pi = (A, C) = (G^a, G^c)$ such that

$$a = \alpha + \gamma \sum_{i=0}^{m} s_i u_i(x) + r$$
$$c = \sum_{i=l+1}^{m} s_i(\gamma^2 w_i(x) + 2\alpha\gamma u_i(x)) + \gamma^2 t(x) h(x) + 2ra - r^2 + \delta a \mathcal{H}(A)$$

- $0/1 \leftarrow \mathsf{Vfy}(crs, \phi, \pi)$: Parse ϕ as $(s_1, \ldots, s_l) \in \mathbb{Z}_p^l$ and π as $(A, C) \in \mathbb{G} \times \mathbb{G}$. Set $s_0 = 1$ and check that

$$e(AG^{\delta\mathcal{H}(A)}, A) = e(G^{\alpha}, G^{\alpha}) e(G^{\sum_{i=0}^{l} s_i(\gamma w_i(x)+2\alpha u_i(x))}, G^{\gamma}) e(C, G)$$

  Accept the proof if and only if the test passes.
- $\pi \leftarrow \mathsf{SimProve}(crs, \tau, \phi)$: Pick $\mu \leftarrow \mathbb{Z}_p$ and compute $\pi = (A, C)$ such that

$$A = G^{\mu}, C = G^{\mu^2 - \alpha^2 + \delta\mu\mathcal{H}(A) - \gamma \sum_{i=0}^{l} s_i(\gamma w_i(x)+2\alpha u_i(x))}$$

6.3 Security Proof

Theorem 2. *The SAP-based SE-SNARK protocol is a non-interactive zero-knowledge argument of knowledge with perfect completeness and perfect zero-knowledge. It is simulation-extractable (implying it also has knowledge soundness) provided that the $D-HAK$ assumption holds and a collision-resistant hash function exists.*

Proof. PERFECT COMPLETENESS: First, we state that the prover can compute the proof (A, C) as described from the common reference string. The prover can compute the coefficients of

$$h(X) = \frac{(\sum_{i=0}^{m} s_i u_i(X))^2 - (\sum_{i=0}^{m} s_i w_i(X))}{t(X)} = \sum_{j=0}^{n-2} h_j X^j.$$

It can now compute the proof elements as

$$A = G^{\alpha} \prod_{j=0}^{n-1} (G^{\gamma x^j})^{u_j} \cdot G^r$$
$$C = \prod_{i=l+1}^{m} G^{s_i(\gamma^2 w_i(x)+2\alpha\gamma u_i(x))} \cdot A'^{\mathcal{H}(A)} \cdot G^{-r^2} \cdot \prod_{j=0}^{n-1} (G^{\gamma^2 t(x) x^j})^{h_j}$$

where let $A' = G^{\alpha\delta}A^{\delta} = G^{\alpha\delta}\prod_{j=0}^{n-1}(G^{\delta\gamma x^j})^{u_j} \cdot G^{\delta r}$.

This computation provides us the proof elements specified in the construction

$$A = G^{\alpha+\gamma\sum_{i=0}^{m} s_i u_i(x)+r}$$
$$C = G^{\sum_{i=l+1}^{m} s_i(\gamma^2 w_i(x)+2\alpha\gamma u_i(x))+\gamma^2 t(x)h(x)+2ra-r^2+\delta a\mathcal{H}(A)}$$

Here we show that the verification equation holds.

$$e(AG^{\delta\mathcal{H}(A)}, A) = e(G^{\alpha}, G^{\alpha})e(G^{\sum_{i=0}^{l} s_i(\gamma w_i(x)+2\alpha u_i(x))}, G^{\gamma})e(C, G)$$

Taking discrete logarithms, this is equivalent to showing that

$$\begin{aligned}
(a+&\delta\mathcal{H}(A)) \cdot a = a^2 + \delta a\mathcal{H}(A) \\
&=(\alpha + \gamma\sum_{i=0}^{m} s_i u_i(x) + r)^2 + \delta a\mathcal{H}(A) \\
&=\alpha^2 + \gamma^2(\sum_{i=0}^{m} s_i u_i(x))^2 + 2\alpha\gamma\sum_{i=0}^{m} s_i u_i(x) + 2ra - r^2 + \delta a\mathcal{H}(A) \\
&=\alpha^2 + \sum_{i=0}^{m} s_i(\gamma^2 w_i(x) + 2\alpha\gamma u_i(x)) + \gamma^2 t(x)h(x) + 2ra - r^2 + \delta a\mathcal{H}(A) \\
&=\alpha^2 + \gamma\sum_{i=0}^{l} s_i(\gamma w_i(x) + 2\alpha u_i(x)) \\
&\quad + \sum_{i=l+1}^{m} s_i(\gamma^2 w_i(x) + 2\alpha\gamma u_i(x)) + \gamma^2 t(x)h(x) + 2ra - r^2 + \delta a\mathcal{H}(A) \\
&=\alpha^2 + \gamma\sum_{i=0}^{l} s_i(\gamma w_i(x) + 2\alpha u_i(x)) + c
\end{aligned}$$

where $A = G^a$, and $C = G^c$.

Note that since the vector $(s_{l+1}, \ldots, s_m)$ is a valid witness for the instance $(s_1, \ldots, s_l)$, $(\sum_{i=0}^{m} s_i u_i(X))^2 = \sum_{i=0}^{m} s_i w_i(X) + h(X)t(X)$ for all $X \in \mathbb{Z}_p$.

Zero-Knowledge: The zero-knowledge is similar to the proof in 5.4; the Sim-Prove in the algorithm provides the proof simulation, which is sufficient for the zero-knowledge.

SIMULATION-EXTRACTABILITY: By $D-HAK$ assumption, there is an extractor and $a(\mathbf{X})$, and $c(\mathbf{X})$ are extracted as following:

$$\begin{aligned}
a(\mathbf{X}) &= a_0 + a_\alpha X_\alpha + a_\delta X_\delta + a_{\alpha\delta} X_\alpha X_\delta + \textstyle\sum_{i=0}^{n-1} a_{\gamma x^i} X_\gamma X_x^i \\
&+ \textstyle\sum_{i=0}^{n-1} a_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} a_{\gamma\delta x^i} X_\gamma X_\delta X_x^i \\
&+ \sum_{i=0}^{l} a_{s_i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)) + \sum_{i=l+1}^{m} a_{s_i}(X_\gamma^2 w_i(X_x) + 2X_\alpha X_\gamma u_i(X_x)) \\
&+ \sum_{j=1}^{q_{Q_1}} a_{\lambda_j} X_{\lambda_j} + \sum_{j=0}^{q} a_{A_j} X_{\mu_j} \\
&+ \sum_{j=0}^{q} a_{C_j}(X_{\mu_j}^2 - X_\alpha^2 + X_\delta X_{\mu_j}\mathcal{H}(A_j) - X_\gamma \sum_{i=0}^{l} s_{j,i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)))
\end{aligned}$$

$$\begin{aligned}
c(\mathbf{X}) &= c_0 + c_\alpha X_\alpha + c_\delta X_\delta + c_{\alpha\delta} X_\alpha X_\delta + \textstyle\sum_{i=0}^{n-1} c_{\gamma x^i} X_\gamma X_x^i \\
&+ \textstyle\sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} c_{\gamma\delta x^i} X_\gamma X_\delta X_x^i \\
&+ \sum_{i=0}^{l} c_{s_i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)) + \sum_{i=l+1}^{m} c_{s_i}(X_\gamma^2 w_i(X_x) + 2X_\alpha X_\gamma u_i(X_x)) \\
&+ \sum_{j=1}^{q_{Q_1}} c_{\lambda_j} X_{\lambda_j} + \sum_{j=0}^{q} c_{A_j} X_{\mu_j} \\
&+ \sum_{j=0}^{q} c_{C_j}(X_{\mu_j}^2 - X_\alpha^2 + X_\delta X_{\mu_j}\mathcal{H}(A_j) - X_\gamma \sum_{i=0}^{l} s_{j,i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)))
\end{aligned}$$

Then by the verification equation, the following equation should hold.

$$(a(\mathbf{X})+\delta\mathcal{H}(A)) \cdot a(\mathbf{X}) = X_\alpha^2 + X_\gamma \sum_{i=0}^{l} s_i(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)) + c(\mathbf{X}) \qquad (6)$$

We will now show that in order to satisfy the formal polynomials equations above, either the adversary must recycle an instance and a proof, or alternatively a witness is extracted. First, suppose we have some $a_{A_k} \neq 0$. Since there are only X_{μ_k}, $X_{\mu_k}X_\delta$, and $X_{\mu_k}^2$ related with X_{μ_k} and there is no X_δ^2 in the right form, $a(\mathbf{X}) = a_0 + a_{A_k}X_{\mu_k}$. Plugging this into (6) gives us,

$$\begin{aligned}
&(a_0 + a_{A_k}X_{\mu_k} + X_\delta\mathcal{H}(A))(a_0 + a_{A_k}X_{\mu_k}) \\
&= X_\alpha^2 + X_\gamma \sum_{i=0}^{l} a_{s_i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)) + c(\mathbf{X})
\end{aligned}$$

The only way this is possible is by setting

$$c(\mathbf{X}) = c_0 + c_\delta X_\delta + c_{A_k} X_{\mu_k} + c_{C_k}(X_{\mu_k}^2 - X_\alpha^2 + \\ X_\delta X_{\mu_k}\mathcal{H}(A_k) - X_\gamma \sum_{i=0}^{l} s_{k,i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)))$$

Since there is no X_α^2 in the left form, $c_{C_k} = 1$. In addition, since there is $a_{A_k}^2 X_{\mu_k}^2$ in the left form, $a_{A_k}^2 = c_{C_k} = 1$, and $a_{A_k} = 1$ or -1. If we consider $X_\delta X_{\mu_k}$ then $a_{A_k}\mathcal{H}(A)X_\delta X_{\mu_k} = \mathcal{H}(A_k)X_\delta X_{\mu_k}$. Hence $a_{A_k}\mathcal{H}(A) = \mathcal{H}(A_k)$, and $a_{A_k}\mathcal{H}(G^{a_0}A_k^{a_{A_k}}) = \mathcal{H}(A_k)$. Assume that $a_{A_k} = -1$. Let $c = -\mathcal{H}(A_k)$ and $z = G^{a_0}A_k^{a_{A_k}}$. Since A_k is given, c is a given value. The problem is to find a preimage of c such that $H(z) = c$, which is hard for collision resistant hash. Therefore $a_{A_k} = 1$. The problem is to find a_0 such that $\mathcal{H}(G^{a_0}A_k) = \mathcal{H}(A_k)$. Since it is hard to find $G^{a_0}A_k \neq A_k$, $a_0 = 0$. Since $u_i(X_x)_{i=1}^{l}$ are linearly independent, we see for $i = 1, \ldots, l$ that $s_i = s_{k,i}$. In other words, the adversary has recycled the k-th instance $\pi = \pi_k$ and proof $(A, C) = (A_k, C_k)$.

Next, suppose for all $j = 1, \ldots, q$ that $a_{A_j} = 0$. Then $c_{C_j} = c_{A_j} = 0$ since there is no X_{μ_j} in the left form. Since there is X_α^2 in the right form, $a_\alpha^2 = 1$. In the right form, there are only X_α, X_α^2, $X_\alpha X_\gamma$, $X_\alpha X_\delta$, and $X_\alpha u_i(X_x)$ related with X_α and there is no X_δ^2, $a(\mathbf{X}) = a_0 + a_\alpha X_\alpha + \sum_{i=0}^{n-1} a_{\gamma x^i} X_\gamma X_x^i$. We are now left with

$$c(\mathbf{X}) = c_0 + c_\alpha X_\alpha + c_\delta X_\delta + c_{\alpha\delta}X_\alpha X_\delta \\ + \sum_{i=0}^{n-1} c_{\gamma x^i} X_\gamma X_x^i + \sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} c_{\gamma\delta x^i} X_\gamma X_\delta X_x^i \\ + \sum_{i=0}^{l} c_{s_i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)) + \sum_{i=l+1}^{m} c_{s_i}(X_\gamma^2 w_i(X_x) + 2X_\alpha X_\gamma u_i(X_x))$$

In (6),

$$(a_\alpha X_\alpha + a_0 + \sum_{i=0}^{n-1} a_{\gamma x^i} X_\gamma X_x^i + \mathcal{H}(A)X_\delta)(a_\alpha X_\alpha + a_0 + \sum_{i=0}^{n-1} a_{\gamma x^i} X_\gamma X_x^i) \\ = X_\alpha^2 + X_\gamma \sum_{i=0}^{l} a_{s_i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)) \\ + c_0 + c_\alpha X_\alpha + c_\delta X_\delta + c_{\alpha\delta} X_\alpha X_\delta \\ + \sum_{i=0}^{n-1} c_{\gamma x^i} X_\gamma X_x^i + \sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} c_{\gamma\delta x^i} X_\gamma X_\delta X_x^i \\ + \sum_{i=0}^{l} c_{s_i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)) + \sum_{i=l+1}^{m} c_{s_i}(X_\gamma^2 w_i(X_x) + 2X_\alpha X_\gamma u_i(X_x))$$

Define for $i = l+1, \ldots, m$ that $s_i = c_{s_i}$. The terms involving $X_\alpha X_\gamma X_x^i$ now give us $a_\alpha \sum_{i=0}^{n-1} a_{\gamma x^i} X_x^i = \sum_{i=0}^{m} s_i u_i(X_x)$. Finally, the terms involving X_γ^2 produce

$$(X_\gamma \sum_{i=0}^{m} s_i u_i(X_x))^2 = X_\gamma^2 (\sum_{i=0}^{n-1} a_{\gamma x^i} X_x^i)^2 = X_\gamma^2 a_\alpha^2 (\sum_{i=0}^{m} s_i w_i(X_x) + t(X_x) \sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_x^i)$$

Defining $h(X_x) = \sum_{i=0}^{n-1} c_{\gamma^2 tx^i} X_x^i$ we see that this means that $(s_{l+1}, \ldots, s_m)$ is a witness for the instance $(s_1, \ldots, s_l)$ (the extracted witness may be one of many possible valid witnesses).

$\square$

7 Conclusion

In this paper, we propose two simulation-extractable succinct non-interactive arguments of knowledge (SE-SNARK) constructions, which achieve minimal proof size and a single verification. Our first construction is based on the quadratic arithmetic program (QAP) representation, with a proof size of 3 group elements (type III). The other construction is based on the square arithmetic program (SAP) representation, with a proof size of 2 group elements (type I). The security of our schemes are proven under the hash-algebraic knowledge (HAK) assumption and the (linear) collision-resistant hash function.

References

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(7), 1996.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *Bulletproofs: Short Proofs for Confidential Transactions and More*, page 0. IEEE, 2018.

[BBFR15] Michael Backes, Manuel Barbosa, Dario Fiore, and Raphael M Reischuk. Adsnark: nearly practical and privacy-preserving proofs on authenticated data. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 271–286. IEEE, 2015.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Omer Paneth, and Rafail Ostrovsky. Succinct non-interactive arguments via linear interactive proofs. In *Theory of Cryptography*, pages 315–333. Springer, 2013.

[BCPR16] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. *SIAM Journal on Computing*, 45(5):1910–1952, 2016.

[BG18] Sean Bowe and Ariel Gabizon. Making groth's zk-snark simulation extractable in the random oracle model. Cryptology ePrint Archive, Report 2018/187, 2018. https://eprint.iacr.org/2018/187.

[BSCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology–CRYPTO 2013*, pages 90–108. Springer, 2013.

[BSCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX Security Symposium*, pages 781–796, 2014.

[BSCTV17] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. *Algorithmica*, 79(4):1102–1160, 2017.

[CFH$^+$15] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy (SP)*, pages 253–270. IEEE, 2015.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pages 78–96, 2006.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 90–112. ACM, 2012.

[DFKP13] George Danezis, Cedric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio coin: building zerocoin from a succinct pairing-based proof system. In *Proceedings of the First ACM workshop on Language support for privacy-enhancing technologies*, pages 27–30, 2013.

[DLFKP16] Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. Cinderella: Turning shabby x. 509 certificates into elegant anonymous credentials with the magic of verifiable computation. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 235–254. IEEE, 2016.

[FFG$^+$16] Dario Fiore, Cédric Fournet, Esha Ghosh, Markulf Kohlweiss, Olga Ohrimenko, and Bryan Parno. Hash first, argue later: Adaptive verifiable computations on outsourced data. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1304–1316. ACM, 2016.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[GGM16] Christina Garman, Matthew Green, and Ian Miers. Accountable privacy for decentralized anonymous payments. In *International Conference on Financial Cryptography and Data Security*, pages 81–98. Springer, 2016.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 113–122. ACM, 2008.

[GM17] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from Simulation-Extractable SNARKs. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference,* *Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 581–612, 2017.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 321–340, 2010.

[Gro16] Jens Groth. On the size of Pairing-Based non-interactive arguments. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 305–326, 2016.

[Ica09] Thomas Icart. How to hash into elliptic curves. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 303–316, 2009.

[KMS+16] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE symposium on security and privacy (SP)*, pages 839–858. IEEE, 2016.

[KPP+14] Ahmed E Kosba, Dimitrios Papadopoulos, Charalampos Papamanthou, Mahmoud F Sayed, Elaine Shi, and Nikos Triandopoulos. Trueset: Faster verifiable set computations. In *USENIX Security Symposium*, pages 765–780, 2014.

[KPS18] Ahmed Kosba, Charalampos Papamanthou, and Elaine Shi. xjsnark: A framework for efficient verifiable computation. In *xJsnark: A Framework for Efficient Verifiable Computation*, page 0. IEEE, 2018.

[Lip19] Helger Lipmaa. Simulation-extractable snarks revisited. *IACR Cryptology ePrint Archive*, 2019:612, 2019.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 238–252, 2013.

[WJB+17] Riad S Wahby, Ye Ji, Andrew J Blumberg, Abhi Shelat, Justin Thaler, Michael Walfish, and Thomas Wies. Full accounting for verifiable outsourcing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2071–2086. ACM, 2017.

[WTTW18] Riad S Wahby, Ioanna Tzialla, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup. 2018.

[ZGK+18] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vram: Faster verifiable ram with program-independent preprocessing. In *Proceeding of IEEE Symposium on Security and Privacy (S&P)*, 2018.