Simulation Extractable Versions of Groth's zk-SNARK Revisited

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

[1] University of Oslo, Oslo, Norway,
oussamaa@math.uio.no
[2] imec-COSIC, KU Leuven, Leuven, Belgium,
karim.baghery@kuleuven.be
[3] Universitat Pompeu Fabra, Barcelona, Spain,
zaira.pindado@upf.edu,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 construction 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 two efficient variants. In the first variant, we show that one can save 1 pairing in the verification and also relax the RO to a collision-resistant hash function. This is achieved at the cost of a single new element in the common reference string, and one exponentiation in $\mathbb{G}_T$ in the verification. In the second variant, we focus on the efficiency and present an SE zk-SNARK that has a minimal overhead on Groth16. Namely, it adds only 1 element to the proof and 1 exponentiation in $\mathbb{G}_2$ to the verification of Groth16. We implement our proposed SE zk-SNARKs along with BG18 in the Arkworks library, and compare the efficiency of our schemes with some related works. Our empirical experiences confirm that our second SE zk-SNARK is more efficient than all previous SE schemes in most dimensions and it has very close efficiency to the original Groth16 [4].

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

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

1 Introduction

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

sion) 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 [Lip19], 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.

Recently, Baghery, Kohlweiss, Siim, and Volkhov [BKSV20] 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 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, BKSV20], 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 efficiency and get the best of both constructions. Namely, achieving *strong* simulation extractability in Groth16 with minimal overhead and with the minimum additional assumption.

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, Lip19, AB19], 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 two *strong* SE zk-SNARKs that are among the most efficient (strong) simulation extractable variants of Groth16 in terms of crs size, prover complexity, and verification time. Both of our SE zk-SNARKs use some sophisticated modification of Boneh-Boyen signatures [BB08] to prove knowledge of the DLOG of δ' which require 1 or 2 pairings less in the verification in comparison with the argument of Bowe and Gabizon [BG18], but at the cost of one or two additional exponentiations in the verification. In the first variant, we mainly focus on mitigating the need for a random oracle without considerably sacrificing the efficiency, while in the second one, we concentrate on achieving the best practical efficiency.

In the first construction, we get rid of the RO in Bowe and Gabizon's variant [BG18] and similar to Groth16, prove the security of construction in the GGM model. This is achieved at the cost of a single new element in the crs, a collision-resistant hash function, and an exponentiation in $\mathbb{G}_T$ in the verification. In the second construction, using a non-programmable RO, we modify the proof generation of Groth16 slightly and include the PoK of the DLOG of $[\delta']_2$

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, 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]	WSE	GGM	$m+2n-l$ $\mathbb{G}_1$ 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
[Lip19]	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
Sec. 3	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
Sec. 4	SSE	GGM, 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

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 cases of verifying a larger number of proofs, the verifiers of our constructions gain efficiency by using *multi-scalar exponentiations*, particularly our second construction achieves almost the same efficiency as Groth16.

Tab. 1 presents a comparison of our proposed variants 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 first construction retains most of the properties, requires 1 less pairing in the verification, and relaxes the RO to a collision resistant hash function, at the cost of 2 additional exponentiations in the verification. On the other hand, our second construction saves 1 $\mathbb{G}_1$ element in the proof, and requires 2 less pairings in the verification, which results in the most efficient strong SE variant of Groth16 in the most dimensions. 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, and above all, our second construction requires 3 parings in the verification, instead of 4. [5]

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 5, in Table 2. As we expected, the implementation results show that, our second 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

[5] 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].

Groth16. Therefore, one can use the same compiler or ad-hoc approach proposed in [Bag19a] and [BKSV20], 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 give our first SE zk-SNARK 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. In Section 4, we present our second 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 5 using a prototype Rust implementation in Arkworks library. 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 6.

Novelty. Additional to the conference version published in CANS 2020 [BPR20], this version includes a more efficient construction presented in Section 4, 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 5.

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 $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathcal{P}_1, \mathcal{P}_2)$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order p, the elements $\mathcal{P}_1, \mathcal{P}_2$ 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 $[a]_i = a\mathcal{P}_i$, where $i \in \{1, 2, T\}$ and $\mathcal{P}_T = e(\mathcal{P}_1, \mathcal{P}_2)$. 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 algorithms $\mathcal{A}$ and $\mathsf{Ext}_\mathcal{A}$, we write $(y \,\|\, y') \leftarrow (\mathcal{A} \,\|\, \mathsf{Ext}_\mathcal{A})(x; r)$ as a shorthand for "$y \leftarrow \mathcal{A}(x; r)$, $y' \leftarrow \mathsf{Ext}_\mathcal{A}(x; r)$". For an algorithm $\mathcal{A}$, let $\mathbf{Im}(\mathcal{A})$ be the image of $\mathcal{A}$, i.e. the set of valid outputs of $\mathcal{A}$, let $\mathsf{RND}(\mathcal{A})$ denote the random tape of $\mathcal{A}$. By $y \leftarrow \mathcal{A}(x; r)$ we denote the fact that $\mathcal{A}$, given an input x and a randomizer r, outputs y.

We use the definitions of NIZK arguments from [Gro16]. Let $\mathcal{R}$ be a relation generator, such that $\mathcal{R}(1^\lambda)$ 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. 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 $\mathcal{L}_{\mathbf{R}} = \{\vec{x} : \exists \vec{w}, (\vec{x}, \vec{w}) \in \mathbf{R}\}$ be an NP-language. A *NIZK argument system* Ψ for $\mathcal{R}$ 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 $(\mathbf{R}, \mathsf{z}_{\mathbf{R}}) \in \mathbf{Im}(\mathcal{R}(1^\lambda))$, 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 1 (Perfect Completeness). *A non-interactive argument Ψ is* perfectly complete for $\mathcal{R}$*, if for all λ, all $(\mathbf{R}, \mathsf{z}_{\mathbf{R}}) \in \mathbf{Im}(\mathcal{R}(1^\lambda))$, and $(\vec{x}, \vec{w}) \in \mathbf{R}$,*

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

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

Definition 2 (Computationally Knowledge-Soundness [Gro16]). *Let* $\mathsf{RND}(\mathcal{A})$ *denote the random tape of $\mathcal{A}$. A non-interactive argument Ψ is computationally (adaptively)* knowledge-sound for $\mathcal{R}$*, if for every non-uniform PPT $\mathcal{A}$, there exists a non-uniform PPT extractor $\mathsf{Ext}_{\mathcal{A}}$, s.t. for all λ, the following probability is $\mathsf{negl}(\lambda)$,*

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

Here, $\mathsf{z}_{\mathbf{R}}$ can be seen as a common auxiliary input to $\mathcal{A}$ that is generated by using a benign [BCPR14] relation generator. 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 3 (Weak Simulation Extractability [KZM+15]). *Let* $\mathsf{RND}(\mathcal{A})$ *denote the random tape of $\mathcal{A}$. A non-interactive argument Ψ is* (non-black-box)

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

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

Here, Q is the set of statements queried by adversary to the simulation oracle O. 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 4 (Simulation Extractability [GM17]). *Let* $\mathsf{RND}(\mathcal{A})$ *denote the random tape of* $\mathcal{A}$. *A non-interactive argument* Ψ *is* (non-black-box strong) simulation-extractable for $\mathcal{R}$, *if for any non-uniform PPT* $\mathcal{A}$, *there exists a non-uniform PPT extractor* $\mathsf{Ext}_{\mathcal{A}}$ *s.t. for all* λ, *the following probability is* $\mathsf{negl}(\lambda)$,

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

Here, Q is the set of simulated statement-proof pairs generated by adversary's queries to the simulation oracle O.

Note that both variants of *simulation extractability* implies *knowledge soundness* (given in Def. 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 5 (Zero-Knowledge (ZK) [Gro16]). *A non-interactive argument* Ψ *is* computationally ZK for $\mathcal{R}$, *if for all* λ, *all* $(\mathbf{R}, \mathsf{z}_{\mathbf{R}}) \in \mathbf{Im}(\mathcal{R}(1^{\lambda}))$, *and for all non-uniform PPT* $\mathcal{A}$, $\varepsilon_0 \approx_c \varepsilon_1$, *where*

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

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 $\mathcal{R}$ *if one requires that* $\varepsilon_0 = \varepsilon_1$.

Intuitively, a non-interactive argument is zero-knowledge 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 generic group model.

3 A Simulation Extractable zk-SNARK without RO

In this section, we present our first (strong) SE variation of Groth16 based on Bowe and Gabizon's construction [BG18].

3.1 Scheme Definition

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 propose the construction of our first variation of Groth16, that similarly works with quadratic arithmetic programs. In this construction, we change the Proof of Knowledge (PoK) of the DLOG of $[\delta']_2$ w.r.t. $[\delta]_2$ to another PoK in the GGM without using random oracles with a variation of Boneh-Boyen signatures, where we just use the collision resistance property of the hash function. In Fig. 1, 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 construction in the following.

Avoiding Random Oracle. Our proof uses the collision resistance property of the hash function and the generic group model. Very roughly, 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 we have the knowledge of the DLOG of δ' respect to δ $(k - m)$, which is straightforward, or either we have re-used δ'_j and m_j from some jth query. The last case is discarded when we reach 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 $x, \alpha, \beta, \delta, \gamma \leftarrow \mathbb{Z}_p^*$, $H \leftarrow \mathcal{H}$, and returns crs defined as the following,

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

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 $[h(x)t(x)/\delta']_1 \leftarrow (1/\zeta)(\sum_{i=0}^{n-2} h_i \left[x^i t(x)/\delta\right]_1)$,
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 $[C]_1 \leftarrow r_b\,[A]_1 + r_a\left(\sum_{j=0}^{m} a_j\,[v_j(x)]_1 + [\beta]_1\right) + (1/\zeta)\sum_{j=l+1}^{m} a_j([(u_j(x)\beta + v_j(x)\alpha + w_j(x))/\delta]_1) + \left[h(x)t(x)/\delta'\right]_1$,
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 $\pi := ([A, C, D]_1\,, \left[B, \delta'\right]_2)$.

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 $\pi := ([A, C, D]_1\,, \left[B, \delta'\right]_2)$.

Fig. 1. The proposed strong SE variant of Groth16 for $\mathbf{R}$ along with a modification of the Boneh-Boyen signature. In the protocol, $\mathcal{H}$ is a family of collision resistant hash functions that map to $\mathbb{Z}_p^*$.

3.2 Security Proofs

We prove security of our construction (in Fig. 1) in the following theorem.

Theorem 1 (Completeness, ZK, SE). *The variation of* Groth16 *described in Fig. 1, guarantees (1) perfect completeness, 2) perfect zero-knowledge and 3) simulation-extractability in the asymmetric Generic Group Model.*

Proof. The proofs of perfect completeness and perfect zero-knowledge are similar to original Groth16, that are omitted. Particularly, in our case one needs to use the simulation algorithm described in Fig. 1.

Simulation extractability is proven by reduction (in the GGM) to the knowledge soundness of Groth16, and the reduction works in these two steps:

Step 1. Extraction of the DLOG of δ'.

Step 2. Reduction to the *knowledge soundness* of Groth16.

Proof of Step 1) Suppose $\mathcal{A}$ has made a sequence of queries $x_1, \cdots, x_v$ to $\mathsf{Sim}(\vec{\mathsf{ts}}, \cdot)$, and received answers $\{\pi_j = ([A_j]_1, [B_j]_2, [C_j]_1, [D_j]_1, [\delta_j]_2)\}_{j=1}^{v}$. Let Q' be the union of elements in the crs together with those from the replies of $\mathsf{Sim}(\vec{\mathsf{ts}}, \cdot)$; namely,

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

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 hash function for each A_j, B_j, C_j, δ_j.

We assume the adversary $\mathcal{A}$ has produced elements (A, B, C, D, δ') such that

$$A \cdot B \equiv C \cdot \delta' + \Big(\sum_{j=0}^{l} a_j (u_j(x)\beta + v_j(x)\alpha + w_j(x))\Big) + \alpha\beta$$

and, for $m := H([A]_1 \parallel [B]_2 \parallel [C]_1 \parallel [\delta']_2)$, $D(\delta' + \delta m) = t(x)(m + \gamma)$. Let Q'_1 the elements in Q' in $\mathbb{G}_1$ and Q'_2 the elements in $\mathbb{G}_2$. Since the adversary is generic it has constructed these elements as a linear combination of the elements in Q' which are in the relevant group (i.e. element of Q'_1 in $\mathbb{G}_1$ for A, C, D and of Q'_2 for B, δ') and we can extract the coefficients of this linear combination.

First, we prove that the adversary has knowledge of the discrete logarithm of δ' w.r.t. δ. From the second verification equation we know that $D = t(x)\frac{\gamma + m}{\delta' + m\delta}$. On the other hand, from adversary $\mathcal{A}$ we can recover a vector $\vec{k}_D$ with the coefficients that it has used to construct D, that is, $D = \sum_{q \in Q'_1} k_{D,q}\ q$. Equating these two expressions,

$$t(x)(m + \gamma) = (\textstyle\sum_{q \in Q'_1} k_{D,q}\ q)(\delta' + m\delta), \tag{1}$$

where $\delta' = \sum_{q \in Q_2'} k_{\delta',q}\ q$ for another vector of coefficients $\vec{k}_{\delta'}$. The terms which include γ in both sides of the equation must be the same.

On the other hand, by assumption, in the asymmetric GGM, the term δ' is constructed as a linear combination of elements in Q_2' and therefore $\delta' + \delta m$ is independent of γ. Then, keeping only the terms with γ in equation (1), we obtain the following relation:

$$t(x)\gamma = k_{D,0}\frac{\gamma t(x)}{\delta}(\delta' + m\delta) + \sum_{j=1}^{v} k_{D,j}\frac{\gamma t(x)}{\delta_j + m_j\delta}(\delta' + m\delta), \tag{2}$$

where we have set $k_{D,0} = k_{D,\frac{\gamma t(x)}{\delta}}$ and $k_{D,j} = k_{D,\frac{\gamma t(x)}{\delta_j + m_j\delta}}$ to simplify the notation.

Dividing both sides of the equation by $t(x)\gamma$ and defining $\delta_0 = \delta'$, $m_0 = 0$, we obtain the following equivalent equation:

$$1 = \left(\sum_{j=0}^{v} k_{D,j}\frac{1}{\delta_j + m_j\delta}\right)(\delta' + m\delta) = \sum_{j=0}^{v} k_{D,j}\frac{\prod_{i=0,i\neq j}^{v}(\delta_i + m_i\delta)}{\prod_{i=0}^{v}(\delta_i + m_i\delta)}(\delta' + m\delta)$$

$$\Leftrightarrow \prod_{i=0}^{v}(\delta_i + m_i\delta) = (\delta' + m\delta)\left(\sum_{j=0}^{v} k_{D,j}\prod_{i=0,i\neq j}(\delta_i + m_i\delta)\right). \tag{3}$$

From the last equation it follows that the term $\delta' + m\delta$ must divide the left side of the equation (3). Therefore, there exists some index j^* and $k \in \mathbb{Z}_p$ such that $\delta' + m\delta = k(\delta_{j^*} + m_{j^*}\delta)$. Now, dividing Eq. (3) by $(\delta_{j^*} + m_{j^*}\delta)$, we come to the following expression

$$0 = (1 - k \cdot k_{D,j^*}) \prod_{i=0,i\neq j^*}^{v}(\delta_i + m_i\delta) - \sum_{j=0,j\neq j^*}^{v} k_{D,j} \prod_{i=0,i\neq j}^{v}(\delta_i + m_i\delta) \tag{4}$$

Since all summands are linearly independent polynomials, $k = k_{D,j^*}^{-1}$, and $k_{D,j} = 0$ if $j \neq j^*$. We distinguish two cases: (1) $\delta' + \delta m = k\delta$ $(j^* = 0)$ or (2) $\delta' + \delta m = k(\delta_{j^*} + m_{j^*}\delta)$ $(j^* \neq 0)$.

In case (1), we are done, as we can extract the DLOG of δ' as $k - m$.

In case (2), from equation (1) and putting everything together, we have that:

$$\begin{aligned} t(x)(m + \gamma) &= k_{D,j^*}\frac{(\gamma + m_{j^*})}{(\delta_{j^*} + m_{j^*}\delta)}(\delta' + m\delta) \\ &= k_{D,j^*}k^{-1}(\gamma + m_{j^*})t(x) = (\gamma + m_{j^*})t(x). \end{aligned} \tag{5}$$

This implies that $m_{j^*} = m$ is a collision of H.

Proof of Step 2) We show that the elements A, B, C do not use the elements of the simulated proofs, say $V := \{[A_j]_1, [B_j]_2, [C_j]_1, [D_j]_1, [\delta_j]_2\}_{j=1}^{v}$, and then, with the knowledge of ζ such that $\delta' = \zeta\delta$, we can reduce our proof to the knowledge soundness proof of Groth16 [Gro16], since $[A]_1, [B]_2, [C\zeta]_1$ is a valid proof of Groth16.

For this, we need to argue that A, B, C cannot have been constructed from any of the elements of the queries. To prove that A, B, C are not constructed from the elements $[A_j]_1, [B_j]_2, [C_j]_1, [\delta_j]_2$, we follow the exact same reasoning as Bowe and Gabizon [BG18] in the GGM and we omit the details. Next, we prove that to construct A, C the prover cannot have used any of the D_j terms, which are the new elements in our proof.

Analogously to proof in Section 4, assume A has been generated from some D_j, so the term $\frac{t(x)(m_j+\gamma)}{\delta_j+m_j\delta}$ appears in the expression of A generated from Q'_1 with the corresponding coefficient different than 0. Observe that the verification equation contains the term $\alpha\beta$ that cannot be manipulated because it is fixed in the crs, and it should be produced by the term AB because $\beta \in Q'_2$ and β is independent of $\delta' = \zeta\delta$. In that case, the product AB would contain a term $\frac{t(x)(m_j+\gamma)}{\delta_j+m_j\delta}\beta$, but this cannot be cancelled out by any of the other terms in the equation. Indeed, this term cannot appear in $\alpha\beta$, or in the sum of public values of a_i. Thus, the only possibility is that it appears in $C\delta'$. However, since β is independent of δ', it should appear in C, but $\frac{t(x)(m_j+\gamma)}{\delta_j+m_j\delta}\beta$ cannot be computed from elements in Q'_1.

Now, assume D_j appears in C, then the term $C\delta'$ of the verification includes $\frac{t(x)(m_j+\gamma)}{\delta_j+m_j\delta}\delta'$. Neither the term $\alpha\beta$ nor the sum of public values can include it, so the only possibility is that it appears in AB. Since $\delta' \in Q'_2$, then A would contain D_j, which we ruled out previously. □

4 A More Efficient SE Variant of Groth16 in the ROM

In Fig. 2, we describe another SE variant of Groth16 that uses a new technique to shorten the proof and verifies it with a single verification equation which similar to Groth16 which requires 3 pairings. This is more efficient than our first construction, however similar to Bowe and Gabizon's construction [BG18], it uses a random oracle. But, in our scheme 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).

In this variant, the prover chooses δ' as before but then uses $\delta' + \delta m$ to create and verify the proof, where, $m := H([A]_1 \parallel [B]_2 \parallel [\delta']_2)$. 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 with gray background.

Theorem 2 (Completeness, ZK, strong SE). *The variant of* Groth16 *described in Fig. 2, 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.*

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

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

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 $[A]_1 \leftarrow \sum_{j=0}^{m} a_j\,[u_j(x)]_1 + [\alpha]_1 + r_a\,[\delta']_1$,
6. Set $r_b \leftarrow_r \mathbb{Z}_p$; Set $[B]_2 \leftarrow \sum_{j=0}^{m} a_j\,[v_j(x)]_2 + [\beta]_2 + r_b\,[\delta']_2$,
7. Sets $m = H([A]_1 \parallel [B]_2 \parallel [\delta']_2)$, 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 $m = H([A]_1 \parallel [B]_2 \parallel [\delta']_2)$ checks if

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

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 $m = H([A]_1 \parallel [B]_2 \parallel [\delta']_2)$
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. 2. A simulation-extractable variation of Groth16 for $\mathbf{R}$. $\mathcal{H}$ is a family of collision resistant hash functions that map to $\mathbb{Z}_p^*$.

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:

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

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

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

Completeness easily follows from these formulae (in fact, it is identical to the completeness of Groth16). Similarly, perfect zero-knowledge can be argued in a standard way. Simulation extractability is proven by reduction (in the GGM) to the knowledge soundness of Groth16. The proof works in two steps, first the extraction of the DLOG of δ' and then the reduction to the knowledge soundness of Groth16.

Proof of Step 1) Assume that an adversary $\mathcal{A}$ has made a sequence of queries $\vec{x}_1, \ldots, \vec{x}_v$ to $\mathsf{Sim}(\vec{\mathsf{ts}}, \cdot)$, and received answers $\{\pi_j = ([A_j, C_j]_1, [B_j, \delta_j]_2)\}_{j=1}^{v}$. Let Q' be the union of elements in the crs together with those from the replies of $\mathsf{Sim}(\vec{\mathsf{ts}}, \cdot)$; namely,

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

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. Now, assume that the adversary $\mathcal{A}$ has produced elements $\pi = (A, C, B, \delta')$ 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 $m = H(A||B||\delta')$.

First, we prove such an adversary must to have knowledge of the $\mathrm{DLOG}_{[\delta]_2}[\delta']_2$. In the GGM from the adversary $\mathcal{A}$ we can recover the coefficients that it has used

to construct δ', i.e. $\delta' = k_0 + k_\beta\beta + k_\delta\delta + \sum_{i=0}^{n-1} k_{x,i}x^i + \sum_{j=1}^{v}(k_{B,j}B_j + k_{\delta,j}\delta_j)$. We argue in the following that δ' must be a polynomial just in δ, i.e. $\delta' = k_\delta\delta$.

Assuming H to be a random oracle, the probability of event $k_\delta = m$ is at most $1/p$. On the other hand, excluding this event, $\delta' + m\delta$ is a polynomial that has a non-zero coefficient accompanying δ. Thus, to find C satisfying the verification equation, the (generic) prover must then divide by a polynomial that has a non-trivial δ coefficient. It follows that $\delta' + m\delta$ has to be a polynomial that is a multiple of $\delta_j + m_j\delta$, or of δ. Indeed, it cannot be a linear combination of these polynomials. For example, say $\delta' + m\delta = z_1(\delta_1 + m_1\delta) + z_2(\delta_2 + m_2\delta_2)$: the adversary only knows C_1 that is a $(\delta_1 + m_1\delta)$-root, and similarly for C_2, but no linear combination of these terms (with coefficients in the field) will result in a $\delta' + \delta m$ root. Therefore, there are two possibilities:

- $\delta' + m\delta = z_j(\delta_j + m_j\delta)$ for some $j = 1, \ldots, v$. This implies that $m = z_j m_j$. Since z_j, m_j are chosen independently of m because of the RO, the probability of this event is at most $1/m$.
- Else, if $\delta' + m\delta = z\delta$ for some $z \in \mathbb{Z}_p$, this means that $\delta' = k_\delta\delta$, for $k_\delta = z - m$.

Proof of Step 2) Now, we show the elements A, B, C do no use elements of the simulated proofs and we can reduce our proof to the knowledge soundness proof of Groth16 with the elements $([A]_1, [B]_2, [\zeta C]_1)$. We have to prove that the elements A, B, C are not constructed from the elements $[A_j]_1, [B_j]_2$, $[C_j]_1$, $[\delta_j]_2$. Following the exact reasoning as Bowe and Gabizon [BG18] in the GGM, since our new elements are C and C_j, we should prove that to construct A and C, the prover cannot have used any of the C_j. However, this is what we have already proven above when we discard the case $k_{\delta,j} \neq 0$. □

5 Empirical Analysis

We evaluate the efficiency of our presented simulation extractable variants of Groth's zk-SNARK using a prototype implementation of both protocols in Arkworks [6] 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] in the same library [7].

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

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

[7] Source codes of our implementations are publicly available on: `https://github.com/Baghery/ABPR21`

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, GGM: Generic Group Model, RO: Random Oracle, CRH: Collision Resistant Hash. Among the strong SE ones, we have highlighted the most efficient verification.

EC	SNARK	SE	Model	PCPT (ns)	Proof (B)	Ver. 1 Proof	Ver. 10^2 Proofs	Ver. 10^3 Proofs
BLS12-381	[Gro16]	WSE	GGM	$\approx$ 5026	127.5	$\approx$ 1.90 ms	$\approx$ 0.190 s	$\approx$ 1.90 s
	[GM17]	SSE	GGM	$\approx$ 11042	127.5	$\approx$ 3.32 ms	$\approx$ 0.332 s	$\approx$ 3.32 s
	[BG18]	SSE	GGM, RO	$\approx$ 5052	223.1	$\approx$ 3.52 ms	$\approx$ 0.352 s	$\approx$ 3.52 s
	Sec. 3	SSE	GGM, CRH	$\approx$ 5042	223.1	$\approx$ 4.85 ms	$\approx$ 0.360 s	$\approx$ 3.50 s
	Sec. 4	SSE	GGM, RO	$\approx$ 5041	191.2	$\approx$ 2.39 ms	$\approx$ 0.194 s	$\approx$ 1.91 s
MNT4-298	[Gro16]	WSE	GGM	$\approx$ 4830	149.0	$\approx$ 2.67 ms	$\approx$ 0.267 s	$\approx$ 2.67 s
	[GM17]	SSE	GGM	$\approx$ 10025	149.0	$\approx$ 3.80 ms	$\approx$ 0.380 s	$\approx$ 3.80 s
	[BG18]	SSE	GGM, RO	$\approx$ 4879	260.7	$\approx$ 4.32 ms	$\approx$ 0.432 s	$\approx$ 4.32 s
	Sec. 3	SSE	GGM, CRH	$\approx$ 4881	260.7	$\approx$ 4.45 ms	$\approx$ 0.311 s	$\approx$ 3.05 s
	Sec. 4	SSE	GGM, RO	$\approx$ 4875	223.5	$\approx$ 3.33 ms	$\approx$ 0.271 s	$\approx$ 2.68 s
MNT6-298	[Gro16]	WSE	GGM	$\approx$ 5794	186.2	$\approx$ 4.94 ms	$\approx$ 0.494 s	$\approx$ 4.94 s
	[GM17]	SSE	GGM	$\approx$ 11427	186.2	$\approx$ 7.07 ms	$\approx$ 0.707 s	$\approx$ 7.07 s
	[BG18]	SSE	GGM, RO	$\approx$ 5831	335.2	$\approx$ 8.07 ms	$\approx$ 0.807 s	$\approx$ 8.07 s
	Sec. 3	SSE	GGM, CRH	$\approx$ 5824	335.2	$\approx$ 8.34 ms	$\approx$ 0.582 s	$\approx$ 5.72 s
	Sec. 4	SSE	GGM, RO	$\approx$ 5810	298.0	$\approx$ 6.11 ms	$\approx$ 0.501 s	$\approx$ 4.97 s
MNT4-753	[Gro16]	WSE	GGM	$\approx$ 30247	376.5	$\approx$ 29.1 ms	$\approx$ 2.91 s	$\approx$ 29.1 s
	[GM17]	SSE	GGM	$\approx$ 83120	376.5	$\approx$ 41.6 ms	$\approx$ 4.16 s	$\approx$ 41.6 s
	[BG18]	SSE	GGM, RO	$\approx$ 30863	658.8	$\approx$ 47.3 ms	$\approx$ 4.73 s	$\approx$ 47.3 s
	Sec. 3	SSE	GGM, CRH	$\approx$ 30887	658.8	$\approx$ 45.5 ms	$\approx$ 3.41 s	$\approx$ 33.8 s
	Sec. 4	SSE	GGM, RO	$\approx$ 30760	564.7	$\approx$ 33.9 ms	$\approx$ 2.94 s	$\approx$ 29.2 s
MNT6-753	[Gro16]	WSE	GGM	$\approx$ 33298	470.6	$\approx$ 53.6 ms	$\approx$ 5.36 s	$\approx$ 53.6 s
	[GM17]	SSE	GGM	$\approx$ 83121	470.6	$\approx$ 76.9 ms	$\approx$ 7.69 s	$\approx$ 76.9 s
	[BG18]	SSE	GGM, RO	$\approx$ 33358	847.1	$\approx$ 88.5 ms	$\approx$ 8.85 s	$\approx$ 88.5 s
	Sec. 3	SSE	GGM, CRH	$\approx$ 33359	847.1	$\approx$ 85.4 ms	$\approx$ 6.33 s	$\approx$ 63.1 s
	Sec. 4	SSE	GGM, RO	$\approx$ 33345	753.0	$\approx$ 64.4 ms	$\approx$ 5.42 s	$\approx$ 53.8 s

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 4 and 3 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 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 4 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 4 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.

6 Conclusion

Over the last few years, various SE zk-SNARKs have been proposed that achieve (strong) simulation extractability [GM17, BG18, Lip19, AB19], 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 two new variations. Our first construction 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 second 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 both of our constructions are among the most efficient SE zk-SNARKs. Particularly, in large-scale applications, the CRS, the prover, and the verifier of our second 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.

Acknowledgements. We thank Alonso González for his helpful discussions. Karim Baghery has been supported in part by ERC Advanced Grant ERC-2015-AdG-IMPaCT, by the Defense Advanced Research Projects Agency (DARPA) under contract No. HR001120C0085, and by CyberSecurity Research Flanders with reference number VR20192203. Carla Ràfols and Zaira Pindado were partially supported by Project RTI2018-102112-B-I00 (AEI/FEDER, UE).

References

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. 3, 4, 5, 18, 19

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. 5

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. 3, 6

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. 5

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. 4, 8

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. 2, 18, 19

BCPR14. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014. 7

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. 16

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. 5

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. 2

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`. 2, 3, 4, 5, 6, 9, 13, 16, 17, 18, 19

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`. 14

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. 5

BKSV20. Karim Baghery, Markulf Kohlweiss, Janno Siim, and Mikhail Volkhov. Another look at extraction and randomization of groth's zk-SNARK. Cryptology ePrint Archive, Report 2020/811, 2020. `https://eprint.iacr.org/2020/811`. 3, 6

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`. 2

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. 1, 6

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. 2, 5, 18

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. 5

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. 2

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. 2, 3, 4, 5, 8, 16, 17, 18

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. 3, 6, 18

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. 2, 4, 5, 6, 7, 8, 12, 16, 17

KDS20. Mojtaba Khalili, Mohammad Dakhilalian, and Willy Susilo. Efficient chameleon hash functions in the enhanced collision resistant model. *Inf. Sci.*, 510:155–164, 2020. 18

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. 2

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. 2

KZM+15. Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. How to use SNARKs in universally composable protocols. Cryptology ePrint Archive, Report 2015/1093, 2015. `https://eprint.iacr.org/2015/1093`. 3, 7

Lip19. Helger Lipmaa. Simulation-extractable SNARKs revisited. Cryptology ePrint Archive, Report 2019/612, 2019. `http://eprint.iacr.org/2019/612`. 3, 4, 18

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. 16

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. 2