Updatable CRS Simulation-Extractable zk-SNARKs 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. This paper proposes pairing-based simulation-extractable zero-knowledge succinct non-interactive arguments of knowledge (SE-SNARK) schemes for QAP (Quadratic Arithmetic Program). In the proposed schemes, the proof size is *3 group elements* for a *QAP* (Quadratic Arithmetic Program) circuit in asymmetric groups (Type III pairing), and *2 group elements* for an *SAP* (Square Arithmetic Program) circuit in symmetric groups (Type I pairing), respectively. Moreover, the proposed schemes have only a single verification equation, while all existing SE-SNARK schemes have two verification equations. Compared with the existing state-of-the-art SE-SNARK schemes, the proof size and the number of verification equations are minimal in the proposed scheme. The soundness of the proposed scheme is proven under subversion algebraic knowledge assumptions. Furthermore, we extend the proposed SE-SNARK to support a two-round updatable CRS in which the CRS size remains linear to the circuit size.

Keywords: zk-SNARK, simulation-extractability, quadratic arithmetic program, square arithmetic program, updatable CRS

1 Introduction

As digital privacy becomes more sensitive, the conflict between privacy and legitimacy often sets a barrier for recent real-life applications. One notable example is privacy-preserving blockchain systems. Since the blockchain is well-known to provide robust integrity due to consensus and distribution, it is often considered as an ideal platform for various attestation-required applications such as digital currencies, smart contract, healthcare, supply chains, voting systems, etc. The blockchain integrity provided by finalizing contents and distributing them to all participants, however, raises a privacy issue for the plain data. Alternatively, if the data is encrypted in a block, then it is hard to know whether the data provider is an authorized candidate or whether the data itself was created in a legitimate way. The contradiction between privacy vs. legitimacy leads the privacy-aware blockchain applications to a dead-end[3].

A zero-knowledge proof system acts as a problem-solver to resolve the legitimacy problem of the private data. If the data provider includes a proof related to the legitimacy of the data, the public can verify it without knowing the data. In practice, anonymous blockchain cryptocurrencies such as Zcash [BCG$^+$14], already deploy the zero-knowledge proof system in their applications. The main concern is practicality: the proof generation needs to be non-interactive when the applications target the unlimited, non-specific, public verification and the proof size/verification time is desired to be scalable regardless of the complexity of the legitimacy.

In the recent history of zero-knowledge proofs, zk-SNARK (zero-knowledge succinct non-interactive arguments of knowledge) have drawn significant attention for its efficiency and theoretical advances. They enable a prover to generate a proof for any NP statements in a manner where the proof is zero-knowledge about its witness and the proof size and the verification cost are succinct. For succinctness, it is often accepted if the size and the verifying computation are logarithmic to the circuit size. Thus the zk-SNARK terminology embraces various types of zero-knowledge proof systems, such as ZKBoo [GMO16] and vRAM [ZGK$^+$18] which are an advanced from of traditional interaction-based proof systems with Fiat-Shamir transformation [FS86].

However, when applied to a massive public infrastructure such as blockchain, a logarithmic (sublinear) size might not be enough for succinctness. For example in zerocash [BCG$^+$14], the membership test circuit has 64 hash functions (approximately 29,000 lines for each hash) which leads to a single proof size of 5MB by rough estimation in ZKBoo [GMO16][4]. Considering that innumerable transactions, each including a proof are distributed to the participants, a proof size of 5MB seems inadmissible as practical.

Therefore, for scalability, it is desirable to adopt zk-SNARK with a *constant* size proof and verification, which is constructed in the paring-based elliptic curve group and Quadratic Arithmetic Program (QAP) [GGPR13]. In the QAP-based SNARKs such as [Gro16], by utilizing polynomial relations, a proof contains 3 group elements and the verification requires 3 pairings regardless of the circuit size. When this scheme is applied to the Zcash, the proof size becomes 60 bytes and verification takes 100ms. Consequently, we focus on the literature of QAP-based (and pairing-based) zk-SNARKs with a constant-sized proof and verification for the rest of the paper. Hereafter, we often use the term zk-SNARK or SNARK mixed with the "QAP-based (and pairing-based) zk-SNARK".

[3] There still are alternative solutions, such as setting a trusted manager or delicately narrowing down the blockchain data contents. However it is often complicated and does not solve the fundamental controversy.

[4] In ZKBoo, the experiment results show that the proof size is 835.91KB for a SHA-256 hash function. We multiply it by 6 (=log(64)) to estimate the 64 sequential executions of hash functions.

Despite the practical functionality, a weakness of zk-SNARK is that they are susceptible to man-in-the-middle attacks. Namely, an adversary who obtains valid proofs could forge a new valid proof without knowing the witness. In consequence, the zk-SNARK's implementation often requires an additional protection method against its malleability. The zerocash [BCG+14], for example, combines one-time signatures within the zk-SNARK circuit.

Groth and Maller [GM17] tackle the malleability problem of existing SNARKs, define the notion of *simulation-extractable*-SNARK (SE-SNARK) which indicates non-malleable zk-SNARK, and propose the corresponding scheme called SE-SNARK. SE-SNARK [GM17] resolves the malleability issue by adopting SAP (Square Arithmetic Program) instead of more general QAP and applying an additional verification equation to existing the state-of-art SNARK [Gro16]. [GM17] also proves that any SE-SNARK scheme in the NILP frame [BCI+13] necessarily requires at least 3 elements in a proof and 2 equations in the verification [GM17]. In this sense, the pairing-based SE-SNARK scheme in [GM17] is optimal in the proof size and the number of verification equations: 3 elements for a proof and 2 equations for verification, however, by sacrificing the circuit representation from the QAP to the SAP.

In this paper, we take the approach to employ the hash function and achieve better results of 3 proof elements and a *single* verification equation. While 3 elements in a proof and 2 verification equations are optimally required at least in a pairing-based SE-SNARK in the NILP frame, it is not known whether further optimization is possible in the non-NILP frame. In fact, the use of a hash function that deviates from the NILP frame to bind elements offers new possibilities to break the existing boundaries. Moreover, the resulting scheme works on the efficient QAP circuit.

Another crucial property for the SNARK is trustable CRS construction. In the SNARK, unless CRS is generated in a trusted way, the trapdoor using in CRS can generate fake proofs for false instances. To solve the trusted setup problem, Zcash [BGG18] builds CRS using secure multi-party computation to distribute the trapdoor information. [GKM+18] introduces the updatable CRS model, in which any user can update CRS at any point. However the CRS size is quadratic to the relation size. To make the size of updatable CRS to linear to the relation size, [MBKM19] proposes a linear size universal and updatable CRS SNARK scheme with sacrificing the proof size and the performance. To support updatable CRS in SNARK, this paper adopts a two round update framework in which universal CRS is updated at the first round and circuit dependent CRS is updated at the second round after building the circuit dependent CRS from a given relation using the universal CRS. The proposed framework is applicable to many existing SNARK schemes. Especially, this paper extends the proposed SE-SNARK to support the two round CRS update approach. In the proposed updatable CRS SE-SNARK, the CRS size is still linear to the relation size without increasing the proof size and the proof time.

Our contributions. In this paper, we first construct a QAP-based SE-SNARK scheme with a single verifying equation in an asymmetric group (Type III pairing). Given three groups with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, our proofs consists of only 3 group elements from the source groups: two from $\mathbb{G}_1$ and one from $\mathbb{G}_2$. Additionally, we also propose a SAP-based SE-SNARK with 2 elements for a proof and a single verifying equation, in a symmetric group (Type I pairing). Finally we extend the proposed QAP-based SE-SNARK to support updatable CRS. We summarize our contributions as follows:

– **QAP-based SE-SNARK with 3 elements (Type III)**
  We propose a first pairing-based SE-SNARK that utilizes QAP circuits, instead of SAP, while maintaining 3 elements for a proof. Note that the SAP circuit size is theoretically double of the QAP circuit size.

– **SAP-based SE-SNARK with 2 elements (Type I)**
  We show that our construction can reduce the number of proof elements to 2 (with utilizing SAP) in symmetric pairing (Type I). Note that this result surpasses the theoretical boundary for SE-SNARKs proven in [GM17].

– **Single verifying equation**
  Our SE-SNARK construction verifies the proof with a *single* verifying equation. By utilizing the hash function to bind the unique proof tuple (A, B, C), we eliminate the additional equation for the malleability check.

– **Updatable CRS**
  Our SE-SNARK schemes are extended to support updatable CRS. The CRS updation is performed in two rounds. At the first round, circuit independent CRS is updated. A circuit dependent CRS for a given relation is constructed from the circuit independent CRS updated. At the second round, the circuit dependent CRS is updated. In the two-round update framework, the CRS size still remains linearly to the circuit size, and the proof size and the performance do not increase.

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 sub-linear 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.

Table 1 summarizes and compares the overall size and performance of our QAP-based SE-SNARK with the state-of-the-art zk-SNARK [Gro16] and SE-SNARK schemes [GM17,BG18,Lip19].

Orthogonal to the simulation-extractability, a zk-SNARK with updatable CRS solves the trust issue of CRS by letting the users independently update the CRS [GKM$^+$18,MBKM19]. The traditional limitation of SNARKs is that they all require trusted CRS generation. Through the updating approach, users who distrust a current CRS can rely on self-updating. Although two approaches proposed in [GKM$^+$18,MBKM19] allow a single round update, the CRS size is quadratic to the circuit size [GKM$^+$18] or the proof size increases to $5\mathbb{G} + 7\mathbb{F}$ or $20\mathbb{G} + 16\mathbb{F}$ where $\mathbb{G}$ and $\mathbb{F}$ denote the group elements and the field elements, respectively.

In this paper, we focus on the simulation-extractable SNARKs, specifically pairing-based SE-SNARKs. Similarly to [Lip19], our scheme is secure under SAK

Table 1: Comparison for arithmetic circuit satisfiability with l element instance, m wires, 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	$\|CRS\|$	$\|\pi\|$	**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	SAK
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	SAK
Ours	SAP	$(2m+6n)\mathbb{G}$	$2\mathbb{G}$	$(2m+6n)E$	$lE_1 + 3P$	1	SAK

assumption and collision resistant hash, and the proof size is further reduced to 3 group elements and a single verification is required which are equal to [Gro16]. Moreover, we extend the proposed schemes to support updatable CRS.

The rest of the paper proceeds as follows: Section 2 provides some necessary notions and backgrounds; Section 3 defines a bilinear group and assumptions; in Section 4, we present our QAP-based SE-SNARK with a single verification; in Section 5, we propose a symmetric SAP-based SE-SNARK with 2 proof elements; Section 6 extends the schemes to support updatable CRS. Section 7 draws a conclusion.

2 Preliminaries

2.1 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.

2.2 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.

2.3 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.

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

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

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.

2.4 Updating common reference strings

We define a two-round updatable CRS scheme consisting of nine PPT algorithms Setup, Update_x, $\mathsf{VerifyCRS}_x$, Derive, Update_R, $\mathsf{VerifyCRS}_R$, Prove, Vfy, $\mathsf{SimProve}$. Three algorithms of Prove, Vfy, $\mathsf{SimProve}$ are equivalent to the components in zk-SNARK *Arg*.

- $(\tau_x, crs_x, \rho_1) \leftarrow \mathsf{Setup}(1^\lambda)$ receives the security parameter as input and returns a simulation trapdoor, a relation independent CRS and a proof of correctness.
- $(\tau'_x, crs'_x, \rho_{k+1}) \leftarrow \mathsf{Update}_x(1^\lambda, crs_x, \{\rho_j\}_{j=1}^{k})$ receives relation independent CRS and a list of update proofs as input and returns an updated simulation trapdoor, an updated CRS and a proof of the correctness of the update.
- $\mathsf{VerifyCRS}_x(1^\lambda, crs_x, \{\rho_j\}_{j=1}^{n_1})$ receives the security parameter, relation independent CRS and a list of update proofs and returns whether it accepts or not.
- $(\tau_R, crs_R, \phi_1) \leftarrow \mathsf{Derive}(1^\lambda, crs_x, R)$ receives the security parameter, relation independent CRS and relation as input and returns a simulation trapdoor, a relation dependent CRS and a proof of correctness.
- $(\tau'_R, crs'_R, \phi_{k+1}) \leftarrow \mathsf{Update}_R(1^\lambda, crs_R, \{\phi\}_{i=1}^{k})$ receives the security parameter, relation dependent CRS and a list of update proofs as input and returns an updated simulation trapdoor, an updated CRS and a proof of the correctness of the update.
- $\mathsf{VerifyCRS}_R(1^\lambda, crs_R, \{\phi_j\}_{j=1}^{n_2})$ receives the security parameter, relation dependent CRS and a list of update proofs and returns whether it accepts or not.

[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.

Definition 3. *An updatable CRS scheme is perfectly correct if*

$$Pr[(crs_x, \rho) \leftarrow \mathsf{Setup}(1^\lambda) : \mathsf{VerifyCRS}_x(1^\lambda, crs_x, \rho) = 1] = 1;$$
$$Pr\begin{bmatrix}(crs'_x, \rho_{k+1}) \leftarrow \mathsf{Update}_x(1^\lambda, crs_x, \{\rho_i\}_{i=1}^k) : \\ \mathsf{VerifyCRS}_x(1^\lambda, crs_x, \{\rho_i\}_{i=1}^k) = 1 \wedge \mathsf{VerifyCRS}_x(1^\lambda, crs'_x, \{\rho_i\}_{i=1}^{k+1}) = 1\end{bmatrix} = 1;$$
$$Pr[(crs_R, \phi) \leftarrow \mathsf{Derive}(1^\lambda, crs_x, R) : \mathsf{VerifyCRS}_R(1^\lambda, crs_R, \phi) = 1] = 1;$$
$$Pr\begin{bmatrix}(crs'_R, \phi_{k+1}) \leftarrow \mathsf{Update}_R(1^\lambda, crs_R, \{\phi_i\}_{i=1}^k) : \\ \mathsf{VerifyCRS}_R(1^\lambda, crs_R, \{\phi_i\}_{i=1}^k) = 1 \wedge \mathsf{VerifyCRS}_R(1^\lambda, crs'_R, \{\phi_i\}_{i=1}^{k+1}) = 1\end{bmatrix} = 1$$

3 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 $\mathbb{G}_T$).

3.1 Power Knowledge of Exponent Assumption

We define q-power knowledge of exponent assumption for updatable CRS SE-SNARK schemes. Using this assumption, we will show that trapdoors are extractable in the proposed scheme.

Definition 4 (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*

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

3.2 Subversion Algebraic Knowledge Assumption

Lipmaa proposes a new knowledge assumption called subversion algebraic knowledge (SAK) assumption [Lip19]. 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 $G^{\mathbf{x}}$ be a vector of G^{x_i}. Similarly, let $G^{\mathbf{y}}$ be a vector 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 subversion-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 $\omega(\log \lambda)$, $\mathbf{Adv}^{sak}_{\mathbb{G}_1,\mathcal{D},\mathcal{D}',\mathcal{A}}(\lambda) :=$

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

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

Definition 5 ($\mathcal{D} - SAK$ 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, $\mathbf{Adv}^{sak}_{\mathbb{G}_\iota,\mathcal{D},\mathcal{D}',\mathcal{A}}(\lambda) \approx 0$ for each PPT adversary $\mathcal{A}$ and each distribution $\mathcal{D}'$ of min-entropy $\omega(\log \lambda)$.*

3.3 Linear Collision Resistant Hash Function

The second intractable assumption is the extended collision resistant hash function. The conventional collision resistant hash function is defined as following:

Definition 6 (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 SHA. Hence, specifically for our purpose, we define an extended collision resistant hash function called *linear collision resistant* (linear CR) hash function. It is assumed that the proposed hash receives a group element as input. In the proposed hash function, it is hard to find non-trivial $a, a' \in \mathbb{Z}_p$ and $x, x' \in \mathbb{G}$ where $\mathbb{G}$ is a cyclic group of prime order p such that $\mathcal{H}(x^a x'^{a'}) = a + a'\mathcal{H}(x')$. Formally, it is defined as follows:

Definition 7 (Linear collision resistance). *$\mathcal{H} : \mathbb{G} \to \mathbb{Z}_p$ is a linear collision resistant hash function if for all PPT adversary $\mathcal{A}$,* $\mathbf{Adv}_{\mathcal{H}}^{LCR}(\mathcal{A}) :=$

$$Pr\left[\begin{array}{c}(x, x', a, a') \leftarrow \mathcal{A}(\mathbb{G}, \mathcal{H}) : \\ (a \neq 0) \wedge (a' \neq 1) \wedge (x \neq 1_{\mathbb{G}}) \wedge (\mathcal{H}(x^a x'^{a'}) = a + a'\mathcal{H}(x'))\end{array}\right] \approx 0$$

Note that $1_{\mathbb{G}}$ denotes the identity in $\mathbb{G}$. Although $\mathcal{H}$ receives a group element as input and outputs an element in $\mathbb{Z}_p$, $\mathcal{H}$ can receive input and output elements as strings if the output string is remapped in $\mathbb{Z}_p$. Hence, many cryptographical hash functions such as SHA, and Ajtai hash [Ajt96] which are used as collision resistant hash can be also adopted as a linear collision resistant hash.

4 QAP-based SE-SNARK Scheme

4.1 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 in [Gro16]. Then

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

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 $e(AG^{\mathcal{H}(A)}, BH^{\delta\mathcal{H}(B)})$, 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.

4.2 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.

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

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

4.3 Construction

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

$$\tau = (G, H, \alpha, \beta, \gamma, \delta, x)$$
$$crs = \begin{pmatrix} R, H_1, H_2, G, G^{\alpha}, G^{\beta}, G^{\delta}, G^{\alpha\delta}, H, H^{\beta}, H^{\delta} \\ \{G^{\gamma x^i}, H^{\gamma x^i}, G^{\gamma^2 t(x) x^i}, G^{\gamma \delta x^i}\}_{i=0}^{n-1}, \{G^{\gamma w_i(x) + \beta u_i(x) + \alpha v_i(x)}\}_{i=0}^{l}, \\ \{G^{\gamma^2 w_i(x) + \beta\gamma u_i(x) + \alpha\gamma v_i(x)}\}_{i=l+1}^{m} \end{pmatrix}$$

- $\pi \leftarrow \mathsf{Prove}(crs, \phi, w)$: 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 $\mu, \nu \leftarrow \mathbb{Z}_p$ and compute $\pi = (A, B, C)$ such that

$$\begin{aligned} &A = G^{\mu}, B = H^{\nu}, \\ &C = G^{\mu\nu - \alpha\beta + h_2\delta\mu + h_1\nu + h_1h_2\delta - \gamma\sum_{i=0}^{l} s_i(\gamma w_i(x) + \beta u_i(x) + \alpha v_i(x))} \end{aligned} \tag{4}$$

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

4.4 Security Proof

Theorem 1. *The protocol given above 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 SAK (subversion 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

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

Now, the proof elements can be computed as follows:

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

$$B = H^{\beta} \prod_{j=0}^{n-1} (H^{\gamma x^j}) \cdot H^s$$

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

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

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

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

Note that since the vector $(s_{l+1},\dots,s_m)$ is a valid witness for the instance $(s_1,\dots,s_l)$, $(\sum_{i=0}^{m} s_i u_i(X))(\sum_{i=0}^{m} s_i v_i(X)) = \sum_{i=0}^{m} s_i w_i(X) + h(X)t(X)$ 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, $X_x, X_{\mu_1}, X_{\nu_1}, \dots, X_{\mu_q}, X_{\nu_q}$ where we plug in randomly generated $\mu_1, \nu_1, \dots, \mu_q, \nu_q$ for the latter ones.

By $D - SAK$, given a proof $\pi = (G^a, H^b, H^c)$, 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:

$$
\begin{aligned}
a(\mathbf{X}) &= a_0 + a_\alpha X_\alpha + a_\beta X_\beta + a_\delta X_\delta + a_{\alpha\delta} X_\alpha X_\delta + \sum\nolimits_{i=0}^{n-1} a_{\gamma x^i} X_\gamma X_x^i \\
&+ \sum\nolimits_{i=0}^{n-1} a_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} a_{\gamma\delta x^i} X_\gamma X_\delta X_x^i \\
&+ \sum_{i=0}^{l} a_{s_i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) \\
&+ \sum_{i=l+1}^{m} a_{s_i} (X_\gamma^2 w_i(X_x) + X_\beta X_\gamma u_i(X_x) + X_\alpha X_\gamma v_i(X_x)) + \sum_{j=1}^{q_{Q_1}} a_{\lambda_j} X_{\lambda_j} + \sum_{j=1}^{q} a_{A_j} X_{\mu_j} \\
&+ \sum_{j=1}^{q} a_{C_j} (X_{\mu_j} X_{\nu_j} - X_\alpha X_\beta - X_\gamma \sum_{i=0}^{l} s_{j,i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) \\
&+ h_2 X_\delta X_{\mu_j} + h_1 X_{\nu_1} + h_1 h_2 X_\delta)
\end{aligned}
$$

$$b(\mathbf{X}) = b_0 + b_\beta X_\beta + b_\delta X_\delta + \sum\nolimits_{i=0}^{n-1} b_{\gamma x^i} X_\gamma X_x^i + \sum_{j=1}^{q_{Q_2}} b_{\rho_j} X_{\rho_j} + \sum_{j=1}^{q} b_{B_j} X_{\nu_j}$$

$$c(\mathbf{X}) = c_0 + c_\alpha X_\alpha + c_\beta X_\beta + c_\delta X_\delta + c_{\alpha\delta} X_\alpha X_\delta + \sum\nolimits_{i=0}^{n-1} c_{\gamma x^i} X_\gamma X_x^i$$

$$+ \sum\nolimits_{i=0}^{n-1} c_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} c_{\gamma\delta x^i} X_\gamma X_\delta X_x^i$$

$$+ \sum_{i=0}^{l} c_{s_i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x))$$

$$+ \sum_{i=l+1}^{m} c_{s_i} (X_\gamma^2 w_i(X_x) + X_\beta X_\gamma u_i(X_x) + X_\alpha X_\gamma v_i(X_x)) + \sum_{j=1}^{q_{Q_1}} c_{\lambda_j} X_{\lambda_j} + \sum_{j=1}^{q} c_{A_j} X_{\mu_j}$$

$$+ \sum_{j=1}^{q} c_{C_j} (X_{\mu_j} X_{\nu_j} - X_\alpha X_\beta - X_\gamma \sum_{i=0}^{l} s_{j,i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x))$$

$$+ h_2 X_\delta X_{\mu_j} + h_1 X_{\nu_j} + h_1 h_2 X_\delta)$$

$a(\mathbf{X}), b(\mathbf{X})$, 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.

$$
\begin{aligned}
&(a_0 + a_{\mu_k} X_{\mu_k} + h'_1)(b_0 + b_\delta X_\delta + b_{\nu_k} X_{\nu_k} + h'_2 X_\delta) \\
&= c_0 + c_{A_k} X_{\mu_k} + X_{\mu_k} X_{\nu_k} + h_2 X_\delta X_{\mu_k} + h_1 X_{\nu_k} + h_1 h_2 X_\delta \\
&- X_\gamma \sum_{i=0}^{l} s_{k,i}(X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x))
\end{aligned}
$$

Since $a_{\mu_k} b_{\nu_k} = 1$, there is $(a_0 + h'_1) b_{\nu_k} X_{\nu_k}$ in the left form, and there is $h_1 X_{\nu_k}$ in the right form, $(a_0 + h'_1) b_{\nu_k} = h_1$, $h'_1 = -a_0 + a_{\mu_k} h_1$, and $\mathcal{H}(G^{-(-a_0)} A_k^{a_{\mu_k}}) = -a_0 + a_{\mu_k} \mathcal{H}(A_k)$. Since $\mathcal{H}$ is linear collision resistant, it is hard to find non trivial $-a_0$ and a_{μ_k}. Hence $a_0 = 0$, and $a_{\mu_k} = 1$. Similarly, Since there is $(b_\delta + h'_2) a_{\mu_k} X_\delta X_{\mu_k}$ in the left form, and there is $h_2 X_\delta X_{\mu_k}$ in the right form, $(b_\delta + h'_2) a_{\mu_k} = h_2$, $h'_2 = -b_\delta + b_{\nu_k} h_2$, and $\mathcal{H}(H^{b_0} H^{-(-b_\delta)} B_k^{b_{\nu_k}}) = -b_\delta + b_{\nu_k} \mathcal{H}(B_k)$. Since $\mathcal{H}$ is collision resistant, it is hard to find non trivial b_0 such that $\mathcal{H}(H^{b_0} H^{b_\delta} B_k^{b_{\nu_k}}) = \mathcal{H}(H^{b_\delta} B_k^{b_{\nu_k}})$. Hence $b_0 = 0$. In addition, since $\mathcal{H}$ is linear collision resistant, it is hard to find non trivial $-b_\delta$ and b_{ν_k} satisfying $\mathcal{H}(H^{-(-b_\delta)} B_k^{b_{\nu_k}}) = -b_\delta + b_{\nu_k} \mathcal{H}(B_k)$. Hence $b_\delta = 0$, and $b_{\nu_k} = 1$.

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 $a_{A_j} = b_{B_j} = 0$. 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

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

In (5),

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

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

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

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).

5 SAP-based SE-SNARK Scheme

5.1 Two Group Elements SE-SNARK

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.

5.2 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 $\mathbb{Z}_p$, $1 \leq l \leq m$, 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.

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

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

5.3 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 $G \xleftarrow{\$} \mathbb{G}$, hash functions $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p$, and parameters $\alpha, \gamma, \delta, x \xleftarrow{\$} \mathbb{Z}_p$, such that $t(x) \neq 0$, 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

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

Theorem 2. *The protocol given above 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 $D-SAK$ assumption holds and a collision resistant hash function exists.*

6 Extension to updatable CRS SE-SNARK

In this section, we extend the proposed SE-SNARK schemes to updatable common reference string simulation-extractable subversion-resistant SNARK schemes. In the proposed scheme, the CRS size is linear to the relation size and the proof size is 3 group elements. The proposed approach devises the two-round CRS update. In the two-round CRS update, universal CRS which consists of monomials of x^i is updated in the first round. And then relation dependent CRS_R for relation R is generated. In the second update round, the other trapdoor variables except x are updated in CRS_R. For every update, a update proof is generated to ensure the CRS update with a new trapdoor. In security analysis, we prove that our approach can extract all trapdoor contributions from update proofs, even if CRS size changes after updating, which has not been considered in the existing updatable CRS scheme [GKM+18]. We describe the updatable CRS scheme by extending the QAP SE-SNARK scheme in section 4. Note that the proposed framework can be easily applied to the SAP SE-SNARK in section 5 although it is omitted.

6.1 Construction

- $(\tau_x, crs_x, \rho_1) \leftarrow \mathsf{Setup}(1^\lambda)$: Select generators $G, \hat{G} \leftarrow \mathbb{G}_1, H \leftarrow \mathbb{G}_2$, and parameter $x \in \mathbb{Z}_p$ and set

$$\begin{aligned} \tau_x &= (x) \\ crs_x &= \Big(G, H, \{G^{x^i}, H^{x^i}\}_{i=0}^{n-1} \quad \Big) \\ \rho_1 &= (\hat{G}^x, \hat{G}^x, H^x) \end{aligned}$$

- $(\tau'_x, crs'_x, \rho_{k+1}) \leftarrow \mathsf{Update}_x(1^\lambda, crs_x, \{\rho_j\}_{j=1}^k)$: Select $x' \in \mathbb{Z}_p$ randomly and compute

$$\begin{aligned} \tau'_x &= (xx') \\ crs'_x &= \Big(G, H, \{(G^{x^i})^{x'^i}, (H^{x^i})^{x'^i}\}_{i=0}^{n-1} \quad \Big) \\ \rho_{k+1} &= (\hat{G}^{xx'}, \hat{G}^{x'}, H^{x'}) \end{aligned}$$

- $\mathsf{VerifyCRS}_x(1^\lambda, crs_x, \{\rho_j\}_{j=1}^{n_1})$: Parse

$$crs_x = \Big(G, H, \{G^{x^i}\}_{i=1}^{2n-1}, \{H^{x^i}\}_{i=1}^{n-1} \quad \Big)$$

  and parse $\{\rho_j\}_{j=1}^{n_1} = \{(\bar{G}_j, \hat{G}_j, H_j)\}_{j=1}^{n_1}$.
  Assert the proofs are correct for $1 \le j \le n_1$:

$$\begin{aligned} e(\hat{G}_j, H) &= e(\hat{G}, H_j) \\ e(\bar{G}_j, H) &= e(\bar{G}_{j-1}, H_j) \end{aligned}$$

  where $\bar{G}_0 = G$. Assert the proofs are correct:

$$\begin{aligned} &e(\hat{G}, H^x) = e(\bar{G}_{n_1}, H) \\ &\{e(G^{x^i}, H) = e(G^{x^{i-1}}, H^x)\}_{i=2}^{2n-1} \\ &\{e(G, H^{x^i}) = e(G^{x^i}, H)\}_{i=1}^{n-1} \end{aligned}$$

- $(\tau_R, crs_R, \phi_1) \leftarrow \mathsf{Derive}(1^\lambda, crs_x, R)$: Choose randomly parameters $\alpha, \beta, \gamma, \delta \in \mathbb{Z}_p$ where $t(x) \neq 0$, and set

$$\tau_R = (\alpha, \beta, \gamma, \delta, x)$$

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

$$\phi_1 = (G^{\alpha}, G^{\beta}, G^{\gamma}, G^{\delta}, \hat{G}^{\alpha}, \hat{G}^{\beta}, \hat{G}^{\gamma}, \hat{G}^{\delta},$$
$$\hat{G}^{\alpha}, \hat{G}^{\beta}, \hat{G}^{\gamma}, \hat{G}^{\delta}, H^{\alpha}, H^{\beta}, H^{\gamma}, H^{\delta})$$

– $(\tau'_R, crs'_R, \phi_{k+1}) \leftarrow \mathsf{Update}_R(1^\lambda, crs_R, \{\phi\}_{i=1}^{k})$: Select $\alpha', \beta', \gamma', \delta' \in \mathbb{Z}_p$ randomly and compute

$$\tau'_R = (\alpha + \alpha', \beta + \beta', \gamma + \gamma', \delta + \delta', x)$$

$$crs'_R = \begin{pmatrix} R, G, G^{\alpha}G^{\alpha'}, G^{\beta}G^{\beta'}, G^{\alpha\delta}G^{\alpha\delta'}G^{\delta\alpha'}G^{\alpha'\delta'}, H, H^{\beta}H^{\beta'}, H^{\delta}H^{\delta'} \\ \{G^{x^i}, G^{\gamma x^i}G^{x^i\gamma'}, G^{\gamma^2 t(x)x^i}G^{\gamma t(x)x^i 2\gamma'}G^{t(x)x^i\gamma'^2} \\ G^{\gamma\delta x^i}G^{\delta x^i\gamma'}G^{\gamma x^i\delta'}G^{x^i\gamma'\delta'}, G^{\delta x^i}G^{x^i\delta'}, H^{x^i}, H^{\gamma x^i}H^{x^i\gamma'}\}_{i=0}^{n-1}, \\ \{G^{\gamma w_i(x)+\beta u_i(x)+\alpha v_i(x)}G^{w_i(x)\gamma'}G^{u_i(x)\beta'}G^{v_i(x)\alpha'}\}_{i=0}^{l}, \\ \{G^{\gamma^2 w_i(x)+\beta\gamma u_i(x)+\alpha\gamma v_i(x)}G^{2w_i(x)\gamma\gamma'}G^{w_i(x)\gamma'^2} \\ G^{\gamma u_i(x)\beta'}G^{\beta u_i(x)\gamma'}G^{u_i(x)\beta'\gamma'}G^{\gamma v_i(x)\alpha'}G^{\alpha v_i(x)\gamma'}G^{v_i(x)\alpha'\gamma'}, \\ G^{\beta u_i(x)}G^{u_i(x)\beta'}, G^{\alpha v_i(x)}G^{v_i(x)\alpha'}\}_{i=l+1}^{m} \end{pmatrix}$$

$$\phi_{k+1} = (G^{\alpha+\alpha'}, G^{\beta+\beta'}, G^{\gamma+\gamma'}, G^{\delta+\delta'}, \hat{G}^{\alpha\alpha'}, \hat{G}^{\beta\beta'}, \hat{G}^{\gamma\gamma'}, \hat{G}^{\delta\delta'},$$
$$\hat{G}^{\alpha'}, \hat{G}^{\beta'}, \hat{G}^{\gamma'}, \hat{G}^{\delta'}, H^{\alpha'}, H^{\beta'}, H^{\gamma'}, H^{\delta'})$$

Note that $\alpha, \beta, \gamma, \delta$ are updated as $\alpha + \alpha', \beta + \beta', \gamma + \gamma', \delta + \delta'$ by the above update.

– $\mathsf{VerifyCRS}_R(1^\lambda, crs_R, \{\phi_j\}_{j=1}^{n_2})$: Let $Y = \{\alpha, \beta, \gamma, \delta\}$. Parse crs_R and parse $\phi_j = (\{G_{y,j}\}_{y\in Y}, \{\bar{G}_{y,j}\}_{y\in Y}, \{\hat{G}_{y,j}\}_{y\in Y}, \{H_{y,j}\}_{y\in Y})$.

Let $\bar{G}_{y,0} = 1$ for $y \in Y$. Assert the proofs are correct for $1 \le j \le n_2$, $y \in Y$:

$$e(G_{y,j}/G_{y,j-1}, H) = e(G, H_{y,j})$$
$$e(\bar{G}_{y,j}, H) = e(\bar{G}_{y,j-1}, H_{y,j})$$
$$e(\hat{G}_{y,j}, H) = e(\hat{G}, H_{y,j})$$

Let $\{G^y = G_{y,n_2}\}_{y \in Y}$. Assert the proofs are correct for $1 \leq i \leq n$:

$$\begin{aligned}
&e(G^{\beta}, H) = e(G, H^{\beta})\\
&e(G^{\delta}, H) = e(G, H^{\delta})\\
&e(G^{\alpha\delta}, H) = e(G^{\alpha}, H^{\delta})\\
&e(G^{\gamma x^i}, H) = e(G, H^{\gamma x^i}) = e(G^{\gamma}, H^{x^i})\\
&e(G^{\gamma^2 t(x) x^i}, H) = e(G^{\gamma}, H^{\gamma t(x) x^i})\\
&e(G^{\gamma\delta x^i}, H) = e(G^{\delta}, H^{\gamma x^i})\\
&e(G^{\delta x^i}, H) = e(G^{\delta}, H^{x^i})
\end{aligned}$$

Assert the proofs are correct for $0 \leq i \leq l$

$$e(G^{\gamma w_i(x)+\beta u_i(x)+\alpha v_i(x)}, H) = e(G^{\gamma}, H^{w_i(x)})e(G^{\beta}, H^{u_i(x)}), e(G^{\alpha}, H^{v_i(x)})$$

Assert the proofs are correct for $l+1 \leq i \leq m$

$$\begin{aligned}
&e(G^{\gamma^2 w_i(x)+\beta\gamma u_i(x)+\alpha\gamma v_i(x)}, H) = e(G^{\gamma}, H^{\gamma w_i(x)})e(G^{\beta}, H^{\gamma u_i(x)})e(G^{\alpha}, H^{\gamma v_i(x)})\\
&e(G^{\beta u_i(x)}, H) = e(G^{\beta}, H^{u_i(x)})\\
&e(G^{\alpha v_i(x)}, H) = e(G^{\alpha}, H^{v_i(x)})
\end{aligned}$$

– $\pi \leftarrow \mathsf{Prove}(crs_R, \phi, w)$: 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 $r, s \overset{\$}{\leftarrow} \mathbb{Z}_p$ and compute $\pi = (A, B, C) = (G^a, H^b, G^c)$ such that

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

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

– $0/1 \leftarrow \mathsf{Vfy}(crs_R, \phi, \pi)$: 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:

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

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

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

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

Note that Prove, Vfy, $\mathsf{SimProve}$ are equivalent to the QAP SE-SNARK scheme in section 4.

6.2 Security Proof

Lemma 1 (Correctness of the CRS generation). *The scheme is perfectly correct in the sense that*

$$Pr[(crs_x, \rho) \leftarrow \mathsf{Setup}(1^{\lambda}) : \mathsf{VerifyCRS}_x(1^{\lambda}, crs_x, \rho) = 1] = 1;$$

$$Pr\begin{bmatrix}(crs'_x, \rho_{k+1}) \leftarrow \mathsf{Update}_x(1^{\lambda}, crs_x, \{\rho_i\}_{i=1}^{k}) : \\ \mathsf{VerifyCRS}_x(1^{\lambda}, crs_x, \{\rho_i\}_{i=1}^{k}) = 1 \wedge \mathsf{VerifyCRS}_x(1^{\lambda}, crs'_x, \{\rho_i\}_{i=1}^{k+1}) = 1\end{bmatrix} = 1$$

$$Pr[(crs_R, \phi) \leftarrow \mathsf{Derive}(1^{\lambda}, crs_x, R) : \mathsf{VerifyCRS}_R(1^{\lambda}, crs_R, \phi) = 1] = 1;$$

$$Pr\begin{bmatrix}(crs'_R, \phi_{k+1}) \leftarrow \mathsf{Update}_R(1^{\lambda}, crs_R, \{\phi_i\}_{i=1}^{k}) : \\ \mathsf{VerifyCRS}_R(1^{\lambda}, crs_R, \{\phi_i\}_{i=1}^{k}) = 1 \wedge \mathsf{VerifyCRS}_R(1^{\lambda}, crs'_R, \{\phi_i\}_{i=1}^{k+1}) = 1\end{bmatrix} = 1$$

Proof. The correctness is trivial for crs_x since $\bar{G} = G^x$ in ρ where x is the multiplication of all trapdoors. For crs_R, the correctness is also trivial. In $\phi_k = (G^{\alpha}, \dots)$, α is the sum of all trapdoors related with α. Hence $e(G_{\alpha,j}/G_{\alpha,j-1}, H) = e(G_{\alpha,j-1}G^{\alpha_j}/G_{\alpha,j-1}, H) = e(G^{\alpha_j}, H) = e(G, H^{\alpha_j}) = e(G, H_{\alpha,j})$. $e(\bar{G}_{\alpha,j}, H) = e(\bar{G}_{\alpha,j-1}^{\alpha_j}, H) = e(\bar{G}_{\alpha,j-1}, H^{\alpha_j})$. Similarly, we can check the correctness for β, γ, δ.

In the proof ρ of CRS update, $\hat{G}$ which is not used in CRS is essential to extract trapdoor parameters when CRS size changes after updating. For instance, assume that proof ρ_i is constructed using G rather than $\hat{G}$. Given $crs_i = (G^x, G^{x^2}, G^{x^3}, H^x, H^{x^2}, H^{x^3}, \dots)$, suppose that a new trapdoor is $x' = (\eta_0 + \eta_1 x^2)$ and updated CRS $crs_{i+1} = (G^{x(\eta_0+\eta_1 x^2)}, H^{x(\eta_0+\eta_1 x^2)}, \dots)$, in which the size of crs_{i+1} is smaller than that of crs_i. For given CRS update proofs $\rho_i = (G^x, G^x, H^x)$ and $\rho_{i+1} = (G^{xx'}, G^{x'}, H^{x'})$, an extractor can extract x' if G^{x^2} and H^{x^2} are given when $x' = (\eta_0 + \eta_1 x^2)$. However, since G^{x^2} and H^{x^2} are not available in CRS_{i+1} any more, the extractor cannot extract x'. If proof ρ is constructed based on $\hat{G}$ then x' does not include x^2. Hence we can prevent the above case.

Lemma 2 (Trapdoor extraction of x in crs_x). *Suppose that there exists a PPT adversary $\mathcal{A}$ such that given $\{(crs_{x,j}, \rho_j)\}_{j=1}^{i-1}$, $\mathcal{A}$ outputs a ρ_i where $crs_{x,j}$ is the $j-$th crs_x, and $\rho_j = (\hat{G}^{x_1 \cdots x_j}, \hat{G}^{x_j}, H^{x_j})$. Then by PKE assumption, there exists a PPT extractor χ that, given the random tape of $\mathcal{A}$ as input, outputs x_i.*

Proof. First, consider $i = 1$ which is $\mathsf{Setup}(1^\lambda; x_1)$. Since $\rho_1 = (\hat{G}^{x_1}, \hat{G}^{x_1}, H^{x_1})$ for a common exponent x_1. Since only $\hat{G}, H$ are given, it is obvious that there exists a PPT extractor χ outputting x_1 by PKE assumption.

Second, consider a case that $i = k + 1$. Assume that there exists a PPT extractor χ outputting $x_1, \ldots, x_k$ from a given $\{\rho_j\}_{j=1}^{k}$ where $\rho_j = (\bar{G}_j, \hat{G}_j, H_j) = (\hat{G}^{x_1 \ldots x_j}, \hat{G}^{x_j}, H^{x_j})$. $\rho_{k+1} = (\bar{G}_{k+1}, \hat{G}_{k+1}, H_{k+1})$ where $e(\hat{G}_{k+1}, H) = e(\hat{G}, H_{k+1})$ and $e(\bar{G}_{k+1}, H) = e(\bar{G}_k, H_{k+1})$. From $e(\hat{G}_{k+1}, H) = e(\hat{G}, H_{k+1})$, there exists a common exponent x_{k+1} such that $\hat{G}_{k+1} = \hat{G}^{x_{k+1}}$ and $H_{k+1} = H^{x_{k+1}}$. Assume that $G = \hat{G}^g$ for unknown g. By PKE assumption, $\eta \leftarrow \chi_{\mathcal{A}}$ such that $x_{k+1} = \eta_0 + \sum_{h_i \in Q \cup \{Q_g \cap Q_h\}} \eta_i h_i$ where $Q = \{\prod_{j=1}^{i} X_j\}_{i=1}^{k}$, $Q_g = \{g \cdot p_i(X)\}$, and $Q_h = \{q_i(X)\}$. Since $\{Q_g \cap Q_h\} = \emptyset$, $x_{k+1} = \eta_0 + \sum_{h_i \in Q} \eta_i h_i$. Note that set Q denotes polynomials appearing in proof ρ's, and $p_i(X)$ and q_i represent exponent polynomials for G and H in $\{crs_{x,j}\}_{j=1}^{k}$, respectively. If $\eta_i \neq 0$ for $h_i \in Q$ then $\bar{G}_{k+1} = \hat{G}^{x_1 \cdots x_{k+1}} = \hat{G}^{x_1 \cdots x_{k+1}} = \hat{G}^{x_j^2 \cdots}$. However, since $x_j^2 \notin Q$, $\mathcal{A}$ cannot compute $\bar{G}_{k+1}$. Therefore, $x_{k+1} = \eta_0$ and $\bar{G}_{k+1} = \bar{G}_k^{x_{k+1}} = \bar{G}_k^{\eta_0}$, $\hat{G}_{k+1} = \hat{G}^{x_{k+1}} = \hat{G}^{\eta_0}$ and $\hat{H}_{k+1} = H^{x_{k+1}} = H^{\eta_0}$.

Lemma 3 (Trapdoor extraction of $\alpha, \beta, \gamma, \delta$ in crs_R). *Let $Y_j = \{\alpha_j, \beta_j, \gamma_j, \delta_j\}$. Suppose that there exists a PPT adversary $\mathcal{A}$ such that given $\{(crs_{R,j}, \phi_j)\}_{j=1}^{i-1}$, $\mathcal{A}$ outputs a ϕ_i where $crs_{R,j}$ denotes j−th crs_R, and $\phi_j = (\{G_{y_j}\}_{y_j \in Y_j}, \{\bar{G}_{y_j}\}_{y_j \in Y_j}, \{\hat{G}_{y_j}\}_{y_j \in Y_j}, \{\hat{H}_{y_j}\}_{y_j \in Y_j})$. Then by PKE assumption, there exists a PPT extractor χ that, given the random tape of $\mathcal{A}$ as input, outputs Y_i.*

Proof. First, consider that $i = 1$ which is $\mathsf{Derive}(1^\lambda, crs_x, R; \alpha_1, \beta_1, \gamma_1, \delta_1)$. Since $\phi_1 = (\{G^{y_1}\}_{y_1 \in Y_1}, \{\hat{G}^{y_1}\}_{y_1 \in Y_1}, \{\hat{G}^{y_1}\}_{y_1 \in Y_1}, \{H^{y_1}\}_{y_1 \in Y_1})$. Since only $\hat{G}, \hat{H}$ are given, it is obvious that there exists a PPT extractor χ outputting $\{y_1\}_{y \in Y}$ by PKE assumption.

Second, consider a case that $i = k+1$. Assume that there exists a PPT extractor χ outputting Z_k from a given $\{\phi_j\}_{j=1}^{k}$ where $Z_k = \cup_{j=1}^{k} Y_j$, $\phi_j = (\{G_{y_j}\}_{y_j \in Y_j}, \{\bar{G}_{y_j}\}_{y_j \in Y_j}, \{\hat{G}_{y_j}\}_{y_j \in Y_j}, \{\hat{H}_{y_j}\}_{y_j \in Y_j}) = (\{G^{\sum_{i=1}^{j} y_i}\}_{y_j \in Y_j}, \{\hat{G}^{\prod_{i=1}^{j} y_i}\}_{y_j \in Y_j}, \{\hat{G}^{y_j}\}_{y_j \in Y_j}, \{H^{y_j}\}_{y_j \in Y_j})$. $\phi_{k+1} = (\{G_{y_{k+1}}\}_{y_{k+1} \in Y_{k+1}}, \{\bar{G}_{y_{k+1}}\}_{y_{k+1} \in Y_{k+1}}, \{\hat{G}_{y_{k+1}}\}_{y_{k+1} \in Y_{k+1}}, \{\hat{H}_{y_{k+1}}\}_{y_{k+1} \in Y_{k+1}})$. From $e(\hat{G}_{y_{k+1}}, H) = e(\hat{G}, \hat{H}_{yk+1})$, there exists a common exponent y_{k+1} such that $\hat{G}_{y_{k+1}} = \hat{G}^{y_{k+1}}$ and $\hat{H}_{y_{k+1}} = H^{y_{k+1}}$. Assume that $G = \hat{G}^g$ for unknown g.

By PKE assumption, $\eta \leftarrow \chi_{\mathcal{A}}$ such that $y_{k+1} = \eta_0 + \sum_{h_i \in Q \cup \{Q_g \cap Q_h\}} \eta_i h_i$ where $Q = \{\prod_{y_i \in Z_k} y_i^{b_i}\}$, $Q_g = \{g \cdot p_i(Z_k)\}$, and $Q_h = \{q_i(Z_k)\}$. Note that $Z_k = \{\alpha_1, \cdots, \delta_k\}$, $b_i = \{0, 1\}$. Since $\{Q_g \cap Q_h\} = \emptyset$, $x_{k+1} = \eta_0 + \sum_{h_i \in Q} \eta_i h_i$. If $\eta_i \neq 0$ then $\bar{G}_{y_{k+1}} = (\hat{G}^{y'_1 \cdots y'_k \cdot y_{k+1}} = \hat{G}^{y'_1 \cdots y'_k \cdot y_{k+1}} = \hat{G}^{{y'_j}^2 \cdots}$. Since ${y'_j}^2 \notin Q$, $y_{k+1} = \eta_0$ and $\bar{G}_{y_{k+1}} = \bar{G}_{y_k}^{y_{k+1}} = \bar{G}_{y_k}^{\eta_0}$, $\hat{G}_{y_{k+1}} = \hat{G}^{y_{k+1}} = \hat{G}^{\eta_0}$ and $H_{y_{k+1}} = H^{y_{k+1}} = H^{\eta_0}$.

Theorem 3. *The protocol given above 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 SAK (subversion algebraic knowledge) assumption holds, and a linear collision resistant hash exists.*

Proof. Since the updatable CRS scheme does not change Prove, Vfy, SimProve, the perfect completeness and the zero-knowledge are proven by the proof in the base scheme. Since more terms are added in CRS, the simulation extractability is needed to be proved with more terms. For instance, additional terms of $G^{\gamma x^i}$, $G^{\delta x^i}$, ... are included in CRS for updating. In a similar way as the proof in section 4, we can prove the simulation extractability which is available in Appendix B.

7 Conclusion

In this paper, we propose the efficient quadratic arithmetic program based simulation-extractable succinct non-interactive arguments of knowledge (QAP-based SE-SNARK) with 3 group elements, which requires a single equation for verification. The soundness of the proposed scheme is proven under subversion algebraic knowledge assumption and linear collision resistance, even with leveraging QAP. We also propose an SE-SNARK scheme with 2 elements as proof with the SAP representation in a symmetric group, although it is difficult to construct a scheme with 2 elements as proof from existing SE-SNARK. Furthermore, we propose an updatable CRS QAP SE-SNARK scheme with two-round update, which generates the minimal proof size (3 group elements) with a linear size CRS to the relation size.

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.

[BGG18] Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-snark. In *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*, pages 64–77, 2018.

[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.

[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.

[GKM$^+$18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-snarks. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pages 698–728, 2018.

[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.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *USENIX Security Symposium*, pages 1069–1083, 2016.

[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.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. *IACR Cryptology ePrint Archive*, 2019:99, 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.

Appendix

A Security Proof for SAP-based SE-SNARK

We show the security proof of our proposed SE-SNARKs with two group elements in section 5.3.

Theorem 2. *The protocol given above 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 $D-SAK$ 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

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

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 $X \in \mathbb{Z}_p$.

SIMULATION-EXTRACTABILITY:

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

$$
\begin{aligned}
a(\mathbf{X}) &= a_0 + a_\alpha X_\alpha + a_\delta X_\delta + a_{\alpha\delta} X_\alpha X_\delta + \textstyle\sum_{i=0}^{n-1} a_{\gamma x^i} X_\gamma X_x^i \\
&+ \textstyle\sum_{i=0}^{n-1} a_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} a_{\gamma\delta x^i} X_\gamma X_\delta X_x^i \\
&+ \sum_{i=0}^{l} a_{s_i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)) + \sum_{i=l+1}^{m} a_{s_i}(X_\gamma^2 w_i(X_x) + 2X_\alpha X_\gamma u_i(X_x)) \\
&+ \sum_{j=1}^{q_{Q_1}} a_{\lambda_j} X_{\lambda_j} + \sum_{j=0}^{q} a_{A_j} X_{\mu_j} \\
&+ \sum_{j=0}^{q} a_{C_j}(X_{\mu_j}^2 - X_\alpha^2 + X_\delta X_{\mu_j} \mathcal{H}(A_j) - X_\gamma \sum_{i=0}^{l} s_{j,i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x))) \\
c(\mathbf{X}) &= c_0 + c_\alpha X_\alpha + c_\delta X_\delta + c_{\alpha\delta} X_\alpha X_\delta + \textstyle\sum_{i=0}^{n-1} c_{\gamma x^i} X_\gamma X_x^i \\
&+ \textstyle\sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} c_{\gamma\delta x^i} X_\gamma X_\delta X_x^i \\
&+ \sum_{i=0}^{l} c_{s_i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)) + \sum_{i=l+1}^{m} c_{s_i}(X_\gamma^2 w_i(X_x) + 2X_\alpha X_\gamma u_i(X_x)) \\
&+ \sum_{j=1}^{q_{Q_1}} c_{\lambda_j} X_{\lambda_j} + \sum_{j=0}^{q} c_{A_j} X_{\mu_j} \\
&+ \sum_{j=0}^{q} c_{C_j}(X_{\mu_j}^2 - X_\alpha^2 + X_\delta X_{\mu_j} \mathcal{H}(A_j) - X_\gamma \sum_{i=0}^{l} s_{j,i}(X_\gamma w_i(X_x) + 2X_\alpha u_i(X_x)))
\end{aligned}
$$

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

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

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 $X_\alpha X_\beta$ 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, $a_{A_k}^2 = c_{C_k} = 1$. 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 $a_{A_k}\mathcal{H}(G^{a_0}A_k^{a_{A_k}}) = a_{A_k}\mathcal{H}(A_k)$. Since $\mathcal{H}$ is collision resistant, it is hard to find $G^{a_0}A_k^{a_{A_k}}$ and $A_k^{a_{A_k}}$ such that $\mathcal{H}(G^{a_0}A_k^{a_{A_k}}) = \mathcal{H}(A_k^{a_{A_k}})$ and $G^{a_0} \neq 1$. Therefore $G^{a_0} = 1$ and $a_0 = 0$. Furthermore, since $\mathcal{H}$ is linear collision resistant, it is hard to find non trivial a_{A_k} such that $\mathcal{H}(A_k^{a_{A_k}}) = a_{A_k}\mathcal{H}(A_k)$. So $a_{A_k} = 1$. 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 $a_{A_j} = 0$. Since there is X_α^2 in the right forms, $a_\alpha^2 = 1$. 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, $a(\mathbf{X}) = a_0 + \sum_{i=0}^{n-1} a_{\gamma x^i} X_\gamma X_x^i$. 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 $h(X_x) = \sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_x^i$ 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).

B Security Proof for CRS updatable QAP-based SE-SNARK

Theorem 3. *The protocol given above 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 SAK (subversion algebraic knowledge) assumption holds, and a linear collision resistant hash exists.*

Proof. 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 $D - SAK$, given a proof $\pi = (G^a, H^b, H^c)$, 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:

$$
\begin{aligned}
a(\mathbf{X}) &= a_\alpha X_\alpha + a_\beta X_\beta + a_{\alpha\delta} X_\alpha X_\delta + \textstyle\sum_{i=0}^{n-1} a_{\gamma x^i} X_\gamma X_x^i + \textstyle\sum_{i=0}^{n-1} a_{x^i} X_x^i \\
&+ \textstyle\sum_{i=0}^{n-1} a_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} a_{\gamma\delta x^i} X_\gamma X_\delta X_x^i + \sum_{i=0}^{n-1} a_{\delta x^i} X_\delta X_x^i \\
&+ \sum_{i=0}^{l} a_{s_i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) \\
&+ \sum_{i=l+1}^{m} a_{s_i} (X_\gamma^2 w_i(X_x) + X_\beta X_\gamma u_i(X_x) + X_\alpha X_\gamma v_i(X_x)) + \sum_{j=1}^{q_{Q_1}} a_{\lambda_j} X_{\lambda_j} \\
&+ \sum_{i=l+1}^{m} a_{\beta u_i} X_\beta u_i(X_x) + \sum_{i=l+1}^{m} a_{\alpha v_i} X_\alpha v_i(X_x) + \sum_{j=0}^{q} a_{A_j} X_{\mu_j} \\
&+ \sum_{j=1}^{q} a_{C_j} (X_{\mu_j} X_{\nu_j} - X_\alpha X_\beta - X_\gamma \sum_{i=0}^{l} s_{j,i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) \\
&+ h_2 X_\delta X_{\mu_j} + h_1 X_{\nu_1} + h_1 h_2 X_\delta)
\end{aligned}
$$

$$
\begin{aligned}
b(\mathbf{X}) &= b_\beta X_\beta + b_\delta X_\delta + \textstyle\sum_{i=0}^{n-1} b_{\gamma x^i} X_\gamma X_x^i + \textstyle\sum_{i=0}^{n-1} b_{x^i} X_x^i + \sum_{j=1}^{q_{Q_2}} b_{\rho_j} X_{\rho_j} + \sum_{j=1}^{q} b_{B_j} X_{\nu_j} \\
c(\mathbf{X}) &= c_\alpha X_\alpha + c_\beta X_\beta + c_{\alpha\delta} X_\alpha X_\delta + \textstyle\sum_{i=0}^{n-1} c_{\gamma x^i} X_\gamma X_x^i + \textstyle\sum_{i=0}^{n-1} c_{x^i} X_x^i \\
&+ \sum_{i=0}^{n-1} c_{\gamma x^i} X_\gamma X_x^i + \textstyle\sum_{i=0}^{n-1} c_{\gamma^2 t x^i} X_\gamma^2 t(X_x) X_x^i + \sum_{i=0}^{n-1} c_{\gamma\delta x^i} X_\gamma X_\delta X_x^i \\
&+ \sum_{i=0}^{n-1} c_{\delta x^i} X_\delta X_x^i + \sum_{i=0}^{l} c_{s_i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) \\
&+ \sum_{i=l+1}^{m} c_{s_i} (X_\gamma^2 w_i(X_x) + X_\beta X_\gamma u_i(X_x) + X_\alpha X_\gamma v_i(X_x)) + \sum_{j=1}^{q_{Q_1}} c_{\lambda_j} X_{\lambda_j} \\
&+ \sum_{i=l+1}^{m} c_{\beta u_i} X_\beta u_i(X_x) + \sum_{i=l+1}^{m} c_{\alpha v_i} X_\alpha v_i(X_x) + \sum_{j=1}^{q} c_{A_j} X_{\mu_j} \\
&+ \sum_{j=1}^{q} c_{C_j} (X_{\mu_j} X_{\nu_j} - X_\alpha X_\beta - X_\gamma \sum_{i=0}^{l} s_{j,i} (X_\gamma w_i(X_x) + X_\beta u_i(X_x) + X_\alpha v_i(X_x)) \\
&+ h_2 X_\delta X_{\mu_j} + h_1 X_{\nu_j} + h_1 h_2 X_\delta)
\end{aligned}
$$

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

$$\begin{aligned}&(a(\mathbf{X})+h_1)(b(\mathbf{X})+h_2X_\delta)\\&=X_\alpha X_\beta+X_\gamma\sum_{i=0}^{l}a_{s_i}(X_\gamma w_i(X_x)+X_\beta u_i(X_x)+X_\alpha v_i(X_x))+c(\mathbf{X})\end{aligned}\tag{7}$$

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 $X_\gamma X_{\mu_k}$, $X_x X_{\mu_k}$ 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 $X_{\mu_k}X_\nu$ in the left form, 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 (7) 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

$$\begin{aligned}c(\mathbf{X})=c_0+c_{A_k}X_{\mu_k}+c_{C_k}(&X_{\mu_k}X_{\nu_k}-X_\alpha X_\beta+h_2X_\delta X_{\mu_k}+h_1X_{\nu_k}+h_1h_2X_\delta\\&-X_\gamma\sum_{i=0}^{l}s_{k,i}(X_\gamma w_i(X_x)+X_\beta u_i(X_x)+X_\alpha v_i(X_x))\end{aligned}$$

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

Finally, we obtain the following equation.

$$\begin{aligned}&(a_0+a_{\mu_k}X_{\mu_k}+h_1')(b_0+b_\delta X_\delta+b_{\nu_k}X_{\nu_k}+h_2'X_\delta)\\&=c_0+c_{A_k}X_{\mu_k}+X_{\mu_k}X_{\nu_k}+h_2X_\delta X_{\mu_k}+h_1X_{\nu_k}+h_1h_2X_\delta\\&-X_\gamma\sum_{i=0}^{l}s_{k,i}(X_\gamma w_i(X_x)+X_\beta u_i(X_x)+X_\alpha v_i(X_x))\end{aligned}$$

Since $a_{\mu_k}b_{\nu_k}=1$, there is $(a_0+h_1')b_{\nu_k}X_{\nu_k}$ in the left form, and there is $h_1X_{\nu_k}$ in the right form, $(a_0+h_1')b_{\nu_k}=h_1$, $h_1'=-a_0+a_{\mu_k}h_1$, and $\mathcal{H}(G^{-(-a_0)}A_k^{a_{\mu_k}})$ $=-a_0+a_{\mu_k}\mathcal{H}(A_k)$. Since $\mathcal{H}$ is linear collision resistant, it is hard to find non trivial $-a_0$ and a_{μ_k}. Hence $a_0=0$, and $a_{\mu_k}=1$. Similarly, Since there is $(b_\delta + h_2')a_{\mu_k}X_\delta X_{\mu_k}$ in the left form, and there is $h_2X_\delta X_{\mu_k}$ in the right form, $(b_\delta + h_2')a_{\mu_k} = h_2$, $h_2' = -b_\delta + b_{\nu_k}h_2$, and $\mathcal{H}(H^{b_0}H^{-(-b_\delta)}B_k^{b_{\nu_k}}) = -b_\delta + b_{\nu_k}\mathcal{H}(B_k)$. Since $\mathcal{H}$ is collision resistant, it is hard to find non trivial b_0 such that $\mathcal{H}(H^{b_0}H^{b_\delta}B_k^{b_{\nu_k}}) = \mathcal{H}(H^{b_\delta}B_k^{b_{\nu_k}})$. Hence $b_0 = 0$. In addition, since $\mathcal{H}$ is linear collision resistant, it is hard to find non trivial $-b_\delta$ and b_{ν_k} satisfying $\mathcal{H}(H^{-(-b_\delta)}B_k^{b_{\nu_k}}) = -b_\delta + b_{\nu_k}\mathcal{H}(B_k)$. Hence $b_\delta = 0$, and $b_{\nu_k} = 1$.

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 $a_{A_j} = b_{B_j} = 0$. Since there is $X_\alpha X_\beta$ in the right form, $a_\alpha b_\beta = 1$.

In the right form of (7), 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

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

In (7),

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

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

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

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).