Simulation Extractability in Groth's zk-SNARK

Shahla Atapoor and Karim Baghery

University of Tartu, Estonia

Abstract. A Simulation Extractable (SE) zk-SNARK enables a prover to prove that she knows a witness for an instance in a way that the proof: (1) is succinct and can be verified very efficiently; (2) does not leak information about the witness; (3) is simulation-extractable -an adversary cannot come out with a new valid proof unless it knows a witness, even if it has already seen arbitrary number of simulated proofs. Non-malleable succinct proofs and very efficient verification make SE zk-SNARKs an elegant tool in various privacy-preserving applications such as cryptocurrencies, smart contracts and etc. In Eurocrypt 2016, Groth proposed the most efficient pairing-based zk-SNARK in the CRS model, but its proof is vulnerable to the malleability attacks. In this paper, we show that one can efficiently achieve simulation extractability in Groth's zk-SNARK by some changes in the underlying language using an OR construction. Analysis and implementations show that in practical cases overload has minimal effects on the efficiency of original scheme which currently is the most efficient zk-SNARK. In new construction, proof size is extended with one element from $\mathbb{G}_1$, one element from $\mathbb{G}_2$, plus a bit string that totally is less than 256 bytes for 128-bit security. Its verification is dominated with 4 pairings which is the most efficient verification among current SE zk-SNARKs.

Keywords: ZK proofs, SNARKs, simulation extractability, CRS model

1 Introduction

Non-Interactive Zero-Knowledge (NIZK) proofs are one of the central design tools in cryptographically secure systems, allowing one to verify the veracity of statements without leaking extra information. Technically speaking, a NIZK allows a prover to prove that, for a public statement x she knows a witness w which hold in a relation $\mathbf{R}$, $(\mathsf{x},\mathsf{w}) \in \mathbf{R}$, without leaking any information about her witness w. In the Common Reference String (CRS) model [BFM88], a NIZK is a three-party protocol that works as the following. First, there exists a trusted party K (a.k.a. CRS generator) who takes security parameter λ as an input and generates CRS elements $\mathbf{crs} := (\mathbf{crs}_{\mathsf{P}}, \mathbf{crs}_{\mathsf{V}})$ which later will be used by prover and verifier for proof generation and proof verification, respectively. Then the prover P gets $\mathbf{crs}_{\mathsf{P}}$, the statement x and her witness w and generates a proof π, attesting that for the statement x, I know a witness w s.t. $(\mathsf{x},\mathsf{w}) \in \mathbf{R}$. Finally, a verifier V takes $\mathbf{crs}_{\mathsf{V}}$, the statement x and the proof π and returns either accept (if proof is valid) or reject (if verification failed). If V does not need any secret

information to verify the proof π, the proof is called *publicly verifiable* that can be verified by many public verifiers (e.g. by nodes of a distributed network).

Generally, a NIZK argument satisfies three properties known as *completeness*, *soundness* and *zero-knowledge*. *Completeness* guarantees that an honest P always convinces an honest V. The *soundness* ensures that a malicious P cannot convince the honest V except with negligible probability. *Zero-knowledge property* assures that the proof generated by P does not leak any information about the witness w. Formal definitions will be given later in Sec. 2.1.

During last few years, a very efficient family of NIZK proof systems are developed which are known as zero-knowledge Succinct Non-interactive Argument of Knowledge (zk-SNARK) [Gro10,Lip12,PHGR13,BCTV13,Gro16,GM17]. A zk-SNARK generates a *succinct* proof that allows a computationally weak verifier to efficiently verify the proof. Differently from a standard NIZK, an SNARK guarantees *knowledge-soundness* that is a stronger notion in comparison with standard soundness. Knowledge-soundness (more precisely non-black-box knowledge soundness) guarantees that if an adversarial prover manages to come out with an acceptable proof, there exists an efficient extractor which given source code and random coins of the adversary can efficiently extract the witness. Knowledge-soundness of zk-SNARKs is non-black-box and achieved under knowledge assumptions [Dam92][1]. Impossibility result of Gentry and Wichs [GW11] also confirms that extraction in zk-SNARKs should be based on non-falsifiable assumptions (e.g. knowledge assumptions). By the date, the most efficient zk-SNARK is proposed by Groth [Gro16] in Eurocrypt 2016, that is constructed for Quadratic Arithmetic Programs (QAPs) and works in a bilinear group. The proof in Groth's zk-SNARK consists of 2 elements in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$, and V needs to check one equation that is dominated with 3 pairings.

In practice, however knowledge-soundness is an amplified notion in comparison with standard soundness but still a knowledge-sound proof is vulnerable to the man-in-the-middle attacks [2]. In other words, knowledge soundness only guarantees that a successful prover knows the witness, and it does not guarantee non-malleability of proofs. Due to this fact, zk-SNARKs that only guarantee knowledge-soundness cannot be deployed in many of practical applications straightforwardly [BCG+14,KMS+16,JKS16,Bag19]. For instance, privacy-preserving crypocurrencies such as Zerocash that uses zk-SNARKs [BCTV13,Gro16] as a subroutine, takes extra steps to prevent malleability attacks in the SNARK proofs for *pour* transactions [BCG+14]. Similarly, privacy-preserving smart contract systems [KMS+16,JKS16] show that knowledge-soundness of zk-SNARKs is not enough for their systems. *Simulation-*

[1] In non-black-box extraction, extractor $\mathsf{ext}_{\mathcal{A}}$ needs to get full access to the source code and random coins of adversary $\mathcal{A}$ to be able to extract the witness. But in black-box extraction, one can extract the witnesses straightforwardly from the proof using CRS trapdoors [Bag19].

[2] For instance, in the verification equations that have paring structure such as $a \bullet b = c$, where a and b are proof elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ with prime orders, one can see that such verification equation will be satisfied also for new proof elements such as $a' = a^r$ and $b' = b^{1/r}$, for arbitrary $r \leftarrow \mathbb{Z}_p$.

knowledge soundness which also is known as *simulation extractability*, is an amplified version of knowledge-soundness that is proposed to achieve extractability and non-malleable proofs. A Simulation-Extractable (SE) zk-SNARK guarantees that the proof is succinct, zero-knowledge and *simulation-extractable*. Simulation extractability implies that an adversary cannot generate a new proof unless he knows a witness, even if he has seen arbitrary number of simulated proofs.

In Crypto 2017, Groth and Maller [GM17] proposed the first SE zk-SNARK in the CRS model that allows to generate non-malleable proofs (referred as GM zk-SNARK). They also proved that a SE zk-SNARK requires at least two verification equations. Their scheme is constructed in the bilinear groups for Square Arithmetic Programs (SAPs) and achieves the lower bound in the number of verification equations. To verify a proof, V needs to check two equations that are dominated with 5 pairings [GM17]. To guarantee non-malleability in proofs, their scheme removes one of the bilinear group generators from the CRS, which might create some different challenges in some practical cases (e.g. in CRS generation by multi-party computation protocols [ABL+19], or in achieving subversion security [ABLZ17]). Above all, GM zk-SNARK is constructed for SAPs that require twice number of multiplication (MUL) gates, as $ab = ((a+b)^2 - (a-b)^2)/4$. Implementations also approves that for a particular arithmetic circuit, Groth's zk-SNARK [Gro16] considerably has better efficiency than GM zk-SNARK [GM17], but Gorth's scheme does not achieve simulation extractability, which makes its proofs vulnerable to the malleability attacks. For a Rank-1 Constraint System (R1CS) instance, efficiency metrics of both schemes are compared in Tab. 1.

Table 1: Performance of zk-SNARKs proposed by Groth and Maller [GM17], Groth [Gro16] and this work for arithmetic circuit satisfiability with an R1CS instance with 10^6 constraints and 10^6 variables, where 10 are input variables. SE: Simulation Extractable, KS: Knowledge Soundness, BS: λ-Bit String.

SNARK	CRS size, run time	Proof size, P run time	Verification, V time	Security
[GM17]	376 MB, 103 sec	$2\mathbb{G}_1 + 1\mathbb{G}_2$, 120 sec	5 pairings, 2.3 ms	SE
[Gro16]	196 MB, 75 sec	$2\mathbb{G}_1 + 1\mathbb{G}_2$, 83 sec	3 pairings, 1.4 ms	KS
this work	205 MB, 80.5 sec	$3\mathbb{G}_1 + 2\mathbb{G}_2 + 1$BS, 90 sec	4 pairings, 2.0 ms	SE

Problem statement. By reminding that currently Groth's zk-SNARK [Gro16] has the best efficiency but only achieves knowledge-soundness, and the fact that GM zk-SNARK [GM17] ensures simulation extractability but with less efficiency and only one group generator in the CRS, a research question can be raised as if we can achieve simulation extractability in Groth's scheme efficiently? Such that, new scheme will 1) work for QAPs 2) have both generators of bilinear groups in the CRS 3) have a comparable or even better efficiency than GM zk-SNARK.

Our Contribution. In this paper, we address the questions discussed above and propose a variation of Groth's zk-SNARK that can achieve simulation extractability with minimal efficiency loss in practical cases. To this end, we use the known OR technique and define a new language $\mathbf{L}'$ based on the language $\mathbf{L}$ in Groth's zk-SNARK that is inspired by the works of De Santis et al. [DDO+01] and Kosba et al. [KZM+15].

Defining new language based on original language results some changes in algorithms of the original scheme. Evaluations show that in practical cases, new changes have minimal affects on the efficiency of original scheme which currently is the state-of-the-art. Strictly speaking, the verification of new scheme has two equations as the optimal case, and only adds 1 pairing to the verification of Groth's scheme. As a result, totally verification of new scheme is dominated with 4 pairings which is less than 5 pairings in GM SE zk-SNARK [GM17]. Empirical analysis show that, for the considered instance in Tab. 1, verification of new scheme takes 2.0 milliseconds. In the proposed variation, the proof size will be extended by one element from $\mathbb{G}_1$, one element from $\mathbb{G}_2$ plus a 256-bit string, that totally will be 3 elements from $\mathbb{G}_1$, 2 elements from $\mathbb{G}_2$ and one 256-bit string, which for 128-bit security still it is less than 256 bytes. The prover should give a proof for a new circuit that has around 50×10^3 gates more than before, where in practical scenarios the overload is very small. I.e. Zerocash uses zk-SNARKs to give a proof for a circuit with approximately 2×10^6 MUL gates[3]. In comparison with P running times in Tab. 1, prover of new scheme requires 90 sec to generate a proof; particularly with smaller CRS in comparison with GM [GM17] scheme (with 205 MB, instead of 376 MB). Efficiency of the proposed variation is summarized in Tab. 1.

Discussion and Related Works. Among different NIZK arguments, zk-SNARKs are the most practically-interested ones; because of their succinct proofs and very efficient verifications. But as majority of them guarantee knowledge-soundness by default, that is vulnerable to the man-in-the-middle attacks, so they cannot be deployed directly in practical systems. Actually, in constructing large cryptographic systems, this issue can make some challenges for non-expert users. To address this, recently constructing efficient SE zk-SNARKs, that by default can guarantee non-malleability of proofs, has gotten more attention [GM17,BG18,KLO19,Lip19]. In [BG18], Bowe and Ariel also proposed a variation of Groth's scheme that achieves simulation extractability but in the Random Oracle (RO) model. In their variation, the proof consists of 3 elements from $\mathbb{G}_1$ and 2 elements from $\mathbb{G}_2$, and verification is dominated with 5 pairings. A good point about their case is that they keep the language as original one, and add some computations to the proof generation and verification with relying on a random oracle that returns group elements[4]. Implementing such random oracle might cause some challenges in practice. Since Groth's zk-SNARK is constructed and proven in the CRS model, so we aim to achieve simulation extractability in the same model using more practical cryptographic primitives.

The rest of paper is organized as follows; Sec. 2 introduces notations and preliminaries. A simulation-extractable version of Groth's zk-SNARK is presented in Sec. 3. In Sec. 4, we discuss about instantiation and efficiency of the proposed construction. Finally we conclude the paper in Sec. 5.

[3] Their initial circuit had $\approx 4 \times 10^6$ gates, but recently they optimized the system and reduced the number of gates to $\approx 2 \times 10^6$, but still it is very larger than $\approx 50 \times 10^3$.

[4] Intuitively, some part of their changes play the role of a one-time secure signature scheme, but add two pairings to the verification of original scheme.

2 Preliminaries

Let PPT denote probabilistic polynomial-time, and NUPPT denote non-uniform PPT. Let $\lambda \in \mathbb{N}$ be the information-theoretic security parameter, say $\lambda = 128$. All adversaries will be stateful. For an algorithm $\mathcal{A}$, let $\mathrm{im}(\mathcal{A})$ 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 $\mathcal{A}$, and let $r \leftarrow \mathsf{RND}(\mathcal{A})$ denote sampling of a randomizer r of sufficient length for $\mathcal{A}$'s needs. By $y \leftarrow \mathcal{A}(x;r)$ we denote the fact that $\mathcal{A}$, given an input x and a randomizer r, outputs y. 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 "$y \leftarrow \mathcal{A}(x;r)$, $y' \leftarrow \mathsf{ext}_{\mathcal{A}}(x;r)$". We denote by $\mathsf{negl}(\lambda)$ an arbitrary negligible function in λ. For distributions A and B, $A \approx_c B$ means that they are computationally indistinguishable.

In pairing-based groups, we use additive notation together with the bracket notation, i.e., in group $\mathbb{G}_\mu$, $[a]_\mu = a[1]_\mu$, where $[1]_\mu$ is a fixed generator of $\mathbb{G}_\mu$. A *bilinear group generator* $\mathsf{BGgen}(1^\lambda)$ returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where p (a large prime) is the order of cyclic abelian groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Finally, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient non-degenerate bilinear pairing, s.t. $\hat{e}([a]_1, [b]_2) = [ab]_T$. Denote $[a]_1 \bullet [b]_2 = \hat{e}([a]_1, [b]_2)$. The current recommendation is to use an optimal (asymmetric) Ate pairing [HSV06] over Barreto-Naehrig curves [BN05]. In that case, at security level of $\lambda = 99$, an element of $\mathbb{G}_1/\mathbb{G}_2/\mathbb{G}_T$ can be represented in respectively 256/512/3072 bits.[5]

Next we review QAPs that defines NP-complete language specified by a quadratic equation over polynomials and have reduction from the language Circuit-SAT [GGPR13,Gro16].

Quadratic Arithmetic Programs. QAP was introduced by Gennaro *et al.* [GGPR13] as a language where for an input x and witness w, $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$ can be verified by using a parallel quadratic check. Consequently, any efficient simulation-extractable zk-SNARK for QAP results in an efficient simulation-extractable zk-SNARK for Circuit-SAT. An QAP instance $\mathcal{Q}_p$ is specified by the so defined $(\mathbb{Z}_p, m_0, \{u_j, v_j, w_j\}_{j=0}^m, \ell(X))$. This instance defines the following relation, where we assume that $A_0 = 1$:

$$\mathbf{R} = \left\{ \begin{array}{l} (\mathsf{x},\mathsf{w}) \colon \mathsf{x} = (A_1, \ldots, A_{m_0})^\top \wedge \mathsf{w} = (A_{m_0+1}, \ldots, A_m)^\top \wedge \\ \left(\sum_{j=0}^m A_j u_j(X)\right)\left(\sum_{j=0}^m A_j v_j(X)\right) \equiv \sum_{j=0}^m A_j w_j(X) \pmod{\ell(X)} \end{array} \right\}.$$

Alternatively, $(\mathsf{x},\mathsf{w}) \in \mathbf{R}$ if there exists a (degree $\leq n-2$) polynomial $h(X)$, s.t. $\left(\sum_{j=0}^m A_j u_j(X)\right)\left(\sum_{j=0}^m A_j v_j(X)\right) - \sum_{j=0}^m A_j w_j(X) = h(X)\ell(X)$, where $\ell(X) = \prod_{i=1}^{\mathsf{n}}(X - \omega^{i-1})$ is a polynomial related to Lagrange interpolation, and ω is an n-th primitive root of unity modulo p. Roughly speaking, the goal of the prover of a zk-SNARK for QAP [GGPR13] is to prove that for public statement $(A_1, \ldots, A_{m_0})$ and $A_0 = 1$, she knows the witnesses $(A_{m_0+1}, \ldots, A_m)$ and a degree $\leq n-2$ polynomial $h(X)$, such that above equation holds.

[5] The value $\lambda = 99$ takes account recent cryptanalysis of the Barreto-Naehrig curves by Kim and Barbulescu [KB16,BD17]. One can use different settings for 128-bit security. Since we use the library `libsnark` [BCTV13] that currently offers the mentioned security level, we just refer the reader to [KB16,BD17] for more discussion.

One-time Signature Schemes [Lam79]. A one-time signature (OTS) scheme is a digital signature scheme that can be used to sign one message per key pair. An OTS scheme is made up three PPT algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{SigVerify})$, for key generation, signing, and verification, respectively. A signature scheme is *complete* if an honesty generated signature by Sign always successfully passes the verifications by $\mathsf{SigVerify}$. We say that a signature scheme is *strong unforgeability under a one-time message attack* (SUF-1CMA) if all PPT adversaries have at most negligible advantage in the following experiment.
$\mathsf{EXP}_{\mathsf{SUF-1CMA}}$:

- *Setup:* The challenger C runs $\mathsf{KGen}(\lambda)$ to generate a signing-verification key pair $(\mathsf{sk}, \mathsf{pk})$ and gives pk to the adversary $\mathcal{A}$,
- *Signing Query:* $\mathcal{A}$ selects a message m from message space and gives it to challenger C. Challenger C computes $\sigma = \mathsf{Sign}(\mathsf{sk}, m)$ and sends it to $\mathcal{A}$,
- *Forgery:* $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*),

where adversary's advantage in above experiment is defined as $\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr[\mathsf{SigVerify}(\mathsf{pk}, m^*, \sigma^*) = 1 \wedge (m^*, \sigma^*) \neq (m, \sigma)]$.

2.1 Definitions

We use the definitions of NIZK arguments from [Gro16,GM17]. Let $\mathcal{R}$ be a relation generator, such that $\mathcal{R}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathbf{R} = \{(\mathsf{x}, \mathsf{w})\}$. Here, x is the statement and w is the witness. Security parameter λ can be deduced from the description of $\mathbf{R}$. The relation generator also outputs auxiliary information ξ that will be given to the honest parties and the adversary. As in [Gro16,ABLZ17], ξ is the value returned by $\mathsf{BGgen}(1^\lambda)$, so ξ is given as an input to the honest parties; if needed, one can include an additional auxiliary input to the adversary. Let $\mathbf{L_R} = \{\mathsf{x} : \exists \mathsf{w}, (\mathsf{x}, \mathsf{w}) \in \mathbf{R}\}$ 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 $(\mathbf{R}, \xi)$ where $(\mathbf{R}, \xi) \in \mathrm{im}(\mathcal{R}(1^\lambda))$, outputs $\mathbf{crs} := (\mathbf{crs}_\mathsf{P}, \mathbf{crs}_\mathsf{V})$ and stores trapdoors of $\mathbf{crs}$ as $\mathbf{ts}$. We distinguish $\mathbf{crs}_\mathsf{P}$ (needed by the prover) from $\mathbf{crs}_\mathsf{V}$ (needed by the verifier).
Prover: P is a PPT algorithm that, given $(\mathbf{R}, \xi, \mathbf{crs}_\mathsf{P}, \mathsf{x}, \mathsf{w})$, where $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$, outputs an argument π. Otherwise, it outputs $\perp$.
Verifier: V is a PPT algorithm that, given $(\mathbf{R}, \xi, \mathbf{crs}_\mathsf{V}, \mathsf{x}, \pi)$, returns either 0 (reject) or 1 (accept).
Simulator: Sim is a PPT algorithm that, given $(\mathbf{R}, \xi, \mathbf{crs}, \mathbf{ts}, \mathsf{x})$, outputs a simulated argument π.

A zk-SNARK system is required to be complete, knowledge-sound, ZK, and succinct as in the following definitions.

Definition 1 (Perfect Completeness). *A non-interactive argument Ψ is* perfectly complete for $\mathcal{R}$, *if for all λ, all $(\mathbf{R}, \xi) \in \mathrm{im}(\mathcal{R}(1^\lambda))$, and $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$,* $\Pr\left[\mathbf{crs} \leftarrow \mathsf{K}(\mathbf{R}, \xi),\ \pi \leftarrow \mathsf{P}(\mathbf{R}, \xi, \mathbf{crs}_\mathsf{P}, \mathsf{x}, \mathsf{w}) : \mathsf{V}(\mathbf{R}, \xi, \mathbf{crs}_\mathsf{V}, \mathsf{x}, \pi) = 1 \right] = 1$.

Definition 2 (Computationally Knowledge-Soundness [Gro16]). *A non-interactive argument Ψ is computationally (adaptively)* knowledge-sound for *$\mathcal{R}$, if for every NUPPT $\mathcal{A}$, there exists a NUPPT extractor $\mathsf{ext}_{\mathcal{A}}$, s.t. for all λ,*

$$\Pr\left[\begin{matrix}\mathsf{crs} \leftarrow \mathsf{K}(\mathbf{R},\xi), r \leftarrow \mathsf{RND}(\mathcal{A}), ((\mathsf{x},\pi)\,\|\,\mathsf{w}) \leftarrow (\mathcal{A}\,\|\,\mathsf{ext}_{\mathcal{A}})(\mathbf{R},\xi,\mathsf{crs};r) : \\ (\mathsf{x},\mathsf{w}) \notin \mathbf{R} \wedge \mathsf{V}(\mathbf{R},\xi,\mathsf{crs}_{\mathsf{V}},\mathsf{x},\pi) = 1\end{matrix}\right] = \mathsf{negl}(\lambda) \ .$$

Here, ξ can be seen as a common auxiliary input to $\mathcal{A}$ and $\mathsf{ext}_{\mathcal{A}}$ that is generated by using a benign [BCPR14] relation generator.

Definition 3 (Computationally Zero-Knowledge (ZK) [Gro16]). *A non-interactive argument Ψ is* computationally ZK for *$\mathcal{R}$, if for all λ, all $(\mathbf{R},\xi) \in \mathrm{im}(\mathcal{R}(1^{\lambda}))$, and for all NUPPT $\mathcal{A}$, $\varepsilon_0 \approx_c \varepsilon_1$, where*

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

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

Definition 4 (Succinctness [GM17]). *A non-interactive argument Ψ is* succinct *if the proof size is polynomial in λ and the verifier's computation time is polynomial in security parameter λ and the size of instance x.*

In the rest, we recall the definition of (non-black-box) simulation extractability that we aim to achieve in a variation of Groth's zk-SNARK.

Definition 5 ((Non-Black-Box) Simulation Extractability [GM17]). *A non-interactive argument Ψ is* (non-black-box) simulation-extractable for *$\mathcal{R}$, if for any NUPPT $\mathcal{A}$, there exists a NUPPT extractor $\mathsf{ext}_{\mathcal{A}}$ s.t. for all λ,*

$$\Pr\left[\begin{matrix}\mathsf{crs} \leftarrow \mathsf{K}(\mathbf{R},\xi), r \leftarrow \mathsf{RND}(\mathcal{A}), ((\mathsf{x},\pi)\,\|\,\mathsf{w}) \leftarrow (\mathcal{A}^{\mathsf{O}(.)}\,\|\,\mathsf{ext}_{\mathcal{A}})(\mathbf{R},\xi,\mathsf{crs};r) : \\ (\mathsf{x},\pi) \notin Q \wedge (\mathsf{x},\mathsf{w}) \notin \mathbf{R} \wedge \mathsf{V}(\mathbf{R},\xi,\mathsf{crs}_{\mathsf{V}},\mathsf{x},\pi) = 1\end{matrix}\right] = \mathsf{negl}(\lambda) \ .$$

Here, Q is the set of (x,π)-pairs generated by the adversary's queries to $\mathsf{O}(.)$. Note that *(non-black-box) simulation extractability* implies *knowledge-soundness.*

3 A Variation of Groth's zk-SNARK

As briefly discussed in the introduction, Groth's zk-SNARK [Gro16] guarantees knowledge-soundness (defined in Def. 2) which is weaker than simulation extractability. Technically speaking, knowledge-sound proofs are not secure against man-in-the-middle attacks. In this section, we present a variation of Groth's zk-SNARK which can achieve (non-black-box) simulation extractability, defined in Def. 5, that can guarantee non-malleability of the proofs.

3.1 New Construction

In construction of new variation, we define a new language $\mathbf{L}'$, using an OR technique [DDO+01,KZM+15], which combines original language $\mathbf{L}$ in Groth's zk-SNARK with a commitment scheme which commits to a secret randomness as a key for a pseudorandom function. Let $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{SigVerify})$ be a one-time signature scheme and $(\mathsf{Com}, \mathsf{ComVerify})$ be a perfectly binding commitment scheme.

Given the language $\mathbf{L}$ with the corresponding NP relation $\mathbf{R_L}$, we define a new language $\mathbf{L}'$ such that $((\mathsf{x}, \mu, \mathsf{pk_{Sign}}, \rho), (\mathsf{w}, s, r)) \in \mathbf{R}_{\mathbf{L}'}$ iff:

$$\big((\mathsf{x}, \mathsf{w}) \in \mathbf{R_L} \vee (\mu = f_s(\mathsf{pk_{Sign}}) \wedge \rho = \mathsf{Com}(s, r))\big),$$

where $\{f_s : \{0,1\}^* \to \{0,1\}^\lambda\}_{s\in\{0,1\}^\lambda}$ is a pseudo-random function family. The intuition for a pseudo-random function $f_s(\cdot)$ is that without the knowledge of the key s, $f_s(\cdot)$ behaves like a true random function. However, given s, one can compute $f_s(\cdot)$ easily. In new language $\mathbf{L}'$, for a statement-witness pair to be valid, either a witness for $\mathbf{R_L}$ is provided (by honest prover) or an opening to ρ together with the value of $\mu = f_s(\mathsf{pk_{Sign}})$ is provided (by simulator), where s is the open value of ρ (in CRS). One may note that in order for a statement to pass the verification without a valid witness, the prover must generate $f_s(\mathsf{pk_{Sign}})$ without the knowledge of s (thus breaking the pseudo-random function f_s). By considering new language $\mathbf{L}'$, zk-SNARK of Groth for the relation $\mathbf{R}$ constructed from PPT algorithms $(\mathsf{K}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ can be lifted to a simulation-extractable zk-SNARK Ψ' with PPT algorithms $(\mathsf{K}', \mathsf{P}', \mathsf{V}', \mathsf{Sim}')$ as described in Fig. 1. To simplify the description, we assume Com takes exactly λ random bits as randomness and that the witness for original language $\mathbf{L}$ is exactly λ bits; it is straight forward to adapt the proof when they are of different lengths [KZM+15]. Note that in the simulation-extractable zk-SNARK Ψ', the algorithms of original scheme will be executed for a new arithmetic circuit which encodes new language $\mathbf{L}'$ and has slightly larger number of gates. Namely, CRS generation algorithm of Groth's zk-SNARK will be executed with a new QAP instance that has larger parameters; $(\mathbf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathbf{R}_{\mathbf{L}'}, \xi)$. Similarly prover of new variation will execute prover of Groth's zk-SNARK with a new arithmetic circuit that has larger number of gates; namely $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathbf{L}'}, \xi, \mathbf{crs}, (\mathsf{x}, z_0, \mathsf{pk_{Sign}}, \rho), (\mathsf{w}, z_1, z_2))$, where z_1 and z_2 play the role of witnesses s and r for prover.

3.2 Security Proofs

In the rest we present security proofs of the proposed scheme.

Theorem 1 (Completeness). *The variation of Groth's zk-SNARK described in Sec. 3.1, guarantees completeness.*

Proof. In new scheme internal computations of P and V are the same as original one, except few extra efficient computations. Precisely, P needs to do the computation for a new instance that has slightly larger size (e.g. $n = n_{old} + n_{new}$, where n_{new} is the number of MUL gates added to the old circuit) and sign the proof

CRS generator $\mathsf{K}'(\mathbf{R_L}, \xi)$**:** Sample $(\mathbf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathbf{R_{L'}}, \xi)$; $s, r \leftarrow \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s, r)$; and output $(\mathbf{crs}' \,\|\, \mathsf{ts}') := ((\mathbf{crs}, \rho) \,\|\, (\mathsf{ts}, (s, r)))$; where ts' is new simulation trapdoor.

Prover $\mathsf{P}'(\mathbf{R_L}, \xi, \mathbf{crs}', \mathsf{x}, \mathsf{w})$**:** Parse $\mathbf{crs}' := (\mathbf{crs}, \rho)$; abort if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R_L}$; generate $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$; sample $z_0, z_1, z_2 \leftarrow \{0,1\}^\lambda$; generate $\pi \leftarrow \mathsf{P}(\mathbf{R_{L'}}, \xi, \mathbf{crs}, (\mathsf{x}, z_0, \mathsf{pk}_{\mathsf{Sign}}, \rho), (\mathsf{w}, z_1, z_2))$ using the prover of Groth's scheme; sign $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (\mathsf{x}, z_0, \pi))$; and return $\pi' := (z_0, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$.

Verifier $\mathsf{V}'(\mathbf{R_L}, \xi, \mathbf{crs}', \mathsf{x}, \pi')$**:** Parse $\mathbf{crs}' := (\mathbf{crs}, \rho)$ and $\pi' := (z_0, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$; abort if $\mathsf{SigVerify}(\mathsf{pk}_{\mathsf{Sign}}, (\mathsf{x}, z_0, \pi), \sigma) = 0$; call the verifier of Groth's scheme $\mathsf{V}(\mathbf{R_{L'}}, \xi, \mathbf{crs}, (\mathsf{x}, z_0, \mathsf{pk}_{\mathsf{Sign}}, \rho), \pi)$ and abort if it outputs 0.

Simulator $\mathsf{Sim}'(\mathbf{R_L}, \xi, \mathbf{crs}', \mathsf{ts}', \mathsf{x})$**:** Parse $\mathbf{crs}' := (\mathbf{crs}, \rho)$ and $\mathsf{ts}' := (\mathsf{ts}, (s, r))$; generate $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$; set $\mu = f_s(\mathsf{pk}_{\mathsf{Sign}})$; generate $\pi \leftarrow \mathsf{Sim}(\mathbf{R_{L'}}, \xi, \mathbf{crs}, (\mathsf{x}, \mu, \mathsf{pk}_{\mathsf{Sign}}, \rho), (\mathsf{ts} \,\|\, (s, r)))$; sign $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (\mathsf{x}, \mu, \pi))$; and output $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$.

Fig. 1: A variation of Groth's zk-SNARK.

and statement with a one-time signature scheme. So following the completeness of original scheme, and the fact that the deployed signature scheme is complete, which means $\mathsf{SigVerify}(\mathsf{pk}_{\mathsf{Sign}}, m, \mathsf{Sign}(m, \mathsf{sk}_{\mathsf{Sign}})) = 1$, one can conclude that the modified construction satisfies *completeness.* □

Theorem 2 (Zero-Knowledge). *Assume that Groth's zk-SNARK satisfies computational zero-knowledge, that the pseudo-random function family is secure, that the commitment scheme is perfectly binding and computational hiding, and that the one-time signature scheme is unforgeable, then the presented SNARK described in Sec. 3.1, guarantees computational zero-knowledge.*

Proof. We write a series of hybrid experiments which start from an experiment with the simulator and ends with an experiment that uses the real prover. We show that all experiments are two-by-two indistinguishable. Changes between successive experiments are shown with highlights. Recall that Groth's zk-SNARK guarantees perfect zero-knowledge and the simulator of the modified scheme is expressed in Fig. 1. Now consider the following experiments,

$\underline{\mathsf{EXP}_1}$(simulator):

- *Setup:* Sample $(\mathbf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathbf{R_{L'}}, \xi)$; $s, r \leftarrow \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s, r)$; and output $(\mathbf{crs}' \,\|\, \mathsf{ts}') := ((\mathbf{crs}, \rho) \,\|\, (\mathsf{ts}, (s, r)))$; where ts' is simulation trapdoor.
- *Define function* $\mathsf{O}(\mathsf{x}, \mathsf{w})$: Abort if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R_L}$; $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$; set $\mu = f_s(\mathsf{pk}_{\mathsf{Sign}})$; generate $\pi \leftarrow \mathsf{Sim}(\mathbf{R_{L'}}, \xi, \mathbf{crs}, (\mathsf{x}, \mu, \mathsf{pk}_{\mathsf{Sign}}, \rho), \mathsf{ts}')$; sign $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (\mathsf{x}, \mu, \pi))$; return $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$.
- $b \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x},\mathsf{w})}(\mathbf{crs}')$

$\underline{\mathsf{EXP}_2}$(separate secret key of pseudo random function):

- *Setup:* Sample $(\mathbf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathbf{R_{L'}}, \xi)$; $s', s, r \leftarrow \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s', r)$; and output $(\mathbf{crs}' \,\|\, \mathsf{ts}') := ((\mathbf{crs}, \rho) \,\|\, (\mathsf{ts}, (s, s', r)))$; where ts' are new trapdoors.
- *Define function* $\mathsf{O}(\mathsf{x}, \mathsf{w})$: Abort if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R_L}$; generate $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$; set $\mu = f_s(\mathsf{pk}_{\mathsf{Sign}})$; generate $\pi \leftarrow$

$\mathsf{Sim}(\mathbf{R}_{\mathbf{L}'}, \xi, \mathbf{crs}, (\mathsf{x}, \mu, \mathsf{pk}_{\mathsf{Sign}}, \rho), (\mathsf{ts} \,\|\, (s, r)))$; sign $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (\mathsf{x}, \mu, \pi))$; return $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$.

- $b \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x},\mathsf{w})}(\mathbf{crs}')$

Lemma 1. *If the underlying commitment scheme is computationally hiding, then for two experiments* EXP_2 *and* EXP_1 *we have* $\Pr[\mathsf{EXP}_2] \approx_c \Pr[\mathsf{EXP}_1]$.

Proof. Computationally hiding property of a commitment scheme implies that $\mathsf{Com}(m_1, r)$ is computationally indistinguishable from $\mathsf{Com}(m_2, r)$. So this property straightforwardly results the lemma. □

$\underline{\mathsf{EXP}_3}$(replace pseudo random function):

- *Setup:* Sample $(\mathbf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathbf{R}_{\mathbf{L}'}, \xi)$; s', $\not{s}$, $r \leftarrow \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s', r)$; and output $(\mathbf{crs}' \,\|\, \mathsf{ts}') := ((\mathbf{crs}, \rho) \,\|\, (\mathsf{ts}, (\not{s}, s', r)))$; where ts' is simulation trapdoor and barred characters such as $\not{s}$ are removed characters.
- *Define function* $\mathsf{O}(\mathsf{x}, \mathsf{w})$: Abort if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathbf{L}}$; $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$; set $\mu \leftarrow \{0,1\}^\lambda$; generate $\pi \leftarrow \mathsf{Sim}(\mathbf{R}_{\mathbf{L}'}, \xi, \mathbf{crs}, (\mathsf{x}, \mu, \mathsf{pk}_{\mathsf{Sign}}, \rho), (\mathsf{ts} \,\|\, (s', r)))$; sign $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (\mathsf{x}, \mu, \pi))$; return $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$.
- $b \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x},\mathsf{w})}(\mathbf{crs}')$

Lemma 2. *If the pseudo random function* $f_s(\cdot)$ *is secure and the underlying one-time signature scheme is unforgeable, we have* $\Pr[\mathsf{EXP}_3] \approx_c \Pr[\mathsf{EXP}_2]$.

Proof. By considering that the signature scheme is secure, we note that the generated $\mathsf{pk}_{\mathsf{Sign}}$ is unique with overwhelming probability. Additionally, we can replace the pseudo random function $f_s(\cdot)$ with a true random function that will result EXP_4. By considering unique values of $\mathsf{pk}_{\mathsf{Sign}}$ and indistinguishability of output of $f_s(\cdot)$ and truly random function, one can conclude the claim. □

$\underline{\mathsf{EXP}_4}$(prover):

- *Setup:* Sample $(\mathbf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathbf{R}_{\mathbf{L}'}, \xi)$; $s', r \leftarrow \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s', r)$; and output $(\mathbf{crs}' \,\|\, \mathsf{ts}') := ((\mathbf{crs}, \rho) \,\|\, (\mathsf{ts}, (s', r)))$; where ts' is simulation trapdoor.
- *Define function* $\mathsf{O}(\mathsf{x}, \mathsf{w})$: Abort if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathbf{L}}$; $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$; set $\mu \leftarrow \{0,1\}^\lambda$ (μ plays the role of z_0 in Fig. 1); sample $z_1, z_2 \leftarrow \{0,1\}^\lambda$; generate $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathbf{L}'}, \xi, \mathbf{crs}, (\mathsf{x}, \mu, \mathsf{pk}_{\mathsf{Sign}}, \rho), (\mathsf{w}, z_1, z_2))$; sign $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (\mathsf{x}, \mu, \pi))$; return $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$.
- $b \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x},\mathsf{w})}(\mathbf{crs}')$

Lemma 3. *If Groth's zk-SNARK guarantees zero-knowledge, then we have* $\Pr[\mathsf{EXP}_4] \approx_c \Pr[\mathsf{EXP}_3]$.

Proof. The last experiment exactly models the real prover of construction in Fig. 1, and as already Groth's scheme guarantees zero-knowledge, so one can conclude that the real proof in experiment EXP_4 is indistinguishable from the simulated proof in EXP_3. Intuitively this is because all new elements added to the new construction are chosen randomly and independently. □

This concludes the main theorem. □

Theorem 3 ((Non-Black-Box) Simulation Extractability). *Assume that Groth's zk-SNARK satisfies knowledge soundness and computational zero-knowledge, that the pseudo-random function family is secure, that the commitment scheme is perfectly binding and computational hiding, and that the one-time signature scheme is unforgeable, then the presented SNARK described in Sec. 3.1, guarantees (non-black-box) simulation extractability.*

Proof. Similarly, we write a sequence of hybrid experiences and finally show that the success probability in the last game is negligible. Recall that Groth's scheme is proven to achieve knowledge-soundness (defined in Def. 2). Now consider the following game,

EXP_1(main experiment):

- *Setup:* Sample $(\mathbf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathbf{R}_{\mathbf{L}'}, \xi)$; $s, r \leftarrow \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s, r)$; and output $(\mathbf{crs}' \,\|\, \mathsf{ts}') := ((\mathbf{crs}, \rho) \,\|\, (\mathsf{ts}, (s, r)))$; where ts' is new CRS trapdoor.
- *Define function* $\mathsf{O}(\mathsf{x})$: $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$; set $\mu = f_s(\mathsf{pk}_{\mathsf{Sign}})$; generate $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathbf{L}'}, \xi, \mathbf{crs}, (\mathsf{x}, \mu, \mathsf{pk}_{\mathsf{Sign}}, \rho), (\mathsf{w}, (s, r)))$; sign $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (\mathsf{x}, \mu, \pi))$; return $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$.
- $(\mathsf{x}, \pi') \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x})}(\mathbf{crs}')$.
- Parse $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$; $\mathsf{w} \leftarrow \mathsf{ext}_{\mathcal{A}}(\mathbf{crs}', \mathsf{x}, \pi, \xi)$.
- Return 1 iff $((\mathsf{x}, \pi') \notin Q) \wedge \ (\mathsf{V}'(\mathbf{R}_{\mathbf{L}}, \xi, \mathbf{crs}', \mathsf{x}, \pi') = 1) \wedge ((\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathbf{L}})$; where Q shows the set of statement-proof pairs generated by $\mathsf{O}(\mathsf{x})$.

EXP_2(relaxing the return checking):

- *Setup:* Sample $(\mathbf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathbf{R}_{\mathbf{L}'}, \xi)$; $s, r \leftarrow \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s, r)$; and output $(\mathbf{crs}' \,\|\, \mathsf{ts}') := ((\mathbf{crs}, \rho) \,\|\, (\mathsf{ts}, (s, r)))$; where ts' is new CRS trapdoor.
- *Define function* $\mathsf{O}(\mathsf{x})$: $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$; set $\mu = f_s(\mathsf{pk}_{\mathsf{Sign}})$; generate $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathbf{L}'}, \xi, \mathbf{crs}, (\mathsf{x}, \mu, \mathsf{pk}_{\mathsf{Sign}}, \rho), (\mathsf{w}, (s, r)))$; sign $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (\mathsf{x}, \mu, \pi))$; return $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$.
- $(\mathsf{x}, \pi') \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x})}(\mathbf{crs}')$.
- Parse $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$; $\mathsf{w} \leftarrow \mathsf{ext}_{\mathcal{A}}(\mathbf{crs}', \mathsf{x}, \pi, \xi)$.
- Return 1 iff $((\mathsf{x}, \pi') \notin Q) \wedge \ (\mathsf{V}'(\mathbf{R}_{\mathbf{L}}, \xi, \mathbf{crs}', \mathsf{x}, \pi') = 1) \wedge (\mathsf{pk}_{\mathsf{Sign}} \notin \mathcal{PK}) \wedge (\mu = f_s(\mathsf{pk}_{\mathsf{Sign}}))$; where Q is the set of statement-proof pairs and $\mathcal{PK}$ is the set of signature verification keys, both generated by $\mathsf{O}(\mathsf{x})$.

Lemma 4. *If the one-time signature scheme is unforgeable, and Groth's scheme guarantees knowledge-soundness, then* $\Pr[\mathsf{EXP}_2] \leq \Pr[\mathsf{EXP}_1] + \mathsf{negl}(\lambda)$.

Proof. We note that if $(\mathsf{x}, \pi') \notin Q$ and "$\mathsf{pk}_{\mathsf{Sign}}$ from (x, π'), has been generated by $\mathsf{O}(\cdot)$", then the (x, μ, π) is a valid message/signature pairs. Therefore by unforgeability of the signature scheme, we know that $(\mathsf{x}, \pi) \notin Q$ and "$\mathsf{pk}_{\mathsf{Sign}}$ has been generated by $\mathsf{O}(\cdot)$" happens with negligible probability, which allows us to focus on $\mathsf{pk}_{\mathsf{Sign}} \notin \mathcal{PK}$.

Now, due to knowledge-soundness of the original scheme (there is an extractor $\mathsf{ext}_{\mathcal{A}}$ where can extract witness from $\mathcal{A}$), if some witness is valid for $\mathbf{L}'$ and $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathbf{L}}$, so we conclude it must be the case that there exists some s', such that ρ is valid commitment of s' and $\mu = f_{s'}(\mathsf{pk}_{\mathsf{Sign}})$, which by perfectly binding property of the commitment scheme, it implies $\mu = f_s(\mathsf{pk}_{\mathsf{Sign}})$. □

$\underline{\mathsf{EXP}_3}$(simulator):

- *Setup:* Sample $(\mathbf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathbf{R}_{\mathbf{L}'}, \xi)$; $s, r \leftarrow \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s, r)$; and output $(\mathbf{crs}' \,\|\, \mathsf{ts}') := ((\mathbf{crs}, \rho) \,\|\, (\mathsf{ts}, (s, r)))$; where ts' is new CRS trapdoor.
- *Define function* $\mathsf{O}(\mathsf{x})$: $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$; set $\mu = f_s(\mathsf{pk}_{\mathsf{Sign}})$; generate $\pi \leftarrow \mathsf{Sim}(\mathbf{R}_{\mathbf{L}'}, \xi, \mathbf{crs}, (\mathsf{x}, \mu, \mathsf{pk}_{\mathsf{Sign}}, \rho), (\mathsf{ts} \,\|\, (s, r)))$; sign $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (\mathsf{x}, \mu, \pi))$; return $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$.
- $(\mathsf{x}, \pi') \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x})}(\mathbf{crs}')$.
- Parse $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$; $\mathsf{w} \leftarrow \mathsf{ext}_{\mathcal{A}}(\mathbf{crs}', \mathsf{x}, \pi, \xi)$.
- Return 1 iff $((\mathsf{x}, \pi') \notin Q) \wedge \ (\mathsf{V}'(\mathbf{R}_{\mathbf{L}}, \xi, \mathbf{crs}', \mathsf{x}, \pi') = 1) \wedge (\mathsf{pk}_{\mathsf{Sign}} \notin \mathcal{PK}) \wedge (\mu = f_s(\mathsf{pk}_{\mathsf{Sign}}))$; where Q is the set of statement-proof pairs and $\mathcal{PK}$ is the set of signature verification keys, both generated by $\mathsf{O}(\mathsf{x})$.

Lemma 5. *If Groth's SNARK guarantees zero-knowledge, then for two experiments* EXP_3 *and* EXP_2*, we have* $\Pr[\mathsf{EXP}_3] \leq \Pr[\mathsf{EXP}_2] + \mathsf{negl}(\lambda)$.

Proof. As the original scheme ensures (perfect) zero-knowledge, so it implies no polynomial time adversary can distinguish a proof generated by the simulator from a proof that is generated by the prover. So, as we are running in polynomial time, thus two experiments are indistinguishable. □

$\underline{\mathsf{EXP}_4}$(separating secret key of pseudo random function):

- *Setup:* Sample $(\mathbf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathbf{R}_{\mathbf{L}'}, \xi)$; $s', s, r \leftarrow \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s', r)$; and output $(\mathbf{crs}' \,\|\, \mathsf{ts}') := ((\mathbf{crs}, \rho) \,\|\, (\mathsf{ts}, (s, s', r)))$; where ts' is new CRS trapdoor.
- *Define function* $\mathsf{O}(\mathsf{x})$: $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$; set $\mu = f_s(\mathsf{pk}_{\mathsf{Sign}})$; generate $\pi \leftarrow \mathsf{Sim}(\mathbf{R}_{\mathbf{L}'}, \xi, \mathbf{crs}, (\mathsf{x}, \mu, \mathsf{pk}_{\mathsf{Sign}}, \rho), (\mathsf{ts} \,\|\, (s, r)))$; sign $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (\mathsf{x}, \mu, \pi))$; return $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$.
- $(\mathsf{x}, \pi') \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x})}(\mathbf{crs}')$.
- Parse $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$; $\mathsf{w} \leftarrow \mathsf{ext}_{\mathcal{A}}(\mathbf{crs}', \mathsf{x}, \pi, \xi)$.
- Return 1 iff $((\mathsf{x}, \pi') \notin Q) \wedge \ (\mathsf{V}'(\mathbf{R}_{\mathbf{L}}, \xi, \mathbf{crs}', \mathsf{x}, \pi') = 1) \wedge (\mathsf{pk}_{\mathsf{Sign}} \notin \mathcal{PK}) \wedge (\mu = f_s(\mathsf{pk}_{\mathsf{Sign}}))$; where Q is the set of statement-proof pairs and $\mathcal{PK}$ is the set of signature verification keys, both generated by $\mathsf{O}(\mathsf{x})$.

Lemma 6. *If the commitment scheme used in the CRS generation is computationally hiding, then* $\Pr[\mathsf{EXP}_4] \leq \Pr[\mathsf{EXP}_3] + \mathsf{negl}(\lambda)$.

Proof. Computationally hiding of a commitment scheme implies that $\mathsf{Com}(m_1, r)$ and $\mathsf{Com}(m_2, r)$ are computationally indistinguishable, as in this lemma. □

$\underline{\mathsf{EXP}_5}$(replace pseudo random function $f_s(\cdot)$ with true random function $F(\cdot)$):

- *Setup:* Sample $(\mathbf{crs}||\mathsf{ts}) \leftarrow \mathsf{K}(\mathbf{R}_{\mathbf{L}'}, \xi)$; s', $\not{s}$, $r \leftarrow \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s', r)$; and output $(\mathbf{crs}'||\mathsf{ts}') := ((\mathbf{crs}, \rho)||(\mathsf{ts}, (\not{s}, s', r)))$; where ts' is simulation trapdoor.
- *Define function* $\mathsf{O}(\mathsf{x})$: $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$; set $\mu = F(\mathsf{pk}_{\mathsf{Sign}})$; generate $\pi \leftarrow \mathsf{Sim}(\mathbf{R}_{\mathbf{L}'}, \xi, \mathbf{crs}, (\mathsf{x}, \mu, \mathsf{pk}_{\mathsf{Sign}}, \rho), (\mathsf{ts}||(s, r)))$; sign $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (\mathsf{x}, \mu, \pi))$; return $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$.
- $(\mathsf{x}, \pi') \leftarrow \mathcal{A}^{\mathsf{O}(\mathsf{x})}(\mathbf{crs}')$.

- Parse $\pi' := (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, \sigma)$; $\mathsf{w} \leftarrow \mathsf{ext}_{\mathcal{A}}(\mathbf{crs}', \mathsf{x}, \pi, \xi)$.
- Return 1 iff $((\mathsf{x}, \pi') \notin Q) \wedge \ (\mathsf{V}'(\mathbf{R}_{\mathbf{L}}, \xi, \mathbf{crs}', \mathsf{x}, \pi') = 1) \wedge (\mathsf{pk}_{\mathsf{Sign}} \notin \mathcal{PK}) \wedge (\mu = F(\mathsf{pk}_{\mathsf{Sign}}))$; where Q is the set of statement-proof pairs and $\mathcal{PK}$ is the set of signature verification keys, both generated by $\mathsf{O}(\mathsf{x})$.

Lemma 7. *If the truly random function is secure, then* $\Pr[\mathsf{EXP}_4] \leq \Pr[\mathsf{EXP}_5]$.

Proof. By assuming function $F(\cdot)$ is secure, we can conclude no polynomial time adversary can distinguish an output of the true random function $F(\cdot)$ from an output of the pseudo random function $f_s(\cdot)$. Indeed, experiment EXP_5 can be converted to an adversary for the game of a *true random function.* □

Claim. For experiment EXP_5, we have $\Pr[\mathsf{EXP}_5] \leq 2^{-\lambda}$.

Proof. From verification we know $\mathsf{pk}_{\mathsf{Sign}} \notin \mathcal{PK}$, therefore $F(\mathsf{pk}_{\mathsf{Sign}})$ has not been queried already. Thus, we will see $F(\mathsf{pk}_{\mathsf{Sign}})$ as a newly generated random string independent from μ, which implies adversary only can guess. □

This completes proof of the main theorem. □

4 Instantiation and Efficiency Evaluation

We observed in Sec. 3.1 that defining the new language $\mathbf{L}'$ led to some changes in the algorithms of original scheme. In this section, we discuss how efficient can be such changes (described in Fig. 1). We first discuss how the used primitives can be instantiated and then evaluate efficiency of the whole protocol.

Recall that in result of new changes, one needed a pseudo random function, a commitment scheme and a one-time secure signature scheme. In similar practical cases, both pseudo random function and commitment scheme are instantiated using an efficient SHA-256 circuit that has around $\approx 25 \times 10^3$ MUL gates for one block (512-bit input) [BCG+14,KMS+16][6].

The next primitive that we need to instantiate is the digital signature that should be one-time signature scheme and unforgeable. As Groth's zk-SNARK is paring-based and is constructed with bilinear groups, so we instantiate the signature scheme with Boneh and Boyen's signature [BB08] where works in bilinear groups and has very efficient verification; it requires only one pairing and one multi-exponentiation. Their scheme is proven to guarantee unforgeability under chosen message attack and consequently unforgeability under *one-time* chosen message attack. The key generation, signing and verification of Boneh and Boyen's signature scheme [BB08] for message m is summarized below.

- **Key Generation**, $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{KGen}(1^\lambda)$: Given system parameters for a prime-order bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, randomly selects $\mathsf{sk} \leftarrow \mathbb{Z}_p^*$, and computes $\mathsf{sk} \cdot [1]_1$ and returns $(\mathsf{pk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) := ([\mathsf{sk}]_1, \mathsf{sk})$.
- **Signing**, $[\sigma]_2 \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, m)$: Given system parameters, a secret key $\mathsf{sk}_{\mathsf{Sign}}$, and a message m, computes $[\sigma]_2 = [1/(m + \mathsf{sk})]_2$ and returns $[\sigma]_2$ as the signature.

[6] It has 25.538 gates in the `xjsnark` library, `https://github.com/akosba/xjsnark`.

– **Verification**, $\{1,0\} \leftarrow \mathsf{SigVerify}(\mathsf{pk}_{\mathsf{Sign}}, [\sigma]_2)$: Given a public key $\mathsf{pk}_{\mathsf{Sign}}$, a message m, and a signature $[\sigma]_2$, verifies if $[m+\mathsf{sk}]_1 \bullet [1/(m+\mathsf{sk})]_2 = [1]_T$; if so, it returns 1; otherwise it returns 0,

where $\bullet$ denotes the paring operation. In our case, we use the same bilinear group as in the original zk-SNARK and m would be the hash (e.g. with SHA224 or SHA256) of concatenations of the proof elements with the statement, i.e. $m := H(\mathsf{x}\|z_0\|\pi)$[7]. As it can be seen, the scheme generates a single-element signature from $\mathbb{G}_2$, its public key is an element from $\mathbb{G}_1$, and above all its verification only requires one paring. Note that $[1]_T$ can be preprocessed and shared in the CRS.

So by considering the above instantiation, new proof $\pi' = (\mu, \pi, \mathsf{pk}_{\mathsf{Sign}}, [\sigma]_2)$ will be as $\pi' = (\mu, \pi, [\mathsf{sk}]_1, [1/(m+\mathsf{sk})]_2)$ where from original scheme $\pi = ([a]_1, [b]_2, [c]_1)$, and μ is an output of the pseudo random function $f_s(\cdot)$, which is instantiated with SHA-256 hash function [KMS+16]. As a result, the proof in new scheme will be 3 elements from $\mathbb{G}_1$, 2 elements from $\mathbb{G}_2$ and one 256-bit string. Consequently, new changes add only one paring to the verification of original scheme. To the best of our knowledge, this is the first simulation-extractable zk-SNARK in the CRS model which its verification is dominated with 4 pairings.

Next, we empirically analyse efficiency of the proposed scheme from different perspectives. Tab. 2 summarizes asymptotic and empirical performance of new scheme and two zk-SNARKs proposed by Groth's [Gro16] and GM [GM17]. Implementations of Groth's [Gro16] and GM [GM17] zk-SNARKs are available in `libsnark` library [BCTV13] [8], so similarly implementation of new scheme is done in the same library.

In Tab. 2, all implementation results are reported for the same R1CS instance. In the rest of analysis, we evaluate efficiency of new scheme for different R1CS instances and compare with *knowledge sound* scheme of Groth [Gro16] and simulation-extractable scheme of GM [GM17]. Strictly speaking, in top plots of Fig. 2, we compare CRS size and CRS generation time of new scheme with the mentioned zk-SNARKs for R1CS instances with 100 input variables and different number of MUL gates, from range 25×10^3 till 2×10^6 gates. Similarly, in bottom left of Fig. 2, we plot prover's running time in three zk-SNARKs for various R1CS instances with 100 input variables and different number of MUL gates. Finally, the plot in bottom right of Fig. 2, compares the verification time of new SE zk-SNARK for various R1CS instances with 10^5 constraints and various number of input variables.

From the comparisons in Tab. 2 and empirical analysis in Fig. 2, one can observe that in order to give non-malleable proofs for an arithmetic circuit satisfiability in circuits with larger than 50×10^3 MUL gates, the proposed SE

[7] As shown in [BB08], by taking hash of input message the signature scheme can be used to sign arbitrary messages in $\{0,1\}^*$.To do so, a collision resistant hash function $H : \{0,1\}^* \rightarrow \{0,\ldots,2^b\}$ such that $2^b < p$ is sufficient [BB08]. By considering recent analysis on Barreto-Naehrig curves by Kim and Barbulescu [BD17], one can use different settings for various security levels which would need to use different hash functions for signing arbitrary messages in [BB08] signature scheme.

[8] Available on `https://github.com/scipr-lab/libsnark`

Table 2: An efficiency comparison of new scheme with Groth's [Gro16] and GM [GM17] zk-SNARKs for arithmetic circuit satisfiability with m_0 elements instance, m wires, n MUL gates. In [GM17], n MUL gates translate to $2n$ squaring gates. Implementations (Implem.) are done on a Laptop with 2.50 GHz Intel Core i5-7200U CPU, with 16GB RAM, in single-threaded mode, for an R1CS instance with $n = 10^6$ constraints and $m = 10^6$ variables, of which $m_0 = 10$ are input variables. $\mathbb{G}_1$ and $\mathbb{G}_2$: group elements, E: exponentiations, P: pairings. In the new scheme, the statement contains $(\mathsf{x}, \mu, \mathsf{pk}_{\mathsf{Sign}}, \rho)$ which has 3 new elements $(\mu, \mathsf{pk}_{\mathsf{Sign}}, \rho)$, so $m'_0 = m_0 + 3$. All asymptotic analysis of new scheme are done based on our particular instantiation of commitment and pseudo random function. So, as new changes add $\approx 50 \times 10^3$ MUL gates to n and m, so $n' = n + 50.000$ and $m' = m + 50.000$.

SNARK	CRS size & gen. time	Proof size	Comp. & time of P	V	Sec.
[GM17]	$m + 4n + 5$ $\mathbb{G}_1$	2 $\mathbb{G}_1$	$m + 4n - m_0$ E_1	m_0 E_1	
&	$2n + 3$ $\mathbb{G}_2$	1 $\mathbb{G}_2$	$2n$ E_2	5 P	SE
Implem.	376 MB, 103 sec	127 bytes	120 sec	2.3 ms	
[Gro16]	$m + 2n - m_0$ $\mathbb{G}_1$	2 $\mathbb{G}_1$	$m + 3n - m_0 + 3$ E_1	m_0 E_1	
&	$n + 3$ $\mathbb{G}_2$	1 $\mathbb{G}_2$	$n + 1$ E_2	3 P	KS
Implem.	196 MB, 75 sec	127 bytes	83 sec	1.4 ms	
Sec.3.1	$m' + 2n' - m'_0 + 5$ $\mathbb{G}_1$	3 $\mathbb{G}_1$ + 2 $\mathbb{G}_2$	$m' + 3n' - m'_0 + 4$ E_1	$m'_0 + 1$ E_1	
&	$n' + 3$ $\mathbb{G}_2$	1 bit string	$n' + 2$ E_2	4 P	SE
Implem.	205 MB, 80.5 sec	254 bytes	90.1 sec	2.0 ms	

zk-SNARK can outperform GM SE zk-SNARK considerably. Note that, however new construction has larger proof size than GM zk-SNARK, 254 bytes in comparison with 127 bytes, but still its verification requires smaller number of pairings, and in the worst cases it is as efficient as verification of GM SE zk-SNARK [GM17].

5 Conclusion

We proposed a variation of the state-of-the-art zk-SNARK [Gro16] which can achieve simulation extractability; consequently allows to generate non-malleable succinct proofs. We used an efficient OR construction to define a new language $\mathbf{L}'$ from the language $\mathbf{L}$ in original scheme, that led to some changes in the algorithms of original scheme. Analysis and implementation results showed that in practical scenarios, new changes have minimal effect on the efficiency of original scheme which currently is the most efficient paring-based zk-SNARK in the CRS model [Gro16]. Precisely speaking, evaluations showed that for arithmetic circuits with larger than $\approx 50 \times 10^3$ MUL gates, the proposed SE zk-SNARK outperforms GM SE zk-SNARK [GM17]. We emphasize that in current real-life systems that use zk-SNARKs, their underlying arithmetic circuits have much more larger number of gates than 50×10^3. For instance, in Zerocash cryptocurrency [BCG+14] their current circuit for *pour* transactions has 2×10^6 MUL gates; or similarly in Hawk smart contract system [KMS+16], their circuit for

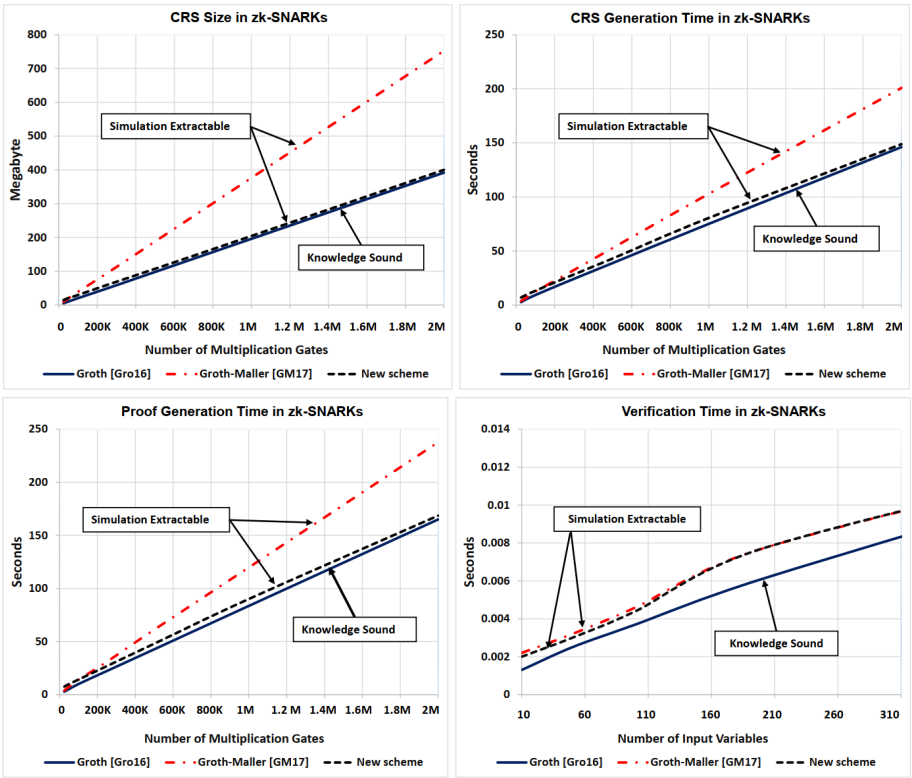

Fig. 2: A comparison of various efficiency metrics in zk-SNARKs of Groth [Gro16], Groth-Maller [GM17] and the proposed variation. Except the plot of verification time in zk-SNARKs (bottom right), all plots are drawn for R1CS instances with 10 input variables and various number of constraints (multiplication gates). In the plot of verification time (bottom right), we draw running time of verifiers in all three zk-SNARKs for R1CS instances with 10^5 constraints and different number of inputs.

finalize operation in an auction with 50 bidders has around 4×10^6 MUL gates. In comparison with GM SE zk-SNARK [GM17], however proof of new scheme is extended slightly, but still its total size is less than 256 bytes for 128-bit security; and importantly its verification is dominated with smaller number of pairings, that allows very efficient verification.

At the end, we highlight that the proposed scheme can be used to construct an efficient *succinct signature of knowledge* scheme, which would be more efficient than the one that is proposed by Groth and Maller [GM17].

Acknowledgement. The authors were supported by the European Union's Horizon 2020 research and innovation programme under grant agreements No 780477 (project PRIViLEDGE), and by the Estonian Research Council grant (PRG49).

References

ABL+19. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michal Zajac. UC-secure CRS generation for SNARKs. In *AFRICACRYPT 19*, LNCS, pages 99–117. Springer, Heidelberg, 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.

Bag19. Karim Baghery. On the efficiency of privacy-preserving smart contract systems. In *AFRICACRYPT 19*, LNCS, pages 118–136. Springer, Heidelberg, 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.

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.

BCTV13. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive arguments for a von neumann architecture. Cryptology ePrint Archive, Report 2013/879, 2013. `http://eprint.iacr.org/2013/879`.

BD17. Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. Cryptology ePrint Archive, Report 2017/334, 2017. `http://eprint.iacr.org/2017/334`.

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. *IACR Cryptology ePrint Archive*, 2018:187, 2018.

BN05. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. Cryptology ePrint Archive, Report 2005/133, 2005. `http://eprint.iacr.org/2005/133`.

Dam92. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992.

DDO+01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001.

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.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010.

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.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

HSV06. F. Hess, N.P. Smart, and F. Vercauteren. The eta pairing revisited. Cryptology ePrint Archive, Report 2006/110, 2006. `http://eprint.iacr.org/2006/110`.

JKS16. Ari Juels, Ahmed E. Kosba, and Elaine Shi. The ring of Gyges: Investigating the future of criminal smart contracts. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 283–295. ACM Press, October 2016.

KB16. Taechan Kim and Razvan Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 543–571. Springer, Heidelberg, August 2016.

KLO19. Jihye Kim, Jiwon Lee, and Hyunok Oh. Qap-based simulation-extractable snark with a single verification. Cryptology ePrint Archive, Report 2019/586, 2019. `https://eprint.iacr.org/2019/586`.

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 E. Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, T.-H. Hubert Chan, Charalampos Papamanthou, Rafael Pass, Abhi Shelat, and Elaine Shi. CØCØ: A Framework for Building Composable Zero-Knowledge Proofs. Technical Report 2015/1093, November 10, 2015. `http://eprint.iacr.org/2015/1093`, last accessed version from 9 Apr 2017.

Lam79. Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012.

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

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.