Simulation Extractable Versions of Groth's zk-SNARK Revisited

Oussama Amine[1], Karim Baghery[2], Zaira Pindado[3], and Carla Ràfols[4]

[1] University of Oslo, Oslo, Norway
[2] COSIC, KU Leuven, Leuven, Belgium
[3] Dusk, Amsterdam, Netherlands
[4] Universitat Pompeu Fabra, Barcelona, Spain
oussamaa@math.uio.no, karim.baghery@kuleuven.be,
zaira@dusk.network, carla.rafols@upf.edu

Abstract. Zero-knowledge Succinct Non-interactive Arguments of Knowledge (zk-SNARKs) are the most efficient proof systems in terms of proof size and verification. Currently, Groth's scheme from EUROCRYPT 2016, Groth16, is the state-of-the-art and is widely deployed in practice. Groth16 is originally proven to achieve knowledge soundness, which does not guarantee the non-malleability of proofs. There has been considerable progress in presenting new zk-SNARKs or modifying Groth16 to efficiently achieve *strong* Simulation Extractability (SE), which is shown to be a necessary requirement in some applications. In this paper, we revise the Random Oracle (RO) based variant of Groth16 proposed by Bowe and Gabizon, BG18, the most efficient one in terms of prover efficiency and CRS size among the candidates, and present a more efficient variant that saves 2 pairings in the verification and 1 group element in the proof. This supersedes our preliminary construction, presented in CANS 2020 [BPR20], which saved 1 pairing in the verification, and was proven in the Generic Group Model (GGM). Our new construction also improves on BG18 in that our proofs are in the Algebraic Group Model (AGM) with Random Oracles and reduces security to standard computational assumptions in bilinear groups (as opposed to using the full power of the GGM). We implement our proposed SE zk-SNARK along with BG18 in the Arkworks library, and compare the efficiency of our scheme with some related works. Our empirical experiences confirm that our SE zk-SNARK is more efficient than all previous SE schemes in most dimensions and it has very close efficiency to the original Groth16 [5].

Keywords: NIZK, zk-SNARK, Strong Simulation Extractability, Algebraic Group Model, Random Oracle Model

1 Introduction

Non-Interactive Zero-Knowledge (NIZK) proof systems [BFM88] are a fundamental family of cryptographic primitives that have appeared recently in a wide

[5] A preliminary version of this paper appeared in the *Proceedings of 19th International Conference on Cryptology and Network Security, CANS 2020* [BPR20].

range of practical applications. A NIZK proof system allows a party to prove that for a public statement $\vec{x}$, she knows a witness $\vec{w}$ such that $(\vec{x}, \vec{w}) \in \mathbf{R}$, for some relation $\mathbf{R}$, without leaking any information about $\vec{w}$ and without interaction with the verifier. Due to their impressive advantages, NIZK proof systems are used ubiquitously to build larger cryptographic protocols and systems.

Zero-knowledge Succinct Arguments of Knowledge (zk-SNARKs) are among the most interesting NIZK proof systems in practice, as they allow to generate very short proofs for NP complete languages, which can be verified in less than 10 milliseconds [GGPR13, Gro16]. Zk-SNARKs have had a tremendous impact in practice and they have found numerous applications, including verifiable computation systems [PHGR13], privacy-preserving (PP) cryptocurrencies [BCG+14], PP smart contract systems [KMS+16], PP proof-of-stake protocols [KKKZ19], and efficient ledger verification protocols [BMRS20], are some of the best known applications that use zk-SNARKs to prove different statements very efficiently while guaranteeing the privacy of the prover. Because of their practical importance, particularly in large-scale applications like blockchains, even minimal savings especially in proof size or verification cost are considered to be relevant.

In 2016, Groth [Gro16] introduced the most efficient zk-SNARK for Quadratic Arithmetic Programs or QAPs, which is still the state-of-the-art construction, Groth16. It is constructed using bilinear groups and its proof is 3 group elements (2 from $\mathbb{G}_1$ and 1 from $\mathbb{G}_2$) and the cost of verification is dominated by 3 pairing computations. In the original paper, it is proven to achieve knowledge soundness in the generic group model (GGM). In 2018, Fuchsbauer, Kiltz, and Loss [FKL18] defined the Algebraic Group Model (AGM) and reproved its security in this weaker model. The proof of Groth16 is malleable, as it is shown in [GM17]. Generating non-malleable proofs is a necessary requirement in building various cryptographic schemes, including *universally composable* protocols [KMS+16, KKKZ19], cryptocurrencies (e.g. Zcash) [BCG+14], signature-of-knowledge schemes [GM17], etc. Practical systems like Zcash cryptocurrency [BCG+14] that uses the original Groth16 [Gro16] make extra efforts to ensure the non-malleability of transactions and the proof of underlying proof system. Considering such concerns, in practice, it is important to have a stronger notion of knowledge soundness, known as (strong) Simulation Extractability (SE). This notion guarantees that a valid witness can be extracted from any adversary producing a proof accepted by the verifier, even after seeing an arbitrary number of simulated proofs.

There have been considerable efforts to construct new SE zk-SNARKs or refine Groth's zk-SNARK to achieve SE and guarantee the non-malleability of proofs. Firstly, in 2017 Groth and Maller [GM17] proposed an SE zk-SNARK,

which is very efficient in terms of proof size but very inefficient in terms of Common Reference String (crs) size and prover time. They also showed how one can use SE zk-SNARKs to build Signature of Knowledge (SoK) schemes [CL06] with *succinct* signatures. In 2018 Bowe and Gabizon [BG18] proposed a less efficient construction in terms of proof size (5 group elements vs 3 in the original version) based on Groth16 which needs a Random Oracle (RO) (apart from GGM) which returns group elements, but with almost no overhead in the crs size or additional cost for the prover. In [Lip22], Lipmaa proposed several constructions, including an efficient QAP-based SE zk-SNARK in terms of proof size and with the same verification complexity as [GM17, BG18], but less efficient in terms of crs size and prover time compared to [BG18] and Groth16. In [AB19], Atapoor and Baghery used the traditional OR technique to achieve SE in Groth16. Their variant requires 1 pairing less for verification in comparison with previous SE constructions, however it comes with an overhead in proof generation, crs size, and even larger overhead in the proof size. For a particular instantiation they add ≈ 52.000 constrains to the underlying QAP instance, which adds fixed overhead to the prover and crs size, that can be considerable for mid-size circuits. They show that for a circuit with 10×10^6 Multiplication (Mul) gates, their prover is about 10% slower, but it can be slower for circuits with less than 10×10^6 gates. In [KLO20], Kim, Lee, and Oh proposed a QAP-based SE zk-SNARK with the same crs size and prover time compared to [Lip22], but with slightly shorter proofs and more efficient verification.

These works also differ significantly in the assumptions they make for security. The scheme of Groth and Maller [GM17] is based on a knowledge assumption and other falsifiable computational assumptions, and they are all q-type assumptions where q is the size of the circuit. In this work, the authors avoid the generic group model by making a concrete knowledge assumption that is essential for extracting the witness. On the other hand, the work of Bowe and Gabizon [BG18] uses the full power of the generic group model to prove the security. The construction of Bowe and Gabizon uses the generic group model plus the assumption that a certain hash function to group elements is a random oracle. All the constructions of Lipmaa [Lip22] are proven secure in a weaker notion of the AGM, where the adversary has access to a random oracle that allows it to sample random elements obliviously in the group, i.e. without knowing the random oracles.

Recently, Baghery, Kohlweiss, Siim, and Volkhov [BKSV21] explore another direction. Instead of modifying Groth16 to achieve *strong* SE, they first show that the original construction of Groth16 achieves *weak* SE with non-black-box extraction. Weak SE allows proof randomization, therefore the proof is malleable, while it guarantees that a proof cannot be changed to prove a new

Table 1. A comparison of our proposed variations of Groth16 along with the other SE zk-SNARKs for arithmetic circuit satisfiability with n Mul gates (constraints) and m wires (variables), of which l are public input wires (variables). A typical set of values is $n = m = 10^6$ and $l = 10$. In the case of crs size and prover's computation we omit constants. In [GM17], n Mul gates and m wires translate to $2n$ squaring gates and $2m$ wires. In [AB19], SE is achieved with an OR approach which requires to add constraints and variables, resulting in $n' \approx n + 52.000$, $m' \approx m + 52.000$, and $l' = l + 4$. $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$: group elements, E_i: exponentiation in group $\mathbb{G}_i$, M_i: multiplication in group $\mathbb{G}_i$, P: pairings. GGM: Generic Group Model, ROM: Random Oracle Model, AGM: Algebraic Group Model, HAK: Hash Algebraic Knowledge assumption, LCR: Linear Collision Resistance hash functions, CRH: Collision Resistant Hash, VE: Number of verification equations, WSE: Weak Simulation Extractable, SSE: Strong Simulation Extractable.

SNARK	SE	Model	CRS size	Prover	Proof	Verifier	VE
[Gro16, FKL18]	WSE	AGM	$m+2n-l$ $\mathbb{G}_1$ n $\mathbb{G}_2$	$m+3n-l$ E_1 n E_2	2 $\mathbb{G}_1$ 1 $\mathbb{G}_2$	l E_1 3 P	1
[GM17]	SSE	GGM	$2m+4n$ $\mathbb{G}_1$ $2n$ $\mathbb{G}_2$	$2m+4n-l$ E_1 $2n$ E_2	2 $\mathbb{G}_1$ 1 $\mathbb{G}_2$	l E_1 5 P	2
[BG18]	SSE	GGM, ROM	$m+2n-l$ $\mathbb{G}_1$ n $\mathbb{G}_2$	$m+3n-l$ E_1 n E_2	3 $\mathbb{G}_1$ 2 $\mathbb{G}_2$	l E_1 5 P	2
[AB19]	SSE	GGM	$m'+2n'-l$ $\mathbb{G}_1$ n' $\mathbb{G}_2$	$m'+3n'-l$ E_1 n' E_2	4 $\mathbb{G}_1$ 2 $\mathbb{G}_2$ + 2 λ	$l'+2$ E_1 4 P	2
[Lip22]	SSE	AGM, tag-based	$m+3n-l$ $\mathbb{G}_1$ n $\mathbb{G}_2$	$m+4n-l$ E_1 n E_2	3 $\mathbb{G}_1$ 1 $\mathbb{G}_2$	$l+1$ E_1 5 P	2
[KLO20]	SSE	HAK, LCR	$m+3n-l$ $\mathbb{G}_1$ n $\mathbb{G}_2$	$m+4n-l$ E_1 n E_2	2 $\mathbb{G}_1$ 1 $\mathbb{G}_2$	$l+1$ E_1, 1 E_2 3 P	1
[BPR20], A	SSE	GGM, CRH	$m+2n-l$ $\mathbb{G}_1$ n $\mathbb{G}_2$	$m+3n-l$ E_1 n E_2	3 $\mathbb{G}_1$ 2 $\mathbb{G}_2$	l E_1, 1 E_2 1 E_T, 4 P	2
Section 3	SSE	AGM, ROM	$m+2n-l$ $\mathbb{G}_1$ n $\mathbb{G}_2$	$m+3n-l$ E_1 n E_2	2 $\mathbb{G}_1$ 2 $\mathbb{G}_2$	l E_1, 1 E_2 3 P	1

statement. Then, considering the first result, they propose two efficient constructions of Groth16 that achieve weak SE with *black-box* extraction which is shown to be necessary for UC-security. Both *weak* and *strong* SE zk-SNARKs can be lifted to achieve black-box simulation extractability with a simpler compiler [Bag19a, BKSV21], rather than with the COCO framework [KZM+15] which is constructed to lift (knowledge) sound NIZK proofs systems to achieve black-box SE. However, to realize the standard ideal functionality defined for NIZK arguments, one would need to use a strong SE NIZK with black-box extraction [Gro06]. Therefore, constructing a more efficient strong SE zk-SNARK, would also allow to build more efficient *black-box* SE zk-SNARK to be used in UC-secure protocols.

Our Contributions. Our main contribution is to revise the simulation extractable variants of Groth16, presented in [BG18] and [AB19], to achieve a better effi-

ciency and get the best of both constructions. Namely, achieving *strong* simulation extractability in Groth16 with minimal overhead.

Our focus is mainly on Bowe and Gabizon's variation [BG18] which has the most efficient prover and the shortest crs among other (strong) SE zk-SNARKs [GM17, BG18, AB19, Lip22, KLO20], while it uses a RO which returns group elements. To achieve (strong) simulation extractability, their prover replaces all the original computations which depend on some parameter δ given in the crs by some δ' and the prover must give $[\delta']_2$ and a proof of knowledge (PoK) of the DLOG of $[\delta']_2$ w.r.t $[\delta]_2$. Using this technique, they present a variation that has the same CRS as Groth16, almost the same prover as Groth16, 2 new elements in the proof (one from $\mathbb{G}_1$ and the other from $\mathbb{G}_2$), and an additional verification equation that adds 2 pairing operations to the verification of Groth16.

In this paper, using the same approach [BG18] and some subtle modifications, we construct a *strong* SE zk-SNARK that results in the most efficient (strong) simulation extractable variant of Groth16 in terms of crs size, prover complexity, and verification time. Our SE zk-SNARK uses some sophisticated modification of Boneh-Boyen signatures [BB08] to prove knowledge of the DLOG of δ' which requires 1 less $\mathbb{G}_1$ element in the proof, and 2 pairings less in the verification in comparison with the argument of Bowe and Gabizon [BG18], but at the cost of one additional exponentiation in the verification. Our construction supersedes and improves a preliminary version of this work presented at CANS 2020 [BPR20], where in all constructions verification required at least one additional pairing and proofs were in the Generic Group Model (GGM).

Our construction modifies the proof generation of Groth16 slightly and include the PoK of the DLOG of $[\delta']_2$ w.r.t $[\delta]_2$ inside the original proof of Groth16. Using this, we manage to save 1 element in the proof, and 2 pairings in the verification of Bowe and Gabizon's construction [BG18], at the cost of a single exponentiation in $\mathbb{G}_2$ in the verification. This construction shows that using a random oracle, we can achieve strong SE in Groth16, at the cost of one additional $\mathbb{G}_2$ element in the proof, and one new exponentiation in $\mathbb{G}_2$ in the verification. In the case of verifying a larger number of proofs where verifiers of our constructions gain efficiency by using *multi-scalar exponentiations*, our construction achieves almost the same efficiency as Groth16.

Tab. 1 presents a comparison of our proposed variant of Groth16 with several other constructions for a particular instance of arithmetic circuit satisfiability. As it can be seen, in comparison with Bowe and Gabizon's construction [BG18], our construction retains most of the properties requires 2 less pairing in the verification, at the cost of 1 additional exponentiation in the verification. We also compare our construction with the results initially obtained and presented in CANS 2020 [BPR20]. We note that in both our constructions, the

hash function maps into $\mathbb{Z}_p$ and not to a source group as in [BG18], which is an additional practical advantage. In comparison with Atapoor and Baghery's construction [AB19], both of our variants have a negligible overhead in the proof generation and crs size, and a smaller overhead in proof size. Above all, our best construction, requires 3 parings in the verification, instead of 4. [6] We reduce security to a q-DLOG in the AGM with random oracles, where q is the size of the circuit. In contrast, our preliminary result [BPR20] was in the GGM but only required the hash function to be collision resistant.

As a part of our contribution, we also present an open-source prototype implementation of our presented constructions and Bowe and Gabizon's scheme in the Arkworks library, which currently is one of the most popular ecosystems written in Rust for developing and programming with zk-SNARKs. Then, we use our implementations along with the implementations of Groth16 [Gro16] and Groth-Maller [GM17], which already exist in Arkworks library, and present a comprehensive benchmark for the relevant simulation extractable zk-SNARKs [Gro16, GM17, BG18]. Full details of our empirical analysis are reported in Section 4, in Table 2. As we expected, the implementation results show that, our new construction is more efficient than the first one, and also it is more efficient than all previous SE zk-SNARKs in most dimensions and more importantly it has a very close efficiency profile to the original Groth16, particularly when we need to verify a large number of proofs.

Finally, we highlight that using the technique proposed in [GM17], both of or proposed SE zk-SNARKs can be turned into *succinct* SoK schemes, which would be more efficient than previous constructions. In general, due to relying on non-falsifiable assumptions, succinct SoK schemes have better efficiency in comparison with constructions that are built under standard assumptions [CL06, BFG13, BGPR20]. We also note that to achieve strong (non-black-box) SE, our proposed zk-SNARKs require minimal changes in comparison with the original Groth16. Therefore, one can use the same compiler or ad-hoc approach proposed in [Bag19a] and [BKSV21], respectively, to construct a more efficient strong *black-box* SE zk-SNARK for UC-protocols [Gro06].

Organization. In Section 2, we introduce notation, the relevant security definitions, and recall the Boneh-Boyen signature scheme. In Section 3, we present our new and the most efficient SE zk-SNARK, that has very close efficiency to the Groth16. We evaluate the practical efficiency of both presented constructions in Section 4 using a prototype Rust implementation in Arkworks library.

[6] In the worst case, our changes add only one element to the crs of Groth16 and since Groth16 is already proven to achieve subversion ZK (ZK without trusting a third party) [ABLZ17, Fuc18], our variants also can be proven to achieve Sub-ZK using the technique proposed in [Fuc18, Bag19b].

We also compare the efficiency of our constructions with several relevant SE zk-SNARKs in the same section. Finally we conclude the paper in Section 5. For the sake of completeness, in A, we also recall our first SE zk-SNARK [BPR20] that relaxes the RO in Bowe and Gabizon's scheme [BG18] to a collision resistant hash function, and also saves 1 pairing in the verification. We implement that scheme as well and include it in our benchmarks.

Novelty. Compared to the conference version published in CANS 2020 [BPR20], this version includes a more efficient construction presented in Section 3, a prototype Rust implementation of our presented constructions along with Bowe and Gabizon's scheme [BG18] in Arkworks library, followed by a comprehensive efficiency comparison of relevant SE zk-SNARKs that are reported with details in Section 4.

2 Preliminaries

2.1 Notation and bilinear groups

We let BGgen be a probabilistic polynomial time algorithm which on input 1^λ, where λ is the security parameter, returns the description of an asymmetric bilinear group $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathscr{P}_1, \mathscr{P}_2)$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order p, the elements $\mathscr{P}_1, \mathscr{P}_2$ are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable, non-degenerate bilinear map, and there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Elements in $\mathbb{G}_i$, are denoted implicitly as $[a]_i = a\mathscr{P}_i$, where $i \in \{1, 2, T\}$ and $\mathscr{P}_T = e(\mathscr{P}_1, \mathscr{P}_2)$. With this notation, $e([a]_1, [b]_2) = [a]_1[b]_2 = [ab]_T$. We extend this notation naturally to vectors and matrices. We denote by $\mathsf{negl}(\lambda)$ an arbitrary negligible function in λ.

2.2 Definitions

For an algorithm $\mathscr{A}$, let $\mathbf{Im}(\mathscr{A})$ be the image of $\mathscr{A}$, i.e. the set of valid outputs of $\mathscr{A}$. By $y \leftarrow \mathscr{A}(x; r)$ we denote the fact that $\mathscr{A}$, given an input x and a randomizer r, outputs y.

We use the definitions of NIZK arguments from [Gro16]. Let $\mathscr{R}$ be a relation generator, such that $\mathscr{R}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathbf{R} = \{(\vec{x}, \vec{w})\}$. Here, $\vec{x}$ is the statement and $\vec{w}$ is the witness. Security parameter λ can be deduced from the description of $\mathbf{R}$. The relation generator also outputs auxiliary information $\mathsf{z}_\mathbf{R}$ that will be given to the honest parties and the adversary. In our constructions, $\mathsf{z}_\mathbf{R}$ will be the description of a bilinear group.

As in [Gro16], $\mathsf{z}_{\mathbf{R}}$ is the value returned by $\mathsf{BGgen}(1^\lambda)$, and is given as an input to the parties.

Let $\mathscr{L}_{\mathbf{R}} = \{\vec{x} : \exists \vec{w}, (\vec{x}, \vec{w}) \in \mathbf{R}\}$ be an NP-language. A *NIZK argument system* Ψ for $\mathscr{R}$ consists of tuple of PPT algorithms $(\mathsf{K}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$, such that:

CRS Generator: K is a PPT algorithm that, given $(\mathbf{R}, \mathsf{z}_{\mathbf{R}})$ where $(\mathbf{R}, \mathsf{z}_{\mathbf{R}}) \in \mathbf{Im}(\mathscr{R}(1^\lambda))$, outputs $\mathsf{crs} := (\mathsf{crs}_{\mathsf{P}}, \mathsf{crs}_{\mathsf{V}})$ and stores trapdoors of crs as $\vec{\mathsf{ts}}$. We distinguish $\mathsf{crs}_{\mathsf{P}}$ (needed by the prover) from $\mathsf{crs}_{\mathsf{V}}$ (needed by the verifier).

Prover: P is a PPT algorithm that, given $(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \vec{x}, \vec{w})$, where $(\vec{x}, \vec{w}) \in \mathbf{R}$, outputs an argument π. Otherwise, it outputs $\perp$.

Verifier: V is a PPT algorithm that, given $(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \vec{x}, \pi)$, returns either 0 (reject) or 1 (accept).

Simulator: Sim is a PPT algorithm that, given $(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}, \vec{\mathsf{ts}}, \vec{x})$, outputs a simulated argument π.

Besides *succinct* proofs, i.e. polynomial in λ, an SE zk-SNARK is required to satisfy *completeness*, *simulation extractability*, and *zero-knowledge*.

Definition 2.1 (Perfect Completeness). *A non-interactive argument Ψ is* perfectly complete for $\mathscr{R}$*, if for all λ, all $(\mathbf{R}, \mathsf{z}_{\mathbf{R}}) \in \mathbf{Im}(\mathscr{R}(1^\lambda))$, and $(\vec{x}, \vec{w}) \in \mathbf{R}$,*

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{K}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}),\ \pi \leftarrow \mathsf{P}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}, \vec{x}, \vec{w}) : \\ \mathsf{V}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}, \vec{x}, \pi) = 1\end{array}\right] = 1.$$

Intuitively, perfect completeness states that an honest prover P always convinces an honest verifier V.

Definition 2.2 (Computationally Knowledge-Soundness [Gro16]). *A non-interactive argument Ψ is computationally (adaptively)* knowledge-sound *for $\mathscr{R}$, if for every non-uniform PPT $\mathscr{A}$, there exists a non-uniform PPT extractor $\mathsf{Ext}_{\mathscr{A}}$, s.t. for all λ, the following probability is $\mathsf{negl}(\lambda)$,*

$$\Pr\left[\begin{array}{l}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}) \leftarrow \mathscr{R}(1^\lambda), (\mathsf{crs} \,\|\, \vec{\mathsf{ts}}) \leftarrow \mathsf{K}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}), \\ (\vec{x}, \pi) \leftarrow \mathscr{A}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}), \vec{w} \leftarrow \mathsf{Ext}_{\mathscr{A}}(\mathsf{trans}_{\mathscr{A}}) : \\ (\vec{x}, \vec{w}) \notin \mathbf{R} \wedge \mathsf{V}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}, \vec{x}, \pi) = 1\end{array}\right].$$

Here, $\mathsf{trans}_{\mathscr{A}}$ is a list containing all of $\mathscr{A}$'s inputs and outputs. Intuitively, the definition states that if an adversary can convince the verifier, she *knows* the witness. A knowledge-sound Ψ also is called an *argument of knowledge*.

Definition 2.3 (Weak Simulation Extractability [KZM+15]). *A non-interactive argument Ψ is* (non-black-box) weak simulation-extractable *for $\mathscr{R}$, if for any*

non-uniform PPT $\mathscr{A}$, there exists a non-uniform PPT extractor $\mathsf{Ext}_{\mathscr{A}}$ *s.t. for all* λ*, the following probability is* $\mathsf{negl}(\lambda)$*,*

$$\Pr\left[\begin{array}{l}(\mathbf{R},\mathsf{z}_{\mathbf{R}}) \leftarrow \mathscr{R}(1^{\lambda}), (\mathsf{crs} \,\|\, \vec{\mathsf{ts}}) \leftarrow \mathsf{K}(\mathbf{R},\mathsf{z}_{\mathbf{R}}), \\ (\vec{x},\pi) \leftarrow \mathscr{A}^{\mathsf{O}(\vec{\mathsf{ts}},.)}(\mathbf{R},\mathsf{z}_{\mathbf{R}},\mathsf{crs}), \vec{w} \leftarrow \mathsf{Ext}_{\mathscr{A}}(\mathsf{trans}_{\mathscr{A}}) : \\ \vec{x} \notin Q \wedge (\vec{x},\vec{w}) \notin \mathbf{R} \wedge \mathsf{V}(\mathbf{R},\mathsf{z}_{\mathbf{R}},\mathsf{crs},\vec{x},\pi) = 1\end{array}\right].$$

Here, Q is the set of statements queried by adversary to the simulation oracle O, and $\mathsf{trans}_{\mathscr{A}}$ is a list containing all of $\mathscr{A}$'s inputs and outputs. Note that this variant of simulation extractability allows proof randomization, while it ensures that a proof cannot be changed to prove a new statement.

Definition 2.4 (Simulation Extractability [GM17]). *A non-interactive argument* Ψ *is* (non-black-box strong) simulation-extractable for $\mathscr{R}$*, if for any non-uniform PPT $\mathscr{A}$, there exists a non-uniform PPT extractor* $\mathsf{Ext}_{\mathscr{A}}$ *s.t. for all* λ*, the following probability is* $\mathsf{negl}(\lambda)$*,*

$$\Pr\left[\begin{array}{l}(\mathbf{R},\mathsf{z}_{\mathbf{R}}) \leftarrow \mathscr{R}(1^{\lambda}), (\mathsf{crs} \,\|\, \vec{\mathsf{ts}}) \leftarrow \mathsf{K}(\mathbf{R},\mathsf{z}_{\mathbf{R}}), \\ (\vec{x},\pi) \leftarrow \mathscr{A}^{\mathsf{O}(\vec{\mathsf{ts}},.)}(\mathbf{R},\mathsf{z}_{\mathbf{R}},\mathsf{crs}), \vec{w} \leftarrow \mathsf{Ext}_{\mathscr{A}}(\mathsf{trans}_{\mathscr{A}}) : \\ (\vec{x},\pi) \notin Q \wedge (\vec{x},\vec{w}) \notin \mathbf{R} \wedge \mathsf{V}(\mathbf{R},\mathsf{z}_{\mathbf{R}},\mathsf{crs},\vec{x},\pi) = 1\end{array}\right].$$

Here, Q is the set of simulated statement-proof pairs generated by adversary's queries to the simulation oracle O, and $\mathsf{trans}_{\mathscr{A}}$ is a list containing all of $\mathscr{A}$'s inputs and outputs.

Note that both variants of *simulation extractability* implies *knowledge soundness* (given in Def. 2.2), as the earlier is a strong notion of the later which additionally the adversary is allowed to send query to the proof simulation oracle.

Definition 2.5 (Zero-Knowledge (ZK) [Gro16]). *A non-interactive argument* Ψ *is* computationally ZK for $\mathscr{R}$*, if for all* λ*, all* $(\mathbf{R},\mathsf{z}_{\mathbf{R}}) \in \mathbf{Im}(\mathscr{R}(1^{\lambda}))$*, and for all non-uniform PPT $\mathscr{A}$,* $\varepsilon_0 \approx_c \varepsilon_1$*, where*

$$\varepsilon_b = \Pr[(\mathsf{crs} \,\|\, \vec{\mathsf{ts}}) \leftarrow \mathsf{K}(\mathbf{R},\mathsf{z}_{\mathbf{R}}) : \mathscr{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathbf{R},\mathsf{z}_{\mathbf{R}},\mathsf{crs}) = 1].$$

Here, the oracle $\mathsf{O}_0(\vec{x},\vec{w})$ *returns* $\perp$ *(reject) if* $(\vec{x},\vec{w}) \notin \mathbf{R}$*, and otherwise it returns* $\mathsf{P}(\mathbf{R},\mathsf{z}_{\mathbf{R}},\mathsf{crs}_{\mathsf{P}},\vec{x},\vec{w})$*. Similarly,* $\mathsf{O}_1(\vec{x},\vec{w})$ *returns* $\perp$ *(reject) if* $(\vec{x},\vec{w}) \notin \mathbf{R}$*, otherwise it returns* $\mathsf{Sim}(\mathbf{R},\mathsf{z}_{\mathbf{R}},\mathsf{crs},\vec{\mathsf{ts}},\vec{x})$*.* Ψ *is* perfect ZK for $\mathscr{R}$ *if one requires that* $\varepsilon_0 = \varepsilon_1$*.*

Intuitively, a non-interactive argument is ZK if it does not leak extra information beyond the truth of the statement.

2.3 Boneh-Boyen signatures

We briefly recall one of the constructions of Boneh-Boyen signatures [BB08], that is used implicitly in our constructions. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear group. Messages are elements of $\mathbb{Z}_p$, and signatures are elements of $\mathbb{G}_1$. The secret key is $\mathsf{sk} \in \mathbb{Z}_p$, and the public key (verification key) is $[\mathsf{sk}]_2 \in \mathbb{G}_2$. To sign a message $m \in \mathbb{Z}_p$, the signer computes

$$[\sigma]_1 = \left[\frac{1}{\mathsf{sk}+m}\right]_1.$$

The verifier accepts the signature if the equation $e([\sigma]_1, [\mathsf{sk}]_2 + [m]_2) = [1]_T$ holds.

Boneh-Boyen signatures are existentially unforgeable under the q-SDH assumption. We use them in our constructions as proofs of knowledge of the secret key in the AGM.

2.4 Algebraic Group Model

The Algebraic Group Model or AGM for short [FKL18] assumes that adversaries are algebraic, i.e. they construct their output group elements as a linear combination of previously seen group elements. This model is a weakening of the Generic Group Model (GGM, [Nec94, Sho97]), since algebraic adversaries have direct access to group elements and can use their representation. In the asymmetric algebraic group model, it is assumed that, for every element π in $\mathbb{G}_1, \mathbb{G}_2$ output by the adversary, it also outputs a set of coefficients in the field that express π as a linear combination of previously received group elements in the same source group. For elements in $\mathbb{G}_T$, the adversary also outputs the coefficients that express every element output by the adversary as a linear combination of elements in $\mathbb{G}_T$ that the adversary has received or can compute as the pairing of elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ it has received.

Several works (e.g. [CHM+20]) have proven security in the AGM with random oracles. In this case, the adversary has oracle access to a certain function $H : \{0,1\}^* \to R$, and the assumption is that for every element π output by the adversary in $\mathbb{G}_1, \mathbb{G}_2$, also outputs a set of coefficients in the field that express π as a linear combination of all previously received group elements, including those obtained as a response to a hash query if the range R is a group.

Note that when when the range of the hash function R is a group, the oracle allows the adversary to sample obliviously from it, i.e. without knowing the discrete logarithm. In our case, the range of the RO is a field (of size of the order of the elliptic curve) and therefore in our model, the adversary cannot obliviously sample in the group. As discussed by Lipmaa [Lip22], we could

consider strengthening our model and give the adversary access to another oracle H_2 mapping to group elements to give this additional power to the adversary. This model is more realistic since in practice there usually exist hash to group algorithms that allow to sample in the curve without knowing the discrete logarithm.

Although the strengthened model is very meaningful and is a more realistic idealization of elliptic curves, we have not considered since it complicates the proof significantly although these additional uniformly and randomly chosen elements that are chosen independently of the input of the adversary, intuitively, cannot help the adversary except with negligible probability.[7]

Following the work of Fuchsbauer et al. [FKL18], we will prove that the security of our scheme reduces to the (q_1, q_2)-DLOG Assumption, for a certain (q_1, q_2) that depends on the size of supported instances. We note that to improve efficiency, as [Lip22] we rely on the asymmetric AGM, as opposed to the proof of Groth16 in [Gro16, FKL18].

Definition 2.6. *The (q_1, q_2)-DLOG Assumption holds relative to $\mathsf{BGgen}(1^\lambda)$ if for all PPT adversaries $\mathscr{A}$, the following probability is $\mathsf{negl}(\lambda)$,*

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{BGgen}(1^\lambda),\ z \leftarrow \mathbb{Z}_p : \\ z \leftarrow \mathscr{A}(\mathsf{gk}, \left\{[z^i]_1\right\}_{i=0}^{q_1}, \left\{[z^i]_2\right\}_{i=0}^{q_2})\end{array}\right].$$

3 SE Variant of Groth16 in the ROM

To achieve (strong) simulation extractability, the prover of Bowe and Gabizon's construction [BG18] replaces all the computations which depend on δ given in the crs by some δ' of its choice, that it must give as part of the proof, together with a proof of knowledge of the DLOG of δ' w.r.t to δ, which given some element $[Y]_1 = H([A]_1 \parallel [B]_2 \parallel [C]_1 \parallel [\delta']_2)$, consists of $[\pi]_1$ such that $e([Y]_1, [\delta']_2) = e([\pi]_1, [\delta]_2)$. In their analysis, H is an RO and their proof requires 2 pairings for verification.

In Fig. 1, we describe a SE variant of Groth16 that uses a new technique to shorten the proof and verifies it with a single verification equation which requires 3 pairings, just as Groth16. The security analysis is done in the AGM assuming the underlying hash function is a random oracle. The SE proof is built using a sequence of games. As part of the reduction we need to rewrite in the

[7] These new group elements can be seen as additional independent variables. Using the same trick as [FKL18], the multivariate case can be reduced to the univariate one by writing each new variable as a degree one polynomial of the same variable. These additional variables ultimately do not change the total degree of the polynomial that the adversary constructs as his output, which is what determines the loss in the reduction.

AGM part of the same proof as Bowe and Gabizon's construction [BG18], that is also in the random oracle but in the generic group model.

A part from the efficiency gain, from a security point of view one additional advantage of our construction is that the RO maps to elements in $\mathbb{Z}_p$ and it does not need the property that H can sample elements of $\mathbb{G}$ obliviously (i.e. soundness does not use that the DLOG of image elements is hard).

The idea of Bowe and Gabizon of using a POK of the DLOG of δ' was also used in our preliminary results presented in [BPR20], included in A. The construction we present below improves on both previous works by choosing δ' as before but then replacing it by $\delta' + \delta m$ to create and verify the proof at once, where, $m := H(\vec{x} \,\|\, [A]_1 \,\|\, [B]_2 \,\|\, [\delta']_2)$. The intuition is that the adversary needs to know the division in the exponent of C by $\delta' + \delta m$. However, this is a degree one polynomial in δ, and this is hard to do unless $\delta' = \zeta\delta$. The verification of this variant requires one additional exponentiation in $\mathbb{G}_2$. In the description of the new construction, we highlight the changes to Groth16 with gray background. We emphasize that the original scheme corresponds to $m = 0$ and $\zeta = 1$.

Theorem 3.1 (Completeness, ZK, strong SE). *The variant of* Groth16 *described in Fig. 1, is a non-interactive zero-knowledge argument that guarantees 1) perfect completeness, 2) perfect zero-knowledge and 3) strong simulation-extractability in the asymmetric Generic Group Model and the RO Model.*

Proof. To see why perfect completeness holds, the easiest is to rewrite this scheme in such a way so that the terms A, B, C correspond exactly to Groth16, except that the original term δ is replaced by $\delta' + \delta m$. The prover creates A, B with the randomizer $r_a\delta', r_b\delta'$, $r_a, r_b \leftarrow \mathbb{Z}_p$. Then, it receives m and reinterprets A, B as being created for the randomized $\delta' + \delta m$ and some random values s_a, s_b. This means the prover finds the value s_a such that $r_a\delta' = s_a(\delta' + \delta m)$. Solving the equation, we get $s_a = \dfrac{\zeta}{\zeta + m} r_a$ (similarly, $s_b = \dfrac{\zeta}{\zeta + m} r_b$). Then it computes C as in the original Groth16 paper but for s_a, s_b and $\delta' + \delta m$, instead of δ. Rewriting, we obtain:

$$[A]_1 \leftarrow \sum_{j=0}^{m} a_j [u_j(x)]_1 + [\alpha]_1 + s_a \left[\delta' + \delta m\right]_1,$$

$$[B]_2 \leftarrow \sum_{j=0}^{m} a_j [v_j(x)]_2 + [\beta]_2 + s_b \left[\delta' + \delta m\right]_2,$$

$$[C]_1 \leftarrow s_b [A]_1 + s_a [B]_1 + \sum_{j=l+1}^{m} a_j \left[(u_j(x)\beta + v_j(x)\alpha + w_j(x))/(\delta' + \delta m)\right]_1 + \left[h(x)t(x)/(\delta' + \delta m)\right]_1 - s_a s_b \left[\delta' + \delta m\right]_1.$$

Setup, $\mathsf{crs} \leftarrow \mathsf{K}(\mathbf{R}, \mathsf{z}_{\mathbf{R}})$**:** Similar to the original scheme it picks $x, \alpha, \beta, \delta \leftarrow \mathbb{Z}_p^*$, $H \leftarrow \mathscr{H}$, and returns crs defined as the following (by considering the observation in [BGM17] that γ in the original scheme can be set 1),

$$(\mathsf{crs}_{\mathsf{P}}, \mathsf{crs}_{\mathsf{V}}) := \mathsf{crs} \leftarrow \begin{pmatrix} [\alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \{u_j(x)\beta + v_j(x)\alpha + w_j(x)\}_{j=0}^{l}, \\ \left\{ \dfrac{u_j(x)\beta + v_j(x)\alpha + w_j(x)}{\delta} \right\}_{j=l+1}^{m}, \{x^i t(x)/\delta\}_{i=0}^{n-2}]_1, \\ [\beta, \delta, \{x^i\}_{i=0}^{n-1}]_2,\ [\alpha\beta]_T, H \end{pmatrix}.$$

Prover, $\pi \leftarrow \mathsf{P}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \vec{x} = (a_1, \ldots, a_l), \vec{w} = (a_{l+1}, \ldots, a_m))$**:** assuming $a_0 = 1$, it acts as follows,

1. Selects a random element $\zeta \leftarrow \mathbb{Z}_p^*$, and sets $[\delta']_2 := \zeta\,[\delta]_2$
2. Let $A^\dagger(X) \leftarrow \sum_{j=0}^{m} a_j u_j(X)$, $B^\dagger(X) \leftarrow \sum_{j=0}^{m} a_j v_j(X)$, $C^\dagger(X) \leftarrow \sum_{j=0}^{m} a_j w_j(X)$,
3. Set $h(X) = \sum_{i=0}^{n-2} h_i X^i \leftarrow (A^\dagger(X)B^\dagger(X) - C^\dagger(X))/t(X)$,
4. Set $[h(x)t(x)/\delta]_1 \leftarrow \sum_{i=0}^{n-2} h_i \left[x^i t(x)/\delta\right]_1$,
5. Set $r_a \leftarrow_r \mathbb{Z}_p$; Set $[A]_1 \leftarrow \sum_{j=0}^{m} a_j \left[u_j(x)\right]_1 + [\alpha]_1 + r_a\,[\delta']_1$,
6. Set $r_b \leftarrow_r \mathbb{Z}_p$; Set $[B]_2 \leftarrow \sum_{j=0}^{m} a_j \left[v_j(x)\right]_2 + [\beta]_2 + r_b\,[\delta']_2$,
7. Sets $m = H(\vec{x} \parallel [A]_1 \parallel [B]_2 \parallel [\delta']_2)$, where $H : \{0,1\}^* \to \mathbb{Z}_p^*$ is a secure hash function,
8. Set $s_a = \dfrac{\zeta}{\zeta + m} r_a$, $s_b = \dfrac{\zeta}{\zeta + m} r_b$

$[C]_1 \leftarrow s_b\,[A]_1 + s_a\,[B]_1 + \sum_{j=l+1}^{m} a_j \left[(u_j(x)\beta + v_j(x)\alpha + w_j(x))/\delta\,(\zeta + m)\right]_1 + \left[h(x)t(x)/(\delta\,(\zeta + m))\right]_1 - s_a s_b (\zeta + m)\,[\delta]_1$,

9. Return $\pi := ([A, C]_1, [B, \delta']_2)$.

Verifier, $\{1,0\} \leftarrow \mathsf{V}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \vec{x} = (a_1, \ldots, a_l), \pi = ([A, C]_1, [B, \delta']_2))$**:** assuming $a_0 = 1$, and setting $m = H(\vec{x} \parallel [A]_1 \parallel [B]_2 \parallel [\delta']_2)$ checks if

$$[A]_1\,[B]_2 = [\alpha\beta]_T + [C]_1\,[\delta' + \delta m]_2 + \left(\sum_{j=0}^{l} a_j \left[u_j(x)\beta + v_j(x)\alpha + w_j(x)\right]_1 \right) [1]_2$$

and return 1 if the check passes, otherwise return 0.

Simulator, $\pi \leftarrow \mathsf{Sim}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \vec{x} = (a_1, \ldots, a_l), \vec{\mathsf{ts}})$**:** Given the simulation trapdoors $\vec{\mathsf{ts}} := (\beta, \delta)$ acts as follows,

1. Choose random $\zeta \leftarrow_r \mathbb{Z}_p^*$ and set $\delta' := \zeta\delta$
2. Choose $A, B \leftarrow_r \mathbb{Z}_p$
3. Let $m = H(\vec{x} \parallel [A]_1 \parallel [B]_2 \parallel [\delta']_2)$
4. Set $[C]_1 = \left[(A \cdot B - \sum_{j=0}^{l} a_j (u_j(x)\beta + v_j(x)\alpha + w_j(x)) - \alpha\beta)/(\delta' + m\delta)\right]_1$
5. Return $\pi := ([A]_1, [B]_2, [C]_1, [\delta']_2)$

Fig. 1. A simulation-extractable variation of Groth16 for $\mathbf{R}$. $\mathscr{H}$ is a family of collision resistant hash functions that map to $\mathbb{Z}_p^*$.

Completeness easily follows from these formulae (in fact, it is identical to the completeness of Groth16 replacing δ by $\delta' + \delta m$). Similarly, perfect zero-knowledge

can be argued in a standard way. Simulation extractability is proven by reduction in the AGM to the knowledge soundness of Groth16.

Since the adversary is algebraic, for each output elements it is possible to extract a list of coefficients that express it as a linear combination of previously seen elements. The view of an adversary $\mathscr{A}$ that has made a sequence of queries $\vec{x}_1,\ldots,\vec{x}_v$ to $\mathsf{Sim}(\vec{\mathsf{ts}},\cdot)$, and received answers $\{\pi_j=([A_j,C_j]_1,[B_j,\delta_j]_2)\}_{j=1}^{v}$ is the set Q', the union of elements in the crs together with those from the replies of $\mathsf{Sim}(\vec{\mathsf{ts}},\cdot)$; namely,

$$Q' := \left(\begin{array}{l} \left[\alpha,\beta,\delta,\{x^i\}_{i=0}^{n-1},\ \{u_j(x)\beta+v_j(x)\alpha+w_j(x)\}_{j=0}^{l},\right. \\ \left.\left\{\dfrac{u_j(x)\beta+v_j(x)\alpha+w_j(x)}{\delta}\right\}_{j=l+1}^{m},\{x^i t(x)/\delta\}_{i=0}^{n-2}\right]_1, \\ [\beta,\delta,\{x^i\}_{i=0}^{n-1}]_2 \end{array}\right)$$
$$\cup\left(\left\{\left[A_j,C_j:=\tfrac{A_jB_j-\mathsf{ic}_j-\alpha\beta}{\delta_j+m_j\delta},\right]_1,[B_j,\delta_j]_2,m_j\right\}_{j=1}^{v}\right)$$

where $\mathsf{ic}_j=\sum_{i=0}^{l}a_i^j(u_i(x)\beta+v_i(x)\alpha+w_i(x))$, $\vec{x}_j=(a_1^j,\ldots,a_l^j)$, and $m_j\in\mathbb{Z}_p$ the message that simulator receives from the RO for each A_j,B_j,δ_j. Let Q'_1 be the elements of Q' in group $\mathbb{G}_1$ and Q'_2 the elements in group $\mathbb{G}_2$.

Now, assume that the adversary $\mathscr{A}$ has produced elements $\pi=([A,C]_1,[B,\delta']_2)$ that pass the verification equation. This implies that $C=(AB-\alpha\beta-\sum_{j=0}^{l}a_j(u_j(x)\beta+v_j(x)\alpha+w_j(x)))/(\delta'+m\delta)$, where $m=H(\vec{x}\parallel[A]_1\parallel[B]_2\parallel[\delta']_2)$. The coefficients extracted for output element $[Y]_i$ for $i\in\{1,2\}$ corresponding to element $q\in Q'_i$ will be denoted by $k_{Y,q}$, so that for each element Y we have that $Y=\sum_{q\in Q'_i}k_{Y,q}q$.

The reduction proceeds in a series of games, $G0,\ldots,G4$.

- G0: This is the original simulation extractability soundness game. The adversary wins if the proof $\pi=([A,C]_1,[B,\delta']_2)$ for some statement $(a_1,\ldots,a_l)$ is accepted and it is not the result of some previous query for the same statement.
- G1: This game is the same as the previous one except that it aborts if π is accepted but $k_{\delta',\delta}=-m$.
- G2: This game is the same as the previous one except that it aborts if π is accepted but for some $j=1,\ldots,v$, $\delta'=k_{\delta',\delta_j}\delta_j+k_{\delta',\delta}\delta$ and $m=m_jk_{\delta',\delta_j}-k_{\delta',\delta}$.
- G3: This game is the same as the previous one except that it aborts if if π is accepted but $\delta'\neq k_{\delta',\delta}\delta$.
- G4: This game is the same as the previous one, except that an abort occurs if π is accepted but to compute π the adversary uses any of the answers of the simulation oracle.

From G3 on, it is clear that the reduction can extract $\zeta = DLOG_{\delta}\ \delta'$ from the adversary, from which it can transform the adversary's output to a proof for Groth16 as $[A]_1, [B]_2, [C(\zeta + m)]_1$. Additionally, since in G4 the adversary does not use any of the answers to the simulation oracle, soundness in that game is implied by the knowledge soundness of Groth16.

We now proceed to bound the difference in the advantage in these games of any algebraic adversary $\mathscr{A}$. Clearly, $|\Pr[\text{G0}(\mathscr{A}) = 1] - Pr[\text{G1}(\mathscr{A}) = 1]| = |\Pr[\text{G1}(\mathscr{A}) = 1] - Pr[\text{G2}(\mathscr{A}) = 1]| = 1/p$ since the output of the random oracle is a uniform value chosen independently of the constants extracted, and the adversary can only be lucky in guessing this value with probability $1/p$.

Next we prove the following lemma:

Lemma 3.1. *For all PPT algebraic adversaries $\mathscr{A}$ there exists an adversary $\mathscr{B}$ against the $(v+2,1)$-DLOG Assumption such that*

$$Pr[G2(\mathscr{A}) = 1] \leq Pr[G3(\mathscr{A}) = 1] + \mathsf{Adv}_{\mathscr{B}}(\lambda) + \mathsf{negl}(\lambda)$$

Proof. Both games are identical except if adversary $\mathscr{A}$ outputs $\delta' \neq k_{\delta',\delta}\delta$. We show that in this case there exists another adversary $\mathscr{B}$ that breaks the $(v+2,1)$-DLOG Assumption.

Given some group key $\mathsf{gk}' = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathscr{P}'_1, \mathscr{P}_2) \leftarrow \mathsf{BGgen}(1^{\lambda})$, adversary $\mathscr{B}$ receives $\{z^i \mathscr{P}'_1\}_{i=0}^{v+2}, \{z^i \mathscr{P}_2\}_{i=0}^{1}$. It then chooses $m_1, \ldots, m_v$ random values in $\mathbb{Z}_p$. It will store these values and give them as a reply to the hash queries related to the simulation queries of $\mathscr{A}$. Next, for $j = 1, \ldots, v$, it defines

$$\delta_j = d_j z + f_j, \quad f_j, d_j \leftarrow \mathbb{Z}_p \ \text{ and } \ \delta = dz + f, \quad f, d \leftarrow \mathbb{Z}_p.$$

It programs the public parameters to compute δ and $\delta_j + m_j\delta$ roots for any j, that is, it defines the new group key included in the public parameters to be $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathscr{P}_1 = \delta \prod_{j=1}^{v} (\delta_j + m_j\delta)\mathscr{P}'_1, \mathscr{P}_2)$. This can be computed from the input of $\mathscr{B}$ since

$$\delta \prod_{j=1}^{v} (\delta_j + m_j\delta) = (dz + f) \prod_{j=1}^{v} ((d_j + m_j d)z + (f_j + m_j f))$$

is a polynomial of degree $v+1$ in the indeterminate z.

Then, adversary $\mathscr{B}$ samples $x, \alpha, \beta \leftarrow \mathbb{Z}_p$ and computes the common reference string honestly based on the new group key gk and sends all this information to $\mathscr{A}$. Note that this requires to compute some expressions involving x, α, β divided by δ but $\mathscr{B}$ can do that by computing $\delta^{-1}\mathscr{P}_1$, which is $\prod_{j=1}^{v} (\delta_j + m_j\delta)\mathscr{P}'_1 = \prod_{j=1}^{v} (d_j z + f_j + m_j)\mathscr{P}_1$. The terms in $\mathbb{G}_1$ have maximal degree $v+2$ so they can be computed by $\mathscr{B}$. Whenever $\mathscr{B}$ receives a simulation

query $\vec{x}_j$, it sets $[A_j]_1 = [\alpha]_1 + r_{a_j}[\delta_j + m_j\delta]_1$ and $[B_j]_2 = [\beta]_2 + r_{b_j}[\delta_j + m_j\delta]_2$, declares $H(\vec{x} \parallel [A]_1 \parallel [B]_2 \parallel [\delta_j])$ and computes

$$[C_j]_1 = [\frac{A_jB_j - \mathsf{ic}_j - \alpha\beta}{\delta_j + m_j\delta}]_1.$$

For this, it will use the fact that it can compute $(\delta_j + m_j\delta)^{-1}\mathscr{P}_1$ as $(dz + f)\prod_{i=1,i\neq j}^{v}(d_iz + f_i + m_i)\mathscr{P}'_1$.

If adversary $\mathscr{A}$ breaks simulation extractability for some $\vec{x} = (a_1, \ldots, a_j)$, it has produced elements (A, B, C, δ') that pass the verification equation so:

$$C = \frac{AB - \mathsf{ic} - \alpha\beta}{\delta' + m\delta}. \tag{1}$$

We now study the denominator and numerator of this expression.

For a second consider $\vec{\Delta} = (\delta, \delta_1, \ldots, \delta_v)$ as formal variables and define the polynomial

$$P_{\delta'}(\vec{\Delta}) \;=\; k_{\delta',1} + k_{\delta',\beta}\beta + k_{\delta',\delta}\delta + \sum_{i=0}^{n-1} k_{\delta',x^i}x^i + \sum_{j=1}^{v}(k_{\delta',B_j}B_j + k_{\delta',\delta_j}\delta_j).$$

The polynomial $P_B(\vec{\Delta})$ is defined analogously for the coefficients $k_{B,q}$, with $q \in Q'_2$. On the other hand, we also define $R_A(\vec{\Delta})$, $R_C(\vec{\Delta})$ in a similar way, except that the result is not a polynomial but a sum of some rational functions since the view of $\mathscr{A}$ in $\mathbb{G}_1$ includes terms that have $\delta, \delta_j + m_j\delta$ in the denominator.

If adversary $\mathscr{A}$ successfully distinguishes between the two games, $k_{\delta',\delta} \neq -m$, so $P_{\delta'}(\vec{\Delta}) + m\delta$ is a polynomial of degree one in δ. Further, there is no j such that $P_{\delta'}(\vec{\Delta}) + m\delta = \chi(\delta_j + m_j\delta)$ for some $\chi \in \mathbb{Z}_p$, since this would imply $\delta' = k_{\delta',\delta_j}\delta_j + k_{\delta',\delta}\delta$ and $m = m_jk_{\delta',\delta_j} - k_{\delta',\delta}$, which is also an abort condition. If $\mathscr{A}$ is successful in distinguishing between the two games, $P_{\delta'}(\vec{\Delta}) \neq k_{\delta',\delta}\delta$, and we are left with two possibilities:

(a)
$$R_C(\vec{\Delta}) = \frac{R_A(\vec{\Delta})P_B(\vec{\Delta}) - \mathsf{ic} - \alpha\beta}{P_{\delta'}(\vec{\Delta}) + m\delta}.$$

But this equation cannot hold, since as we argued, $P_{\delta'}(\vec{\Delta}) + m\delta$ is not a polynomial that is a multiple of δ, or $\delta_j + m_j\delta$, the only terms that appear as denominators in any term in $R_C(\vec{\Delta})$.

(b) otherwise,

$$R_C(\vec{\Delta})(P_{\delta'}(\vec{\Delta}) + m\delta) - R_A(\vec{\Delta})P_B(\vec{\Delta}) + \mathsf{ic} + \alpha\beta \neq 0.$$

Define

$$T(\vec{\Delta}) = \delta \prod_{j=1}^{v} (\delta_j + m_j\delta)\Big(R_C(\vec{\Delta})(P_{\delta'}(\vec{\Delta}) + m\delta) \\ - R_A(\vec{\Delta})P_B(\vec{\Delta}) + \mathsf{ic} + \alpha\beta\Big).$$

Note that this is a polynomial in $\vec{\Delta}$, since $\delta\prod_{j=1}^{v}(\delta_j + m_j\delta)$ cancels out any of the denominators that appear in the terms in $R_A(\vec{\delta})$. Replacing $\delta = dZ + f$ and $\delta_j = d_jZ + f_j$ in T we get a polynomial that depends on a single variable $T'(Z)$. Since $C = \frac{AB - \mathsf{ic} - \alpha\beta}{\delta' + m\delta}$, $T'(z) = 0$. On the other hand, $T'(Z) \neq 0$ except with probability $1/p$. This is justified as follows: if $T'(Z)$ was 0 all its coefficients must be 0. In particular, take the leading terms in Z of $T'(Z)$: this is an expression involving only d, d_j, which are information theoretically hidden from $\mathscr{A}$. If we think of this polynomial as a multivariate one of total degree $v+3$ in variables d, d_j, the probability that $\mathscr{A}$ chooses the coefficients $k_{A,q}, k_{B,q}, k_{\delta',q}, k_{C,q}$ such that when evaluated in d, d_j this polynomial is 0 can be bounded by $(v+3)/p$. Therefore, $\mathscr{B}$ can solve the DLOG challenge by factoring T' and trying all the possible roots.

Lemma 3.2. *For all PPT algebraic adversaries $\mathscr{A}$ there exists an adversary $\mathscr{B}$ against the $(v+2,1)$-DLOG Assumption such that*

$$Pr[G3(\mathscr{A}) = 1] \leq Pr[G4(\mathscr{A}) = 1] + \mathsf{Adv}_{\mathscr{B}}(\lambda) + \mathsf{negl}(\lambda)$$

Proof. Both games are identical except if adversary $\mathscr{A}$ outputs a accepting proof that is built using the output of some simulation query. We show that in this case there exists another adversary $\mathscr{B}$ that breaks the $(v+2,1)$-DLOG Assumption.

Given some group key $\mathsf{gk}' = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathscr{P}'_1, \mathscr{P}_2) \leftarrow \mathsf{BGgen}(1^\lambda)$, adversary $\mathscr{B}$ receives $\{z^i\mathscr{P}'_1\}_{i=0}^{v+1}, \{z^i\mathscr{P}_2\}_{i=0}^{1}$. It then chooses $m_1, \ldots, m_v$ random values in $\mathbb{Z}_p$. It will store these values and give them as a reply to the hash queries related to the simulation queries of $\mathscr{A}$. Next, for $j = 1, \ldots, v$, it defines

$$\alpha = d_\alpha z + f_\alpha,\ f_\alpha, d_\alpha \leftarrow \mathbb{Z}_p \qquad \beta = d_\beta z + f_\beta,\ f_\beta, d_\beta \leftarrow \mathbb{Z}_p,$$

and, as in the previous lemma:

$$\delta_j = d_j z + f_j,\ \ f_j, d_j \leftarrow \mathbb{Z}_p \ \text{ and } \ \delta = dz + f,\ \ f, d \leftarrow \mathbb{Z}_p.$$

It programs the public parameters to compute δ and $\delta_j + m_j\delta$ roots for any j, that is, it defines the new group key included in the public parameters to be

$\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathscr{P}_1 = \delta \prod_{j=1}^{v} (\delta_j + m_j \delta) \mathscr{P}'_1, \mathscr{P}_2)$. This can be computed from the input of $\mathscr{B}$ since

$$\delta \prod_{j=1}^{v} (\delta_j + m_j \delta) = (dz + f) \prod_{j=1}^{v} ((d_j + m_j d)z + (f_j + m_j f))$$

is a polynomial of degree $v+1$ in the indeterminate z.

Then, adversary $\mathscr{B}$ samples $x \leftarrow \mathbb{Z}_p$ and computes the common reference string honestly based on the new group key gk and sends all this information to $\mathscr{A}$. This can be computed from $\mathscr{B}$'s input since these requires to compute polynomials of degree at most 2 in z in each source group.

Whenever $\mathscr{B}$ receives a simulation query $\vec{x}_j$, it samples $\zeta_j, f_{A,j}, d_{A,j}, f_{B,j}, d_{B,j} \leftarrow \mathbb{Z}_p$, and sets

$$A_j = d_{A,j} z + f_{A,j} \qquad B_j = d_{B,j} z + f_{B,j} \qquad \delta_j = \zeta_j \delta,$$

declares $m_j = H(\vec{x}_j \parallel [A_j]_1 \parallel [B_j]_2 \parallel [\delta_j])$ and computes

$$[C_j]_1 = [\frac{A_j B_j - \mathsf{ic}_j - \alpha\beta}{\delta_j + m_j \delta}]_1.$$

For this, it uses the fact that it can compute $\delta_j + m_j \delta_j$ roots in $\mathbb{G}_1$. If adversary $\mathscr{A}$ distinguishes between both games, it outputs some $\vec{x} = (a_1, \ldots, a_j)$, and (A, B, C, δ') that pass the verification equation and, further, it is possible to extract some ζ such that $\delta' + m\delta = (\zeta + m)\delta$, therefore it holds that:

$$C\delta(\zeta + m) - AB + \mathsf{ic} + \alpha\beta = 0. \tag{2}$$

For a second, consider

$$\vec{Y} := (\alpha, \beta, \delta, \delta_1, \ldots, \delta_v, A_1, \ldots, A_v, B_1, \ldots, B_v)$$

as formal variables. Define the polynomial

$$P_B(\vec{Y}) \;=\; k_{B,1} \;+\; k_{B,\beta}\beta \;+\; k_{B,\delta}\delta \;+\; \sum_{i=0}^{n-1} k_{B,x^i} x^i \;+\; \sum_{j=1}^{v} (k_{B,B_j} B_j \;+\; k_{B,\delta_j}.\delta_j).$$

Define $R_A(\vec{Y}), R_C(\vec{Y})$ in a similar way, with the coefficients $k_{A,q}, k_{C,q}$, $q \in Q'_1$ extracted from the adversary, except that the result is not a polynomial but a sum of some rational functions since the view of $\mathscr{A}$ in $\mathbb{G}_1$ includes terms that have δ or $\delta_j + m_j\delta$ in the denominator. Note that $P_A(\vec{Y}) := \delta \prod_{j=1}^{v} (\delta_j + m_j \delta) R_A(\vec{Y})$, $P_C(\vec{Y}) := \delta \prod_{j=1}^{v} (\delta_j + m_j \delta) R_A(\vec{Y})$ are polynomials in $\vec{Y}$ of degree $v+2$ since

all possible denominators are cancelled out. Multiplying on both sides of equation (2) by $\delta\prod_{j=1}^{v}(\delta_j + m_j\delta)$, and replacing each group element by the corresponding polynomial, we get the following polynomial:

$$T(\vec{Y}) = P_C(\vec{Y})(\zeta + m)\delta - P_A(\vec{Y})P_B(\vec{Y}) + \delta\prod_{j=1}^{v}(\delta_j + m_j\delta)\mathsf{ic}(\vec{Y}) + \delta\prod_{j=1}^{v}(\delta_j + m_j\delta)\alpha\beta. \quad (3)$$

If adversary $\mathscr{A}$ distinguishes between the two games, there is at least one coefficient of $P_C(\vec{Y})$ or $P_A(\vec{Y})$ accompanying A_j or C_j which is not zero, or at least one coefficient of $P_B(\vec{Y})$ accompanying δ_j or B_j which is not zero. We show that this implies in all cases that $T(\vec{Y}) \neq 0$.

We start by arguing that $k_{A,\alpha} = 1$ and $k_{B,\beta} = 1$, since otherwise the term $\alpha\beta$ in equation (3) cannot be cancelled out. In other words, $R_A(\vec{Y}) = \alpha + \ldots$ and $P_B(\vec{Y}) = \beta + \ldots$, so $P_A(\vec{Y}) = \delta\prod_{j=1}^{v}(\delta_j + m_j\delta)\alpha + \ldots$. We next argue all cases of interest separately:

(a) If the coefficient $k_{B,\delta_j} \neq 0$ for some j, then in $P_A(\vec{Y})P_B(\vec{Y})$ the coefficient of $\alpha\delta\prod_{j=1}^{v}(\delta_j + m_j\delta)\delta_j$ is k_{B,δ_j} but it is 0 for the rest of the terms (P_C can have no δ_j terms because the group is asymmetric, $\mathsf{ic}(\vec{Y})$ does not have δ_j terms by definition and the last term has no monomials without β). Therefore, the coefficient of this polynomial is not zero and $T'(\vec{Y}) \neq 0$.

(b) Similarly, if the coefficient $k_{B,B_j} \neq 0$ for some j, then in $P_A(\vec{Y})P_B(\vec{Y})$ the coefficient of $\alpha\delta\prod_{j=1}^{v}(\delta_j + m_j\delta)B_j$ is k_{B,B_j}, while in the other terms it is 0, in which case $T'(\vec{Y}) \neq 0$.

(c) If the coefficient $k_{A,A_j} \neq 0$ for some j, then in $P_A(\vec{Y})P_B(\vec{Y})$ the coefficient of monomial $A_j\beta$ is k_{A,A_j}, while in the other terms it is 0 (because in P_C there can be no β term and $\mathsf{ic}(\vec{Y})$ does not have A_j terms by definition). Therefore, $T'(\vec{Y}) \neq 0$.

(d) If the coefficient $k_{A,C_j} \neq 0$ for some j, the analysis is the same as in (b). Therefore, $T'(\vec{Y}) \neq 0$.

(e) If the coefficient $k_{C,A_j} \neq 0$, the only term with A_j would be $P_C(\vec{Y})(\zeta_j + m)\delta$ since we ruled out case (c). Therefore, $T'(\vec{Y}) \neq 0$.

(f) If the coefficient $k_{C,C_j} \neq 0$, the only term with C_j would be $P_C(\vec{Y})(\zeta_j + m)\delta$ since we ruled out case (d). Therefore, $T'(\vec{Y}) \neq 0$.

Finally, we show that if $T(\vec{Y}) \neq 0$, there exists an adversary against the $(v+2,1)$-DLOG Assumption. Indeed, suppose that $T(\vec{Y}) \neq 0$. Define the univariate polynomial $T'(Z)$ as the result of substituting each variable in $\vec{Y}$ by an expression in the same indeterminate Z, as $\alpha = d_\alpha Z + f_\alpha, \beta = d_\beta Z + f_\beta, \delta_j =$

$d_jZ + f_j, \delta = dZ + f$. If $T'(Z) \neq 0$ is not zero and we know from expression (2) that $T'(z) = 0$, adversary $\mathscr{B}$ can find z by factoring T', solving the DLOG challenge. On the other hand, to argue that $T'(Z) \neq 0$ except with probability $(v+3)/p$, we resort to the same argument as in the last step of Lemma 3.1.

This concludes the reduction to the knowledge soundness of Groth16, that was reduced in the symmetric AGM to the $(2n-1)$-DLOG Assumption.

4 Empirical Analysis

We evaluate the efficiency of our presented simulation extractable variants of Groth's zk-SNARK using a prototype implementation in Arkworks [8] which is an ecosystem written in Rust for developing and programming with zk-SNARKs. A prototype implementation of both Groth16 [Gro16] and Groth and Maller's zk-SNARK [GM17] are already presented in Arkworks library, and in order to obtain a fair comparison and a comprehensive outcome, we also present an efficient implementation of Bowe and Gabizon's construction [BG18] and our initial construction [BPR20] in the same library [9]

Our empirical analysis are done with the elliptic curves BLS12-381, MNT4-298, MNT6-298, MNT4-753 and MNT6-753 that BLS12-381 is estimated to achieve between 117 and 120 bits security [NCC19], and the other four curves are estimated to achieve respectively 2^{77}, 2^{87}, 2^{113}, 2^{137} security [BCTV14]. All experiments are done on a desktop machine with Ubuntu 20.4.2 LTS, an Intel Core i9-9900 processor at base frequency 3.1 GHz, and 128GB of memory. Proof generations are done in the multi-thread mode, with 16 threads, while proof verifications are done in a single-thread mode.

Following the benchmark strategy in Arkworks library, we report *Per-Constraint Proving Time* (PCPT) and *verification time* for both the proposed constructions in Sections 3 and A and compare their efficiency with (weak or strong) SE zk-SNARKs of Groth16 [Gro16], Groth-Maller (GM17) [GM17] and Bowe-Gabizon (BG18) [BG18]. Motivated by blockchain and large-scale applications like Zcash [BCG+14], we also compare (deterministic) verifying time of all constructions for the case that one needs to verify a large number of proofs for a particular language simultaneously. In the verification step of our constructions, one needs to compute exponentiation in $\mathbb{G}_2$ and $\mathbb{G}_T$, which can be optimized by Multi-Scalar Multiplication (MSM) techniques.

Tab. 2 presents an empirical analysis of our constructions and compares them with several relevant SE zk-SNARKs for an R1CS instance with 400.000

[8] Available on https://github.com/arkworks-rs

[9] Source codes of our implementations are publicly available on: https://github.com/Baghery/ABPR22.

Table 2. A comparison of practical efficiency of our proposed variants of Groth16 along with the relevant SE zk-SNARKs for arithmetic circuit satisfiability. We report average per-constrain proving time and verification time of $1, 10^2$ and 10^3 proofs for all zk-SNARKs with several elliptic curves. The benchmarks are done with an R1CS instance with 400.000 constrains and 10 input values, and the average of proving times are taken for 100 iterations and the verification for 10^3 iterations. Proof generation is done in multi-thread setting with 16 threads, while the verification is done in the single-thread setting. EC: Elliptic Curve, SE: Simulation Extractability, PCPT: Per-Constraint Proving Time, Ver.: Verifying, ns: nanosecond, ms: millisecond, s: seconds, B: Byte, WSE: Weak Simulation Extractable, SSE: Strong Simulation Extractable, AGM: Algebraic Group Model, GGM: Generic Group Model, RO: Random Oracle, CRH: Collision Resistant Hash. Among the strong SE ones, we have highlighted the most efficient verification.

EC	zk-SNARK	SE	Model	PCPT (ns)	Proof Size (B)	Ver. 1 Proof	Ver. 10^2 Proofs	Ver. 10^3 Proofs
BLS12-381	[Gro16, BKSV21]	WSE	AGM	5026	127.5	1.90 ms	0.190 s	1.90 s
	[GM17]	SSE	GGM	11042	127.5	3.32 ms	0.332 s	3.32 s
	[BG18]	SSE	GGM, RO	5052	223.1	3.52 ms	0.352 s	3.52 s
	[BPR20], App. A	SSE	GGM, CRH	5042	223.1	4.85 ms	0.360 s	3.50 s
	Section 3	SSE	AGM, RO	5041	191.2	2.39 ms	0.194 s	1.91 s
MNT4-298	[Gro16, BKSV21]	WSE	AGM	4830	149.0	2.67 ms	0.267 s	2.67 s
	[GM17]	SSE	GGM	10025	149.0	3.80 ms	0.380 s	3.80 s
	[BG18]	SSE	GGM, RO	4879	260.7	4.32 ms	0.432 s	4.32 s
	[BPR20], App. A	SSE	GGM, CRH	4881	260.7	4.45 ms	0.311 s	3.05 s
	Section 3	SSE	AGM, RO	4875	223.5	3.33 ms	0.271 s	2.68 s
MNT6-298	[Gro16, BKSV21]	WSE	AGM	5794	186.2	4.94 ms	0.494 s	4.94 s
	[GM17]	SSE	GGM	11427	186.2	7.07 ms	0.707 s	7.07 s
	[BG18]	SSE	GGM, RO	5831	335.2	8.07 ms	0.807 s	8.07 s
	[BPR20], App. A	SSE	GGM, CRH	5824	335.2	8.34 ms	0.582 s	5.72 s
	Section 3	SSE	AGM, RO	5810	298.0	6.11 ms	0.501 s	4.97 s
MNT4-753	[Gro16, BKSV21]	WSE	AGM	30247	376.5	29.1 ms	2.91 s	29.1 s
	[GM17]	SSE	GGM	83120	376.5	41.6 ms	4.16 s	41.6 s
	[BG18]	SSE	GGM, RO	30863	658.8	47.3 ms	4.73 s	47.3 s
	[BPR20], App. A	SSE	GGM, CRH	30887	658.8	45.5 ms	3.41 s	33.8 s
	Section 3	SSE	AGM, RO	30760	564.7	33.9 ms	2.94 s	29.2 s
MNT6-753	[Gro16, BKSV21]	WSE	AGM	33298	470.6	53.6 ms	5.36 s	53.6 s
	[GM17]	SSE	GGM	83121	470.6	76.9 ms	7.69 s	76.9 s
	[BG18]	SSE	GGM, RO	33358	847.1	88.5 ms	8.85 s	88.5 s
	[BPR20], App. A	SSE	GGM, CRH	33359	847.1	85.4 ms	6.33 s	63.1 s
	Section 3	SSE	AGM, RO	33345	753.0	64.4 ms	5.42 s	53.8 s

constraints and 10 input variables. The reported times are the average values on 100 iterations for proof generation and 10.000 iterations for verification. As it can be seen, similar to BG18 construction [BG18], provers of our constructions are almost as efficient as Groth's protocol, while due to a different NP characterization, the GM17 scheme is considerably less efficient in comparison with other schemes. For instance, to generate a proof for an arithmetic circuit with 400.000 constraints, with BLS12-381 curve, Groth16, BG18, and both of our constructions require $\approx$ 2.01 seconds, while GM17 needs $\approx$ 4.41 seconds.

Among the compared strong SE constructions, GM17 has the shortest proof size, namely 2 elements from $\mathbb{G}_1$ and 1 element from $\mathbb{G}_2$, and our construction in Section 3 has the second shortest proof size, namely 2 elements from $\mathbb{G}_1$ and 2 elements from $\mathbb{G}_2$.

In the last two columns of Tab. 2, we report the verification time of all constructions for the case that we need to verify 10^2 or 10^3 proofs of the same language. Once verifying a large number of proofs, our constructions use the MSM technique to compute the needed exponentiations in all proofs at the same time, which allows us to save on total verification time. As it can be seen, our construction presented in Section 3 has the most efficient verification among the strong SE constructions, and above all in the case of verifying a large number of proofs, the total verification time in both of our constructions improve significantly using the MSM technique. In particular, the verification of our second construction has very close efficiency to the original Groth16. For instance, in the case of BLS12-381, once we verify 100 proofs, the total verification time for Groth16 is ≈ 0.190 seconds, and for our second construction is ≈ 0.194. As it can be seen the gap is small and actually the larger the number of proofs we verify, the smaller this gap gets.

5 Conclusion

Over the last few years, various SE zk-SNARKs have been proposed that achieve (strong) simulation extractability [GM17, BG18, AB19, Lip22], which is a security property stronger than knowledge soundness and prevents attacks from the adversaries who have seen simulated proofs. Simulation extractability implies non-malleability of proofs [GM17] and its variant with *black-box extraction* is shown to be sufficient for achieving UC-security in NIZK arguments [Gro06]. SE zk-SNARKs allow us to build succinct signature-of-knowledge schemes [CL06, GM17], and they can also be used to build chameleon hash functions [KDS20].

In this paper, we revised the SE variation of Groth16 proposed in [BG18] and presented a new variation. Our initial construction from CANS 2020 ([BPR20], A) requires 4 pairings in verification, instead of 5 in [BG18], and also avoids random oracles in exchange for using a collision resistant hash function. It has a more efficient prover, crs size, and proof size in comparison with [AB19], that has also 4 pairings in the verification. Our new variant used some subtle modifications to shorten the proof size and improved the verification of Bowe and Gabizon's construction significantly [BG18]. In this variant, we showed that using a random oracle, we can achieve strong SE in Groth16, at the cost of one additional $\mathbb{G}_2$ element in the proof, and one new exponentiation in $\mathbb{G}_2$ in the

verification, where the later introduces negligible overhead to the verification of Groth16 in the cases that one needs to verify a large number of proofs for the same circuit (e.g. Zcash [BCG+14]). We evaluated the empirical performance of our constructions in Arkworks library. Our evaluations showed that our constructions are among the most efficient SE zk-SNARKs. Particularly, in large-scale applications, the CRS, the prover, and the verifier of our new SE zk-SNARK are almost as efficient as the original Groth16. Just, in our case the proof consists of 4 group elements, instead of 3 in the original construction of Groth16. This seems to be a minimal cost to achieve *strong* SE in Groth16.

Acknowledgments

We thank Alonso González for his helpful discussions and the reviewers for their valuable comments. Karim Baghery has been supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract No. HR001120C0085, by the FWO under an Odysseus project GOH9718N, and by CyberSecurity Research Flanders with reference number VR20192203. Carla Ràfols was partially supported by Project RTI2018-102112-B-I00 (AEI/FEDER, UE).

Data Availability: All data generated or analyzed during this study are included in this paper and the source codes of implementations are publicly available on https://github.com/Baghery/ABPR22.

Bibliography

[AB19] Shahla Atapoor and Karim Baghery. Simulation extractability in Groth's zk-SNARK. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2019 International Workshops, DPM 2019 and CBT 2019*, volume 11737 of *LNCS*, pages 336–354. Springer, 2019.

[ABLZ17] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2017.

[Bag19a] Karim Baghery. On the efficiency of privacy-preserving smart contract systems. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 118–136. Springer, Heidelberg, July 2019.

[Bag19b] Karim Baghery. Subversion-resistant simulation (knowledge) sound NIZKs. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, pages 42–63. Springer, Heidelberg, December 2019.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[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*, pages 459–474. IEEE Computer Society Press, May 2014.

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 276–294. Springer, Heidelberg, August 2014.

[BFG13] David Bernhard, Georg Fuchsbauer, and Essam Ghadafi. Efficient signatures of knowledge and DAA in the standard model. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 518–533. Springer, Heidelberg, June 2013.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

[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.

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

[BGPR20] Karim Baghery, Alonso González, Zaira Pindado, and Carla Ràfols. Signatures of knowledge for boolean circuits under standard assumptions. In Abderrahmane Nitaj and Amr M. Youssef, editors, *AFRICACRYPT 20*, volume 12174 of *LNCS*, pages 24–44. Springer, Heidelberg, July 2020.

[BKSV21] Karim Baghery, Markulf Kohlweiss, Janno Siim, and Mikhail Volkhov. Another look at extraction and randomization of groth's zk-SNARK. In Nikita Borisov and Claudia Díaz, editors, *FC 2021, Part I*, volume 12674 of *LNCS*, pages 457–475. Springer, Heidelberg, March 2021.

[BMRS20] Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. Coda: Decentralized cryptocurrency at scale. Cryptology ePrint Archive, Report 2020/352, 2020. https://eprint.iacr.org/2020/352.

[BPR20] Karim Baghery, Zaira Pindado, and Carla Ràfols. Simulation extractable versions of groth's zk-SNARK revisited. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 453–461. Springer, Heidelberg, December 2020.

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

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, August 2006.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexan-

dra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

[Fuc18] Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018.

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

[GM17] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, August 2017.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006.

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

[KDS20] Mojtaba Khalili, Mohammad Dakhilalian, and Willy Susilo. Efficient chameleon hash functions in the enhanced collision resistant model. *Inf. Sci.*, 510:155–164, 2020.

[KKKZ19] Thomas Kerber, Aggelos Kiayias, Markulf Kohlweiss, and Vassilis Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *2019 IEEE Symposium on Security and Privacy*, pages 157–174. IEEE Computer Society Press, May 2019.

[KLO20] Jihye Kim, Jiwon Lee, and Hyunok Oh. Simulation-extractable zk-snark with a single verification. *IEEE Access*, 8:156569–156581, 2020.

[KMS+16] Ahmed E. 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*, pages 839–858. IEEE Computer Society Press, May 2016.

[KZM+15] Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and

Elaine Shi. C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093, 2015. https://eprint.iacr.org/2015/1093.

[Lip22] Helger Lipmaa. A unified framework for non-universal SNARKs. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 553–583. Springer, Heidelberg, March 2022.

[NCC19] NCC. Zcash overwinter consensus and sapling cryptography review. https://research.nccgroup.com/wp-content/uploads/2020/07/NCC_Group_Zcash2018_Public_Report_2019-01-30_v1.3.pdf, 2019.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. Mathematical Notes, 55(2):165–172, 1994.

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

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

A A Simulation Extractable zk-SNARK without RO

In this section, we recall the construction of our first (strong) SE variant of Groth16 based on Bowe and Gabizon's scheme [BG18] which is presented in [BPR20].

Scheme Definition. In Fig. 2, we recall the construction of our first variation of Groth16 [BPR20], that similarly works with quadratic arithmetic programs. In this construction, the Proof of Knowledge (PoK) of the DLOG of $[\delta']_2$ w.r.t. $[\delta]_2$ is changed to another PoK in the GGM that relies on the collision resistance property of the hash function. In Fig. 2, the elements $[\alpha\beta, t(x), \gamma t(x)]_T$ are redundant and can in fact be computed from the rest of the elements in the crs. Alternatively, one can describe Groth16 as corresponding to $\zeta = 1, \gamma = 0$ and where the proof consists only of $[A, C]_1, [B]_2$. Differences with Groth16 are highlighted. We briefly give an intuition behind the scheme in the following.

Avoiding Random Oracle. In [BPR20], it is proven that the variation of Groth16 described in Fig. 2, guarantees (1) perfect completeness, 2) perfect zero-knowledge

and 3) simulation-extractability in the asymmetric GGM. The proof of construction uses the collision resistance property of the hash function and the GGM. Roughly speaking, the new variable γ gives some additional guarantees because to compute $t(x)\frac{(\gamma+m)}{(\delta'+\delta m)}$ from D_j such that $m_j \neq m$, it is necessary to know both $\frac{1}{(\delta'+\delta m)}$ and $\frac{\gamma}{(\delta'+\delta m)}$, but this is only possible when $\delta' + \delta m = k\delta$. Then, either one has the knowledge of the DLOG of δ' respect to δ $(k-m)$, which is straightforward, or either one has re-used δ'_j and m_j from some jth query. The last case is discarded when one reaches that same message had to be re-used, $m = m_j$, which breaks collision resistance of the hash.

Setup, $\mathsf{crs} \leftarrow \mathsf{K}(\mathbf{R}, \mathsf{z}_{\mathbf{R}})$**:** Similar to the original scheme pick $x, \alpha, \beta, \delta, \gamma \leftarrow \mathbb{Z}_p^*$, $H \leftarrow \mathscr{H}$, and returns crs defined as the following,

$$(\mathsf{crs}_{\mathsf{P}}, \mathsf{crs}_{\mathsf{V}}) := \mathsf{crs} \leftarrow \left(\begin{array}{l} [\alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \{u_j(x)\beta + v_j(x)\alpha + w_j(x)\}_{j=0}^{l}, \\ \frac{\gamma t(x)}{\delta}, \left\{ \frac{u_j(x)\beta + v_j(x)\alpha + w_j(x)}{\delta} \right\}_{j=l+1}^{m}, \\ \left\{ \frac{x^i t(x)}{\delta} \right\}_{i=0}^{n-2}]_1, [\beta, \delta, \{x^i\}_{i=0}^{n-1}]_2, \ [\alpha\beta, t(x), \gamma t(x)]_T, H \end{array} \right).$$

Prover, $\pi \leftarrow \mathsf{P}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \vec{x} = (a_1, \ldots, a_l), \vec{w} = (a_{l+1}, \ldots, a_m))$**:** assuming $a_0 = 1$, it acts as follows,

1. Selects a random element $\zeta \leftarrow \mathbb{Z}_p^*$, and sets $[\delta']_2 := \zeta [\delta]_2$
2. Let $A^\dagger(X) \leftarrow \sum_{j=0}^{m} a_j u_j(X)$, $B^\dagger(X) \leftarrow \sum_{j=0}^{m} a_j v_j(X)$,
3. Let $C^\dagger(X) \leftarrow \sum_{j=0}^{m} a_j w_j(X)$,
4. Set $h(X) = \sum_{i=0}^{n-2} h_i X^i \leftarrow (A^\dagger(X)B^\dagger(X) - C^\dagger(X))/t(X)$,
5. Set $[h(x)t(x)/\delta']_1 \leftarrow (1/\zeta)(\sum_{i=0}^{n-2} h_i [x^i t(x)/\delta]_1)$,
6. Set $r_a \leftarrow_r \mathbb{Z}_p$; Set $[A]_1 \leftarrow \sum_{j=0}^{m} a_j [u_j(x)]_1 + [\alpha]_1 + r_a [\delta']_1$,
7. Set $r_b \leftarrow_r \mathbb{Z}_p$; Set $[B]_2 \leftarrow \sum_{j=0}^{m} a_j [v_j(x)]_2 + [\beta]_2 + r_b [\delta']_2$,
8. Set $[C]_1 \leftarrow r_b [A]_1 + r_a \left(\sum_{j=0}^{m} a_j [v_j(x)]_1 + [\beta]_1 \right) + (1/\zeta) \sum_{j=l+1}^{m} a_j ([(u_j(x)\beta + v_j(x)\alpha + w_j(x))/\delta]_1) + [h(x)t(x)/\delta']_1$,
9. Set $m = H([A]_1 \parallel [B]_2 \parallel [C]_1 \parallel [\delta']_2)$, where $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a secure hash function,
10. Compute $[D]_1 = \frac{m}{\zeta + m} \left[\frac{t(x)}{\delta} \right]_1 + \frac{1}{\zeta + m} \left[\frac{\gamma t(x)}{\delta} \right]_1 = \left[\frac{(m+\gamma)t(x)}{\delta' + m\delta} \right]_1$
11. Return $\pi := ([A, C, D]_1, [B, \delta']_2)$.

Verifier, $\{1,0\} \leftarrow \mathsf{V}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \vec{x} = (a_1, \ldots, a_l), \pi = ([A, C, D]_1, [B, \delta']_2))$**:** assuming $a_0 = 1$, and setting $m = H([A]_1 \parallel [B]_2 \parallel [C]_1 \parallel [\delta']_2)$ checks if

1. $[A]_1 [B]_2 = [C]_1 [\delta']_2 + \left(\sum_{j=0}^{l} a_j [u_j(x)\beta + v_j(x)\alpha + w_j(x)]_1 \right) [1]_2 + [\alpha\beta]_T$
2. $[D]_1 [\delta' + \delta m]_2 = m[t(x)]_T + [\gamma t(x)]_T$ (Note that $[t(x)]_T$ and $[\gamma t(x)]_T$ are added to the CRS)

and returns 1 if both checks pass, otherwise return 0.

Simulator, $\pi \leftarrow \mathsf{Sim}(\mathbf{R}, \mathsf{z}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \vec{x} = (a_1, \ldots, a_l), \vec{\mathsf{ts}})$**:** Given the simulation trapdoors $\vec{\mathsf{ts}} := (\beta, \delta)$ acts as follows,

1. Choose random $\zeta \leftarrow_r \mathbb{Z}_p^*$ and set $\delta' := \zeta\delta$
2. Choose $A, B \leftarrow_r \mathbb{Z}_p$
3. Let $[C]_1 = [(A \cdot B - \sum_{j=0}^{l} a_j (u_j(x)\beta + v_j(x)\alpha + w_j(x)) - \alpha\beta)/\delta']_1$
4. Let $m = H([A]_1 \parallel [B]_2 \parallel [C]_1 \parallel [\delta']_2)$
5. $[D]_1 = \frac{m}{\zeta + m} \left[\frac{t(x)}{\delta} \right]_1 + \frac{1}{\zeta + m} \left[\frac{\gamma t(x)}{\delta} \right]_1 = \left[\frac{(m+\gamma)t(x)}{\delta' + m\delta} \right]_1$
6. Return $\pi := ([A, C, D]_1, [B, \delta']_2)$.

Fig. 2. Our initial strong SE variant of Groth16 for $\mathbf{R}$ along with a modification of the Boneh-Boyen signature. In the protocol, $\mathscr{H}$ is a family of collision resistant hash functions that map to $\mathbb{Z}_p^*$ [BPR20].