SoK: Lifting Transformations for Simulation Extractable Subversion and Updatable SNARKs*

Behzad Abdolmaleki[1], Sebastian Ramacher[2], and Daniel Slamanig[2]

[1] University of Tartu, Estonia
behzad.abdolmaleki@ut.ee
[2] AIT Austrian Institute of Technology, Austria
{sebastian.ramacher, daniel.slamanig}@ait.ac.at

Abstract. Zero-knowledge proofs and in particular succinct non-interactive zero-knowledge proofs (so called zk-SNARKs) are getting increasingly used in real-world applications, with cryptocurrencies being the prime example. Simulation extractability (SE) is a strong security notion of zk-SNARKs which informally ensures non-malleability of proofs, which is considered highly important in practical applications. Another problematic issue for the practical use of zk-SNARKs is the requirement of a fully trusted setup, as especially for large-scale decentralized applications finding a trusted party that runs the setup is practically impossible. Quite recently, the study of approaches to relax or even remove the trust in the setup procedure, and in particular subversion as well as updatable zk-SNARKs (with latter being the most promising approach), has been initiated and received considerable attention since then. Unfortunately, so far SE-SNARKs with aforementioned properties are only constructed in an ad-hoc manner and no generic techniques are available.
In this SoK paper we present the state-of-the-art in generic techniques to obtain SE subversion and updatable SNARKs. In particular, we present a revisited version of the lifting technique due to Kosba et al. (called C∅C∅). This revisited version called OC∅C∅ explores the design space of many recently proposed SNARK- and STARK-friendly symmetric-key primitives. While C∅C∅ and OC∅C∅ are compatible with subversion SNARKs, they are not compatible with updatable SNARKs. Then, we present another lifting transformation called Lamassu, which is build upon key-homomorphic signatures as well as so called updatable signatures. Lamassu preserves the subversion and in particular updatable properties of the underlying zk-SNARK. Finally, we present an comprehensive comparison of these lifting transformations with ad-hoc techniques as well as a discussion of many aspects regarding the instantiation of the techniques.

1 Introduction

Zero-knowledge (ZK) proofs were introduced by Goldwasser, Micali and Rackoff [GMR85] more than 3 decades ago. They represent a cryptographic protocol between two parties called the prover and the verifier, with the goal that

* This paper is based on and contains a significant part of the results in [ARS20].

the prover convinces the verifier of the membership of a word x in any language in NP without revealing any information about the witness w certifying language membership of word x. Besides this zero-knowledge property, such a system needs to provide soundness, i.e., it must be infeasible for the prover to provide proofs for words outside of the language. While ZK proofs in general may require many rounds of interaction, a variant highly relevant to practical applications are non-interactive zero-knowledge (NIZK) proofs [BFM88]. They require only a single round, i.e., the prover outputs a proof which can then be verified by anybody. (NI)ZK plays a central role in the theory of cryptography and meanwhile increasingly finds its way into practice.[3,4,5] Important applications are electronic voting [SK95, DGS03, Gro10b], anonymous credentials [Cha86, CL01, CL03, CL04, BCC+09, CKL+16, FHS19], and group signatures [Cv91, ACJT00, BBS04, DP06, BCC+16, DS18], including widely deployed schemes such as direct anonymous attestation (DAA) [BCC04, CCD+17] used in the Trusted Platform Module (TPM) or Intel's Enhanced Privacy ID (EPID) [BL09], as well as many other applications that require balancing privacy and integrity (cf. [FPS+18]). They are also a core building block of verifiable computation [GGP10, GGPR13, PHGR13, BCG+18] and in the increasingly popular domain of privacy-respecting cryptocurrencies [BCG+14, CGL+17], smart contracts [KMS+16] and self-sovereign identity systems [MGGM18]. Latter arguably represent the most popular real-world applications of zero-knowledge nowadays, where it sees deployments in systems such as Zcash, Ethereum or sovrin.

A challenging issue, particularly important in context of blockchains, is that users need to download and verify the state of the chain. Thus, small proof sizes and fast verification are important criteria for the practical use of ZK proofs. These desired features are provided by zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs)[6], which are NIZK proofs in which proofs as well as the computation of the verifier are succinct and ideally represent a small constant amount of space and computation respectively. Additionally, they satisfy a stronger notion of soundness called knowledge soundness, which guarantees that if an adversarial prover comes up with a proof that is accepted by the verifier, then there exists an efficient extractor which given some secret information can extract the witness. A combined effort of a large number of recent research works [Gro10a, Lip12, GGPR13, PHGR13, Lip13, DFGK14, Gro16] (to only mention a few) has made it possible to construct very efficient zk-SNARKs for both the Boolean and the Arithmetic Circuit-SAT and thus for NP. The

[3] ZKProof (`https://zkproof.org/`) being the most notable industry and academic initiative towards a common framework and standards in the field of zero-knowledge has been founded in 2018.

[4] Zero-knowledge proofs are *on the rise* in Gartners' Hype Cycle for Privacy 2019, cf. `https://www.gartner.com/en/documents/3947373/hype-cycle-for-privacy-2019`.

[5] MIT technology review named zk-SNARKS as one of the "10 Breakthrough Technologies of 2018" cf. `https://www.technologyreview.com/lists/technologies/2018/`.

[6] We note that we might drop the zk and simply write SNARK occasionally, though we are always talking about zk-SNARKs.

most efficient known approach for constructing zk-SNARKs for the Arithmetic Circuit-SAT is based on Quadratic Arithmetic Programs (QAPs) [GGPR13], where the prover builds a set of polynomial equations that are then checked by the verifier by using a small number of pairings. The current interest in zk-SNARKs is significant and recently first modular frameworks to efficiently compose zk-SNARKs [CFQ19] and also first important steps towards realizing zk-SNARKs from conjectured post-quantum secure assumptions have been made [GMNO18, BBC$^+$18]. We note that in this work we do not consider recent NIZK proofs that allow larger proof sizes, e.g., logarithmic in the witness size, such as Bulletproofs [BBB$^+$18] or STARKs [BBHR19] but do not require a trusted setup. The currently most efficient zk-SNARK for Arithmetic Circuit-SAT was proposed by Groth [Gro16], who proved it to be knowledge-sound in the generic bilinear group model. In Groth's zk-SNARK, a proof consists of only 3 bilinear group elements and the verifier has to check a single pairing equation.

Strong security for zk-SNARKs. For practical applications of NIZKs even stronger security notions than soundness and knowledge soundness, called simulation soundness (SS) and simulation knowledge soundness (or simply simulation extractability or SE) [Sah99, Sah01]), are required. Informally, these notions require soundness and knowledge soundness respectively to hold even if an adversary is allowed to see an arbitrary number of simulated proofs (which she can obtain adaptively on words of her choice). Firstly, these properties are important if NIZKs are used within larger cryptographic protocols and in particular if they are modeled and analyzed in the universal composability (UC) framework [Can01], as frequently used in blockchain-related protocols (e.g., [JKS16, CDD17, KKKZ19, FMMO19] to name a few). Secondly, NIZKs not satisfying this strong security may face severe threats when used in applications. Therefore, let us informally recall what this property does. It guarantees that proofs are non-malleable in a way that one can neither obtain another valid proof for the same word nor a new proof for a potentially related word from a given proof. Now, let us assume the typical blockchain setting where proofs are incorporated into the state of the blockchain via transactions (e.g., as in Zcash). This means that anyone could take a proof π and obtain from it another new proof π' for the same word and could advertise it as its own proof (as $\pi' \neq \pi$). This is what is often called man-in-the-middle attacks in context of NIZKs and SNARKs (cf. [GM17]). Even worse, it might be possible to obtain from a proof π another proof π' for another word $\mathtt{x}' \neq \mathtt{x}$ (in the same language). For example, if π proves that 100$ are transferred from A to B with transaction $ID = id$, π' might transfer 1000$ from A to B with new $ID = id'$, which can be a devastating attack in systems deployed in the real-world. In fact, ECDSA signature malleability already enabled an attack on Bitcoin that is suspected to have caused a loss of over $ 30 million.[7] Therefore, to avoid such attacks in zk-SNARKs based cryp-

[7] https://www.coindesk.com/study-finds-mt-gox-lost-386-bitcoins-due-transaction-malleability

tocurrencies, non-malleability of the proofs is of utmost importance. All these problems are mitigated by the use of simulation-extractable (SE) zk-SNARKs.

Simulation soundness and simulation extractability can be added generically to any NIZK. Compilers for the former are usually inspired by [Sah01, Gro06] and basically use the idea of extending the language to an OR language where the trapdoor for the OR part can be used to simulate proofs. Extractability can be obtained by additionally encrypting the witness under a public key in the common reference string (CRS) and prove correct encryption [DP92]. The most prominent compiler that exactly follows the ideas outlined before is the C∅C∅ framework [KZM+15] (e.g., used in [AB19, Bag19] and most prominently in the celebrated Hawk paper [KMS+16]). In addition to generic compilers, Groth and Maller in [GM17] initiated the study of ad-hoc constructions of SE zk-SNARKs. This work continued in [BG18] by extending Groth's zk-SNARK [Gro16] in a non black-box way to obtain SE. There is also other recent work in this direction proposing other ad-hoc zk-SNARKs with these properties (cf. [KLO19, Lip19]). Beyond the C∅C∅ framework, which, given the progress in the field of SNARKs (such as universal CRS) and SNARK-friendly primitives, is already quite outdated, there is no work towards lifting zk-SNARKs to SE zk-SNARKs generically.

Trust in CRS generation. Another important aspect for practical applications of zk-SNARKs is the question of the generation of the required common reference string (CRS) [BFM88], a structured random string available to the prover and the verifier. While the CRS model is widely accepted, one has to be very careful to ensure that the CRS has been created honestly, meaning that no one knows the associated trapdoor which allows to break zero-knowledge or soundness. In theory, it is simply assumed that some trusted party will perform the CRS generation, but such a party is hard to find in the real-world. After the Snowden revelations, there has been a recent surge of interest in constructing cryptographic primitives and protocols secure against active subversion and the CRS generation is exactly one of those processes where subversion can happen. In [BFS16], Bellare, Fuchsbauer and Scafuro tackled this problem for NIZK proofs by studying how much security one can still achieve when the CRS generator cannot be trusted. They proved several negative and positive results. In particular, they showed that it is impossible to achieve subversion soundness and (even non-subversion) zero knowledge simultaneously. However, subversion zero-knowledge can be achieved. Later, this notion has also be considered for SNARKs [ABLZ17, Fuc18] and used within practical applications in cryptocurrencies [CGGN17, Fuc19]. For deployed systems such as Zcash and Ethereum, instead of building them on top of subversion-resistant zk-SNARKs, they followed an alternative route to reduce the trust put in the CRS generation. Following a generic method to implement the CRS generation within a secure multi-party computation (MPC) protocol [BCG+15], the CRS for Pinocchio zk-SNARKs [PHGR13] was generated in a large "ceremony" [BGG19]. Coincidentally, they end up with a subversion-resistant zk-SNARK with a polynomial error even in the case where all parties are corrupted, and subversion soundness as long as at least one party is honest. While this is an important achievement, MPC protocols for such tasks in practice are complicated and expensive procedures, requiring much effort besides the technical realization. Thus, more practical solutions are desirable.

Quite recently, to overcome this problem Groth et al. [GKM+18] proposed the notion of a so-called updatable CRS, where everyone can update a CRS and there is a way to check correctness of an update. Here, zero-knowledge holds in the face of a malicious CRS generator and the verifier can trust the CRS (soundness holds) as long as one operation, either the creation of the CRS or one update, have been performed honestly. So the verifier could perform this update operation on its own and then send the CRS to the prover. This updatable setting thus seems much more practical than using MPC protocols, it is more promising than the subversion setting (as it overcomes the impossibility of subversion soundness), and thus has recently attracted lots of researchers studying approaches to realize updatable zk-SNARKs (cf. [MBKM19, GR19, KLO19, CHM+19]).

1.1 Outline

(Revisiting) CØCØ. We first present the CØCØ lifting technique [KZM+15] to generically obtain SE-SNARKs from SNARKs. We discuss the concrete instantiation in [KZM+15] and point to efficiency problems and problems regarding provable security of this instantiation. Then, we extensively investigate alternative provably secure instantiations of their techniques by exploring the design space of many recently proposed SNARK- and STARK-friendly symmetric primitives including the most recent proposals Poseidon [GKK+19] as well as Vision and Rescue [AABS+19]. As these primitives are, however, all very recent and their cryptanalysis either still needs to start or has only recently started [ACG+19, LP19, Bon19, BBUV19], confidence in their proposed security is far from certain. Nevertheless, we provide concrete recommendations for the selection of primitives and provide lower bounds for their efficiency based on the currently available parameters. Additionally, we also propose a more conservative instantiation based on LowMC [ARS+15], which is the oldest of these proposals and has already received independent cryptanalysis [DEM16, BDD+15, DLMW15, RST18]. In comparison to the original CØCØ framework, the revisited CØCØ framework (dubbed OCØCØ) yields an improvement by a factor 10.4x in the number of rank-1 constraints with a conservative choice of symmetric primitives, whereas the most aggressive choice yields an improvement by up to a factor 55.4x.

Lamassu Framework. We then discuss the Lamassu [ARS20] framework that is based on completely different cryptographic primitives. In particular, it is based on the ideas of Derler and Slamanig [DS19] using the notion of key-homomorphic signatures and thus only requires signature schemes. It allows instantiations based on well studied and widely used signature schemes such as ECDSA or EC-Schnorr. Also for Lamassu we discuss concrete choices for the primitives and an extensive comparison with ad-hoc constructions. We show that LAMASSU yields efficient instantiations that compared to OC∅C∅ only need 200 rank-1 constraints more than using the most aggressive parameter choice for the most efficient SNARK-friendly primitive Poseidon in OC∅C∅. For all other and more conservative choices of SNARK-friendly symmetric-key primitives and their parameters, LAMASSU beats them in the number of constraints and outperforms OC∅C∅ by a factor of up to 4.2x. Considering that EC-Schnorr and ECDSA signatures are well established primitives, and that the confidence in their security is far bigger than all the recent SNARK/STARK-friendly primitives, this additional confidence comes at only a very small cost and makes LAMASSU an attractive alternative to C∅C∅.

Subversion and updatable CRS. C∅C∅ as well as OC∅C∅ do not support lifting of subversion or updatable CRS zk-SNARKs to SE subversion or updatable SNARKs. While for the case of subversion zero-knowledge, Baghery in [Bag19] shows that using a part of the C∅C∅ framework (without the encryption of the witness) it is possible to preserve the subversion zero-knowledge property, the case of zk-SNARKS with updatable CRS is more problematic. In particular, the C∅C∅ and OC∅C∅ frameworks cannot be easily made updatable due to the missing algebraic structure in the used primitives, i.e., (hash) commitments.[8] LAMASSU does not suffer from this problem and when using LAMASSU with updatable signatures instead of key-homomorphic signatures, the property of updatability is preserved if the underlying zk-SNARK possesses this property, i.e., is updatable. Updatable signatures can be constructed from key-homomorphic signatures with some additional natural properties and can be constructed from widely used signatures such as EC-Schnorr signatures when instantiated in bilinear groups. Besides, LAMASSU also preserves subversion of the underlying SNARK. Consequently, when starting from an subversion/updatable zk-SNARK, LAMASSU yields SE subversion/updatable SNARKs. Consequently, LAMASSU is the only known framework that allows to generically lift updatable zk-SNARKs to SE updatable SNARKs using well established cryptographic primitives.

2 Preliminaries

Let PPT denote probabilistic polynomial-time. Let $\lambda \in \mathbb{N}$ be the security parameter. All adversaries will be stateful. By $y \leftarrow \mathcal{A}(\mathtt{x}; \omega)$ we denote the fact that $\mathcal{A}$, given an input $\mathtt{x}$ and random coins ω, outputs y. By $x \leftarrow_{\$} \mathcal{D}$ we denote that x is sampled according to distribution $\mathcal{D}$ or uniformly randomly if $\mathcal{D}$ is a set. Let $\mathsf{RND}(\mathcal{A})$ denote the random tape of $\mathcal{A}$, and let $\omega \leftarrow_{\$} \mathsf{RND}(\mathcal{A})$ denote the random choice of the random coins ω from $\mathsf{RND}(\mathcal{A})$. We denote by $\mathsf{negl}(\lambda)$ an arbitrary negligible function. We write $a \approx_\lambda b$ if $|a - b| \leq \mathsf{negl}(\lambda)$. A bilinear

[8] Even using the C∅C∅ framework with commitments that have enough algebraic structure, i.e., use of exponential ElGamal or Pedersen commitments, does not seem to yield updatability. And even if it would work, it would be less efficient than LAMASSU.

group generator $\mathsf{Pgen}(1^\lambda)$ returns $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \bar{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three additive cyclic groups of prime order p, and $\bar{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate efficiently computable bilinear map (pairing).

2.1 Pseudorandom Functions

We recall the standard notion of pseudorandom functions.

Definition 1 (PRF). *Let $f\colon \mathcal{S} \times D \to \mathsf{R}$ be a family of functions and let Γ be the set of all functions $D \to \mathsf{R}$. f is a pseudorandom function (PRF) (family) if it is efficiently computable and for all PPT distinguishers $\mathcal{D}$ such that*

$$\left|\Pr\left[s \leftarrow_{\$} \mathcal{S}, \mathcal{D}^{f_s(\cdot)}(1^\kappa)\right] - \Pr\left[g \leftarrow_{\$} \Gamma, \mathcal{D}^{g(\cdot)}(1^\kappa)\right]\right| \approx_\lambda 0.$$

2.2 *X*-SNARK

In the following we provide a formal definition of SNARKs (cf. Appendix A.1 for the basic definition of NIZK proofs).

Definition 2 (SNARK). *A non-interactive system Π is a succinct non-interactive argument of knowledge (SNARK) for relation generator* RGen *if it is complete and knowledge sound, and moreover succinct, meaning that for all λ, all $(\mathcal{R}, \mathsf{aux}_\mathcal{R}) \in \mathrm{image}(\mathsf{RGen}(1^\lambda))$, all $\mathtt{crs} \leftarrow \mathsf{KGen}(\mathcal{R}, \mathsf{aux}_\mathcal{R})$, all $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$ and all proofs $\pi \leftarrow \mathsf{P}(\mathcal{R}, \mathsf{aux}_\mathcal{R}, \mathtt{crs}, \mathtt{x}, \mathtt{w})$ we have $|\pi| = poly(\lambda)$ and $\mathsf{V}(\mathcal{R}, \mathsf{aux}_\mathcal{R}, \mathtt{crs}, \mathtt{x}, \pi)$ runs in time polynomial in $\lambda + |\mathtt{x}|$. Π is a zk-SNARK if it additionally satisfies zero-knowledge and an SE (zk-)SNARK if instead of knowledge soundness it provides strong simulation extractability.*

We adopt the (SE) X-SNARK definitions from [ABLZ17, Fuc18, GKM+18] where $X \in \{\mathsf{trusted}, \mathsf{subverted}, \mathsf{updatable}\}$. In other words, besides considering the standard setting with a trusted CRS generation, we also capture the subversion and updatable CRS setting. Trusted means generated by a trusted third party, or even a more general MPC protocol, subverted means that the setup generator gets the CRS from the adversary and uses it after checking that it is well formed, and, updatable means that an adversary can adaptively generate sequences of CRSs and arbitrarily interleave its own malicious updates into them. The only constraints on the final CRS are that it is well formed and that at least one honest participant has contributed to it by providing an update.

A *X-SNARK* $\Pi = (\mathsf{KGen}, \mathsf{Ucrs}, \mathsf{Vcrs}, \mathsf{P}, \mathsf{V})$ for RGen consists of the following PPT algorithms (it contains Vcrs when $X = \mathsf{subverted}$ and contains Ucrs and Vcrs when $X = \mathsf{update}$):

$\mathsf{KGen}_{\mathsf{crs}}(\mathcal{R}, \mathsf{aux}_\mathcal{R})$: On input $(\mathcal{R}, \mathsf{aux}_\mathcal{R}) \in \mathrm{image}(\mathsf{RGen}(1^\lambda))$, outputs CRS $\mathtt{crs}$ and trapdoor $\mathtt{tc}$.

$\mathsf{Ucrs}(\mathcal{R}, \mathtt{crs})$: On input $(\mathcal{R}, \mathtt{crs})$ outputs $(\mathtt{crs}_{\mathsf{up}}, \zeta_{\mathsf{up}})$ where $\mathtt{crs}_{\mathsf{up}}$ is the updated CRS and ζ_{up} is a proof for the correctness of the updating procedure.

$\mathsf{Vcrs}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \zeta)$: On input $(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \zeta)$, returns either 0 (the CRS is ill-formed) or 1 (the CRS is well-formed).

$\mathsf{P}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathtt{w})$: On input $(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathtt{w})$, where $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$, output a proof π.

$\mathsf{V}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \pi)$: On input $(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \pi)$, returns either 0 (reject) or 1 (accept).

$\mathsf{Sim}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathtt{tc})$: In input $(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathtt{tc})$, outputs a simulated proof π.

Definition 3. *Let* $\Pi = (\mathsf{KGen}_{\mathsf{crs}}, \mathsf{Ucrs}, \mathsf{Vcrs}, \mathsf{P}, \mathsf{V})$ *be a non-interactive argument for the relation* $\mathcal{R}$*. Then the argument* Π *is* X*-secure for* $X \in \{\mathsf{trusted}, \mathsf{subverted}, \mathsf{updatable}\}$*, if it satisfies the following properties:*

X-Completeness. Π *is* complete for RGen*, if for all* λ*,* $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$*, and PPT algorithms* A*,*

$$\Pr\left[\begin{array}{l}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}) \leftarrow \mathsf{RGen}(1^{\lambda}), (\mathtt{crs}, \mathtt{tc}, \zeta) \leftarrow \mathsf{A}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}),\\ 1 \leftarrow \mathsf{Vcrs}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \zeta):\\ \mathsf{V}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathsf{P}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathtt{w})) = 1\end{array}\right] =_{\lambda} 1.$$

Where ζ *is a proof for the correctness of the generating (or updating) the CRS. If* $X = \mathsf{trusted}$ *then* A *is* $\mathsf{KGen}_{\mathsf{crs}}$ *and* $\zeta = \perp$ *and* A *is adversary* $\mathcal{A}$ *otherwise.*

X-Strong simulation extractability. *For* $X \in \{\mathsf{trusted}, \mathsf{subverted}\}$*,* Π *is* strong simulation extractable for RGen*, if for every PPT* $\mathcal{A}$*, there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{A}}$*, s.t.* $\forall$ λ*,*

$$\Pr\left[\begin{array}{l}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}) \leftarrow \mathsf{RGen}(1^{\lambda}),\\ (\mathtt{crs}, \mathtt{tc}) \leftarrow \mathsf{KGen}_{\mathsf{crs}}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}), \omega_{\mathcal{A}} \leftarrow_{\$} \mathsf{RND}(\mathcal{A}),\\ (\mathtt{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{O}(\cdot)}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}; \omega_{\mathcal{A}}),\\ \mathtt{w} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}; \omega_{\mathcal{A}}):\\ (\mathtt{x}, \pi) \notin \mathcal{Q} \wedge (\mathtt{x}, \mathtt{w}) \notin \mathcal{R} \wedge\\ \mathsf{V}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \pi) = 1\end{array}\right] \approx_{\lambda} 0.$$

Here, $\mathsf{O}(\mathtt{x})$ *returns* $\pi := \mathsf{Sim}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathtt{tc})$ *and keeps track of all queries and the result,* $(\mathtt{x}, \pi)$*, via* $\mathcal{Q}$*. For* $X = \mathsf{updatable}$*,* Π *is* strong simulation extractable for RGen*, if for every PPT* $\mathcal{A}$ *and any subverter* Z*, there exists a*

PPT extractor $\mathsf{Ext}_{\mathcal{A}}$*, s.t.* $\forall\ \lambda$*,*

$$\Pr\left[\begin{array}{l}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}) \leftarrow \mathsf{RGen}(1^{\lambda}),\\ (\mathtt{crs}, \mathtt{tc}) \leftarrow \mathsf{KGen}_{\mathsf{crs}}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}})\\ \omega_{\mathsf{Z}} \leftarrow_{\$} \mathsf{RND}(\mathsf{Z}),\\ (\mathtt{crs}_{\mathsf{up}}, \zeta_{\mathsf{up}}, \mathsf{aux}_{\mathsf{Z}}) \leftarrow \mathsf{Z}(\mathtt{crs}, \{\zeta_i\}_{i=1}^{i=n}, \omega_{\mathsf{Z}}),\\ \textbf{if } \mathsf{Vcrs}(\mathtt{crs}, \{\zeta_i\}_{i=1}^{i=n}) = 0 \textbf{ then return } 0,\\ \omega_{\mathcal{A}} \leftarrow_{\$} \mathsf{RND}(\mathcal{A}),\\ (\mathtt{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{O}(\cdot)}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}_{\mathsf{up}}, \mathsf{aux}_{\mathsf{Z}}; \omega_{\mathcal{A}}),\\ \mathtt{w} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}_{\mathsf{up}}, \mathsf{aux}_{\mathsf{Z}}; \omega_{\mathcal{A}}):\\ (\mathtt{x}, \pi) \notin \mathcal{Q} \wedge (\mathtt{x}, \mathtt{w}) \notin \mathcal{R} \wedge\\ \mathsf{V}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}_{\mathsf{up}}, \mathtt{x}, \pi) = 1\end{array}\right] \approx_{\lambda} 0.$$

Here $\mathsf{RND}(\mathsf{Z}) = \mathsf{RND}(\mathcal{A})$ *and* ζ *is a proof for the correctness of the updating procedure.* $\mathsf{O}(\mathtt{x})$ *returns* $\pi := \mathsf{Sim}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathtt{tc})$ *and keeps track of all queried* $(\mathtt{x}, \pi)$ *via* $\mathcal{Q}$*. Note that* Z *can also first generate* $\mathtt{crs}$ *and then an honest updater updates it and outputs* $\mathtt{crs}_{\mathsf{up}}$*.*

X**-Zero-knowledge.** *For* X = trusted*,* Π *is* statistically unbounded ZK for RGen *[Gro06], if for all* λ*, all* $(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}) \in \mathrm{im}(\mathsf{RGen}(1^{\lambda}))$*, and all computationally unbounded* $\mathcal{A}$*,* $\varepsilon_0^{unb} \approx_{\lambda} \varepsilon_1^{unb}$*, where*

$$\varepsilon_b^{unb} = \Pr\left[(\mathtt{crs}, \mathtt{tc}) \leftarrow \mathsf{KGen}_{\mathsf{crs}}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}) : \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}) = 1 \quad\right].$$

Here, the oracle $\mathsf{O}_0(\mathtt{x}, \mathtt{w})$ *returns* $\perp$ *(reject) if* $(\mathtt{x}, \mathtt{w}) \notin \mathcal{R}$*, and otherwise it returns* $\mathsf{P}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathtt{w})$*. Similarly,* $\mathsf{O}_1(\mathtt{x}, \mathtt{w})$ *returns* $\perp$ *(reject) if* $(\mathtt{x}, \mathtt{w}) \notin \mathcal{R}$*, and otherwise it returns* $\mathsf{Sim}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathtt{tc})$*.* Π *is* perfectly unbounded ZK for RGen *if one requires that* $\varepsilon_0^{unb} = \varepsilon_1^{unb}$*.*

For $X \in \{\mathsf{subverted}, \mathsf{updatable}\}$*,* Π *is* statistically unbounded X-ZK for RGen *[ABLZ17, Fuc18, GKM*[+]*18], if for any PPT* Z *there exists a PPT* $\mathsf{Ext}_{\mathsf{Z}}$*, such that for all* λ*,* $(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}) \in \mathrm{im}(\mathsf{RGen}(1^{\lambda}))$*, and computationally unbounded* $\mathcal{A}$*,* $\varepsilon_0^{unb} \approx_{\lambda} \varepsilon_1^{unb}$*, where*

$$\varepsilon_b^{unb} = \Pr\left[\begin{array}{l}\omega_{\mathsf{Z}} \leftarrow_{\$} \mathsf{RND}(\mathsf{Z}), (\mathtt{crs}, \zeta, \mathsf{aux}_{\mathsf{Z}}) \leftarrow \mathsf{Z}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}; \omega_{\mathsf{Z}}),\\ \mathtt{tc} \leftarrow \mathsf{Ext}_{\mathsf{Z}}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}; \omega_{\mathsf{Z}}):\\ \mathsf{Vcrs}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \zeta) = 1 \ \wedge \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{tc}, \mathsf{aux}_{\mathsf{Z}}) = 1\end{array}\right].$$

Here $\mathsf{RND}(\mathsf{Z}) = \mathsf{RND}(\mathcal{A})$*, the oracle* $\mathsf{O}_0(\mathtt{x}, \mathtt{w})$ *returns* $\perp$ *(reject) if* $(\mathtt{x}, \mathtt{w}) \notin \mathcal{R}$*, and otherwise it returns* $\mathsf{P}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathtt{w})$*. Similarly,* $\mathsf{O}_1(\mathtt{x}, \mathtt{w})$ *returns* $\perp$ *(reject) if* $(\mathtt{x}, \mathtt{w}) \notin \mathcal{R}$*, and otherwise it returns* $\mathsf{Sim}(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}, \mathtt{crs}, \mathtt{x}, \mathtt{tc})$*.* Π *is* perfectly unbounded X-ZK for RGen *if one requires that* $\varepsilon_0^{unb} = \varepsilon_1^{unb}$*.*

In all of the above properties, $\mathsf{aux}_{\mathcal{R}}$ *can be seen as a common auxiliary input to algorithm* A *that is generated by using a benign [BCPR14] relation generator; we recall that we just think of* $\mathsf{aux}_{\mathcal{R}}$ *as being the description of a bilinear group.*

We note that what is called strong simulation-sound extractability in this work (in order to be consistent with [KZM⁺15]) is often simply called simulation-sound extractability (e.g., in [DS19] which will be the basis for the LAMASSU framework). For completeness, quadratic arithmetic programs and rank 1 constraint systems are discussed in Appendix A.2

2.3 Signature Schemes

A signature scheme $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ consists of the following PPT algorithms:

$\mathsf{KGen}(1^\kappa)$: On input security parameter κ it outputs a signing key sk and a verification key pk with associated message space $\mathcal{M}$.

$\mathsf{Sign}(\mathsf{sk}, m)$: On input a secret key sk and a message $m \in \mathcal{M}$ it outputs a signature σ.

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: On input a public key pk, a message $m \in \mathcal{M}$ and a signature σ it outputs a bit $b \in \{0, 1\}$.

We note that for a signature scheme many independently generated public keys may be with respect to the same parameters PP, e.g., some elliptic curve group parameters. In such a case we use an additional algorithm PGen and $\mathsf{PP} \leftarrow \mathsf{PGen}(1^\kappa)$ is then given to KGen. We assume that a signature scheme satisfies the usual (perfect) correctness notion. Below, we present the standard existential unforgeability under adaptively chosen message attacks ($\mathsf{EUF\text{-}CMA}$ security) notion.

Definition 4 ($\mathsf{EUF\text{-}CMA}$). *A signature scheme Σ is* $\mathsf{EUF\text{-}CMA}$ *secure, if for all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\begin{aligned}&(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa), (m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{pk}): \\ &\mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1 \ \wedge \ m^\star \notin \mathcal{Q}^{\mathsf{Sign}}\end{aligned}\right] \approx_\lambda 0,$$

where the environment keeps track of the queries to the signing oracle via $\mathcal{Q}^{\mathsf{Sign}}$.

The compiler from [ARS20] also requires one-time signature schemes that are $\mathsf{sEUF\text{-}CMA}$ secure (also called sOTS schemes).

Definition 5 (Strong One-Time Signature Scheme). *A strong one-time signature scheme Σ_{OT} is a signature scheme Σ which satisfies the following unforgeability notion: For all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\begin{aligned}&(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa), (m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{pk}): \\ &\mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1 \ \wedge \ (m^\star, \sigma^\star) \notin \mathcal{Q}^{\mathsf{Sign}}\end{aligned}\right] \approx_\lambda 0,$$

where the oracle $\mathsf{Sign}(\mathsf{sk}, m) \coloneqq \Sigma.\mathsf{Sign}(\mathsf{sk}, m)$ can only be called once.

2.4 Key-Homomorphic Signatures

We recall relevant parts of the definitional framework of key-homomorphic signatures as introduced in [DS19]. Let $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme and the secret and public key elements live in groups $(\mathbb{H}, +)$ and $(\mathbb{E}, \cdot)$, respectively. For these two groups is is required that group operations, inversions, membership testing as well as sampling from the uniform distribution are efficient.

Definition 6 (Secret Key to Public Key Homomorphism). *A signature scheme Σ provides a secret key to public key homomorphism, if there exists an efficiently computable map $\mu : \mathbb{H} \to \mathbb{E}$ such that for all $\mathsf{sk}, \mathsf{sk}' \in \mathbb{H}$ it holds that $\mu(\mathsf{sk} + \mathsf{sk}') = \mu(\mathsf{sk}) \cdot \mu(\mathsf{sk}')$, and for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}$, it holds that $\mathsf{pk} = \mu(\mathsf{sk})$.*

In the discrete logarithm setting, it is usually the case $\mathsf{sk} \leftarrow \mathbb{Z}_p$ and $\mathsf{pk} = g^{\mathsf{sk}}$ with g being the generator of some group $\mathbb{G}$ of prime order p, e.g., for EdDSA/ECDSA or EC-Schnorr signatures (cf. [DS19] for a detailed exposition).

Definition 7 (Key-Homomorphic Signatures). *A signature scheme is called key-homomorphic, if it provides a secret key to public key homomorphism and an additional PPT algorithm* Adapt*, defined as:*

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: *Given a public key pk, a message m, a signature σ, and a shift amount Δ outputs a public key pk' and a signature σ',*

such that for all $\Delta \in \mathbb{H}$ and all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\kappa)$, all messages $m \in \mathcal{M}$ and all $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ and $(\mathsf{pk}', \sigma') \leftarrow \mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$ it holds that

$$\Pr[\mathsf{Verify}(\mathsf{pk}', m, \sigma') = 1] = 1 \quad \wedge \quad \mathsf{pk}' = \mu(\Delta) \cdot \mathsf{pk}.$$

The following notion covers whether adapted signatures look like freshly generated signatures, where we do not need the strongest notion in [DS19], which requires this to hold even if the initial signature used in Adapt is known.

Definition 8 (Adaptability of Signatures). *A key-homomorphic signature scheme provides adaptability of signatures, if for every $\kappa \in \mathbb{N}$ and every message $m \in \mathcal{M}$, it holds that*

$$[(\mathsf{sk}, \mathsf{pk}), \mathsf{Adapt}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m), \Delta)],$$

where $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$, $\Delta \leftarrow \mathbb{H}$, *and*

$$[(\mathsf{sk}, \mu(\mathsf{sk})), (\mu(\mathsf{sk}) \cdot \mu(\Delta), \mathsf{Sign}(\mathsf{sk} + \Delta, m))],$$

where $\mathsf{sk} \leftarrow \mathbb{H}$, $\Delta \leftarrow \mathbb{H}$, *are identically distributed.*

For illustration purposes we will use the Schnorr signature scheme [Sch90], which is very popular in the blockchain and distributed ledger domain, and whose adaption notion we discuss in Appendix A.4.

2.5 Updatable Signature Schemes

Updatable signatures are signatures that allow updates on the key and unforgeability guarantees need to hold as long as either the initial key generation or at least one of the updates was performed honestly. However, signing is performed honestly. We note that like in Groth et al. [GKM+18] for updatable CRS (using Lemma 6), one models only a single update as a single adversarial update implies updatable signatures with arbitrary many updates.

Definition 9 (Updatable signature schemes). *An updatable signature scheme* $\Sigma = (\mathsf{KGen}, \mathsf{Ucrs}, \mathsf{Vpk}, \mathsf{Sign}, \mathsf{Verify})$ *consists of the following PPT algorithms:*

$\mathsf{KGen}(1^\lambda)$: *Given a security parameter* λ *it outputs a signing key* sk *and a verification key* pk *with associated message space* $\mathcal{M}$*.*

$\mathsf{Upk}(\mathsf{pk})$: *Given a verification key* pk *it outputs an updated verification key* $\mathsf{pk}_{\mathsf{up}}$ *with associated secret updating key* $\mathsf{up}_{\mathsf{sk}}$*, and a proof* ζ*.*

$\mathsf{Vpk}(\mathsf{pk}, \mathsf{pk}_{\mathsf{up}}, \zeta)$: *Given a verification key* pk*, a potentially updated verification key* $\mathsf{pk}_{\mathsf{up}}$*, and the proof* ζ *it checks if* $\mathsf{pk}_{\mathsf{up}}$ *has been updated correctly.*

$\mathsf{Sign}(\mathsf{sk}, m)$: *Given potentially updated secret key* sk *(in case of* $\mathsf{sk}_{\mathsf{up}}$ *it contains* sk *and* $\mathsf{up}_{\mathsf{sk}}$*) and a message* $m \in \mathcal{M}$ *it outputs a signature* σ*.*

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: *Given potentially updated public key* pk*, a message* $m \in \mathcal{M}$ *and a signature* σ *it outputs a bit* $b \in \{0,1\}$*.*

Definition 10 (Updatable correctness). *A signature scheme* Σ *is updatable correct, if*

$$\Pr\left[\begin{array}{l}(\mathsf{sk}, \mathsf{pk}, \zeta) \leftarrow \mathsf{KGen}(1^\lambda), (\mathsf{up}_{\mathsf{sk}}, \mathsf{pk}_{\mathsf{up}}, \zeta_{\mathsf{up}}) \leftarrow \mathsf{Upk}(\mathsf{pk}),\\ \mathsf{Vpk}(\mathsf{pk}, \mathsf{pk}_{\mathsf{up}}, \zeta_{\mathsf{up}}) = 1 \ \wedge \ \mathsf{Vpk}(\mathsf{pk}, \mathsf{pk}, \zeta) = 1 :\\ \mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1 \ \wedge\\ \mathsf{Verify}(\mathsf{pk}_{\mathsf{up}}, m, \mathsf{Sign}(\mathsf{sk}_{\mathsf{up}}, m)) = 1\end{array}\right] = 1,$$

where the probability is taken over the randomness of the signing algorithm.

Definition 11 (Updatable strong key hiding). *We have that for* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$ *and* $(\mathsf{up}_{\mathsf{sk}}, \mathsf{pk}_{\mathsf{up}}, \zeta_{\mathsf{up}}) \leftarrow \mathsf{Upk}(\mathsf{pk})$ *it holds that* $(\mathsf{sk}, \mathsf{pk}) \approx_\lambda (\mathsf{sk}_{\mathsf{up}}, \mathsf{pk}_{\mathsf{up}})$ *if one of the following setting holds,*

- *the original* pk *was honestly generated and the key-update verifies:* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$ *and* $\mathsf{Vpk}(\mathsf{pk}, \mathsf{pk}_{\mathsf{up}}, \zeta_{\mathsf{up}}) = 1$*.*
- *the original* pk *verifies and the key-update was honest:* $\mathsf{Vpk}(\mathsf{pk}, \mathsf{pk}, \zeta) = 1$*, and* $(\mathsf{up}_{\mathsf{sk}}, \mathsf{pk}_{\mathsf{up}}, \zeta_{\mathsf{up}}) \leftarrow \mathsf{Upk}(\mathsf{pk})$*.*

Now, we present the updatable EUF-CMA security notion.

Definition 12 (Updatable EUF-CMA). *A signature scheme* Σ *is updatable* EUF-CMA *secure, if for all PPT subverter* Z*, there exists a PPT extractor* $\mathsf{Ext}_{\mathsf{Z}}$*, s.t. for all* λ*, and all PPT adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{l}(\mathsf{sk}, \mathsf{pk}, \zeta) \leftarrow \mathsf{KGen}(1^\kappa), \omega_{\mathsf{Z}} \leftarrow_{\$} \mathsf{RND}(\mathsf{Z}), (\mathsf{pk}_{\mathsf{up}}, \zeta_{\mathsf{up}}, \mathsf{aux}_{\mathsf{Z}}) \leftarrow \mathsf{Z}(\mathsf{pk}; \omega_{\mathsf{Z}}),\\ \mathsf{up}_{\mathsf{sk}} \leftarrow \mathsf{Ext}_{\mathsf{Z}}(\mathsf{pk}, \omega_{\mathsf{Z}}), (m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_{\mathsf{up}}, \cdot)}(\mathsf{pk}_{\mathsf{up}}, \mathsf{aux}_{\mathsf{Z}}):\\ \mathsf{Vpk}(\mathsf{pk}, \mathsf{pk}_{\mathsf{up}}, \zeta_{\mathsf{up}}) = 1 \wedge \mathsf{Verify}(\mathsf{pk}_{\mathsf{up}}, m^\star, \sigma^\star) = 1 \ \wedge \ m^\star \notin \mathcal{Q}^{\mathsf{Sign}}\end{array}\right] \approx_\lambda 0,$$

where the environment keeps track of the queries to the oracle via $\mathcal{Q}^{\mathsf{Sign}}$*. Note that* Z *can also generate the initial* pk *and the an honest updater* Upk *updates it and outputs* $\mathsf{pk}_{\mathsf{up}}$*,* $\mathsf{sk}_{\mathsf{up}}$*, and the proof* ζ *(then we require that* $\mathsf{Vpk}(\mathsf{pk}, \mathsf{pk}, \zeta) = 1$*).*

In [ARS20] the following is shown.

Theorem 1. *Every correct and* EUF-CMA *secure key-homomorphic signature scheme* Σ *that is adaptable according to Definition 8 and provides an efficient extractor* $\mathsf{Ext}_{\mathsf{Z}}$ *satisfies updatable correctness, updatable strong key hiding and updatable* EUF-CMA *security.*

3 Lifting Transformations for SE Subversion/Updatable SNARKs

In this section we briefly present the C∅C∅ framework, then a revisited version of the C∅C∅ framework (called OC∅C∅) and finally the Lamassu framework.

3.1 The C∅C∅ Framework

Kosba et al. [KZM+15] proposed lifting transformations for SNARKs in three different versions basic, improved lifting, and the strengthening lifting. We only consider the strongest version which lifts a SNARK to a strongly simulation extractable (SE) SNARK. In particular, their construction, which we recall in Fig. 1, transforms any NIZK Π to one that satisfies SE. Given a language $\mathcal{L}$ with NP relation $\mathcal{R}_{\mathcal{L}}$, let $\mathcal{L}'$ be s.t. $\{(\mathtt{x}, \mathsf{c}, \mu, \mathsf{pk}_{\mathsf{OT}}, \mathsf{pk}_e, \rho), (\mathtt{w}, r_1, r_0, s_0)\} \in \mathcal{R}_{\mathcal{L}'}$ iff:

$$\begin{aligned}&\boxed{\mathsf{c} = \Omega.\mathsf{Enc}(\mathsf{pk}_e, \mathtt{w}; r_1) \wedge}\ ((\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}}\ \vee \\ &(\mu = f_{s_0}(\mathsf{pk}_{\mathsf{OT}})\ \wedge\ \rho = \mathsf{Commit}(s_0; r_0)))\ ,\end{aligned} \tag{1}$$

where pk_e is the public key of a public key encryption scheme Ω (cf. Appendix A.3), f is a pseudorandom function, and $\mathsf{pk}_{\mathsf{OT}}$ is the verification key of a strong one-time signature (OTS) scheme Σ_{OT} (cf. Definition 5). Note that extraction is defined here with respect to a black-box extractor (i.e., decrypting to obtain the witness), which Kosba et al. [KZM+15] do to support UC-security. If this is not required, then one can use the non black-box extractor of the underlying SNARK and simplify the language $\mathcal{L}'$ by removing the part in the gray box, which we will do subsequently (cf. [Bag19] for a formal treatment). In this case, C∅C∅ retains subversion resistance of the underlying SNARK.

3.2 The Revisited C∅C∅ Framework OC∅C∅

The most efficient version of the C∅C∅ framework is based on a commitment and PRF evaluation (Equation (1) without the gray box). Kosba et al. [KZM+15] proposed to instantiate the commitment and the PRF using hash functions, and in

$\underline{\mathsf{KGen}_{\mathsf{crs}}(1^\lambda, \mathcal{L})}$

- $\Pi.\mathtt{crs} \leftarrow \Pi.\mathsf{KGen}; (\mathsf{pk}_e, \mathsf{sk}_e) \leftarrow \Omega.\mathsf{KGen}(1^\lambda);$
- $\mathtt{tc} \leftarrow (s_0, r_0) \leftarrow_\$ \{0,1\}^\lambda; \rho \leftarrow \mathsf{Commit}(s_0; r_0);$
- **return** $(\mathtt{crs} := (\Pi.\mathtt{crs}, \mathsf{pk}_e, \rho), \mathtt{tc}_{\mathsf{ext}} := \mathsf{sk}_e)$

$\underline{\mathsf{P}(\mathtt{crs}, \mathtt{x}, \mathtt{w})}$

- $(\mathsf{pk}_{\mathsf{OT}}, \mathsf{sk}_{\mathsf{OT}}) \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{KGen}(1^\lambda); r_1, z_0, z_1, z_2 \leftarrow_\$ \{0,1\}^\lambda;$
- $\mathsf{c} = \Omega.\mathsf{Enc}(\mathsf{pk}_e, \mathtt{w}; r_1); \mu \leftarrow z_0;$
- $\pi_\Pi \leftarrow \Pi.\mathsf{P}(\Pi.\mathtt{crs}, (\mathtt{x}, \mathsf{c}, \mathsf{pk}_e, \mathsf{pk}_{\mathsf{OT}}, \mu, \rho), (\mathtt{w}, r_1, z_1, z_2));$
- $\sigma_{\mathsf{OT}} \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{OT}}, (\mathtt{x}, \mathsf{c}, \mu, \pi_\Pi));$
- **return** $\pi := (\mathsf{c}, \mu, \pi_\Pi, \mathsf{pk}_{\mathsf{OT}}, \sigma_{\mathsf{OT}});$

$\underline{\mathsf{V}(\mathtt{crs}, \mathtt{x}, \pi)}$

- **if** $\Sigma_{\mathsf{OT}}.\mathsf{Verify}(\mathsf{pk}_{\mathsf{OT}}, (\mathtt{x}, \mathsf{c}, \mu, \pi_\Pi, \sigma_{\mathsf{OT}})) = 0$
- $\vee\, \Pi.\mathsf{V}(\Pi.\mathtt{crs}, \mathtt{x}, \mathsf{c}, \mu, \mathsf{pk}_e, \mathsf{pk}_{\mathsf{OT}}, \rho, \pi_\Pi) = 0$
- **then return** 0 **else return** 1;

$\underline{\mathsf{Sim}(\mathtt{crs}, \mathtt{x}, \mathtt{tc})}$

- $(\mathsf{pk}_{\mathsf{OT}}, \mathsf{sk}_{\mathsf{OT}}) \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{KGen}(1^\lambda); \mu = f_{s_0}(\mathsf{pk}_{\mathsf{OT}});$
- $r_1, z_3 \leftarrow_\$ \{0,1\}^\lambda; \mathsf{c} \leftarrow \Omega.\mathsf{Enc}(\mathsf{pk}_e, z_3; r_1); \mathtt{w} \leftarrow z_3;$
- $\pi_\Pi \leftarrow \Pi.\mathsf{P}(\Pi.\mathtt{crs}, (\mathtt{x}, \mathsf{c}, \mathsf{pk}_e, \mathsf{pk}_{\mathsf{OT}}, \mu, \rho), (\mathtt{w}, r_1, r_0, s_0));$
- $\sigma_{\mathsf{OT}} \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{OT}}, (\mathtt{x}, \mathsf{c}, \mu, \pi_\Pi));$
- **return** $\pi = (\mathsf{c}, \mu, \pi_\Pi, \mathsf{pk}_{\mathsf{OT}}, \sigma_{\mathsf{OT}});$

$\underline{\mathsf{Ext}(\mathtt{crs}, \mathtt{x}, \pi, \mathtt{tc}_{\mathsf{ext}})}$

- **return** $\mathtt{w} \leftarrow \Omega.\mathsf{Dec}(\mathtt{tc}_{\mathsf{ext}}, \mathsf{c});$

Fig. 1. The strong version of the C∅C∅ transformation.

particular SHA-256. Similarly, the commitment is instantiated as hash commitment using the same hash function. With the development of SNARK/STARK-friendly primitives soon after the introduction of the C∅C∅ framework, we observe that this choice is non-optimal from an efficiency point of view. Moreover, the choice of the commitment is also problematic in a different sense, because the specific commitment used in C∅C∅ is secure in the random oracle model (ROM). Since this implies that statements need to be proven with respect to the preimage of a random oracle, instantiating the framework in a provable secure way is not possible. Consequently, we discuss an alternative approach to commit to the PRF key. It can be instantiated in a provably secure way and, on top of that, is also more efficient while still relying on symmetric-key primitives only.

The problem in the symmetric setting is to find efficient binding commitments. The signature scheme construction in [DOR+16] based on the Bellare-Goldwasser paradigm [BG90] also needs to "commit" to a PRF key. There, signatures consists of a simulation extractable NIZK proof of a PRF key, where the PRF is built from symmetric-key primitives. The standard notion of PRF security, however, does not immediately imply any binding property on the key. Therefore, the construction relies on a *computational fixed-valued-key-binding* PRF [CMR98, Fis99], i.e., a PRF f with the additional property that there exists a β such that for a PRF key s and given $y = f_s(\beta)$ it is hard to provide a second PRF key s', $s \neq s'$, satisfying $y = f_{s'}(\beta)$:

Definition 13 (Computational Fixed-Value-Key-Binding PRF). *A PRF family $f \colon \mathcal{S} \times D \to \mathsf{R}$ is computationally key-binding if there exists a special value $\beta \in D$ so that it holds for all adversaries $\mathcal{A}$ that:*

$$\Pr\Big[s \leftarrow_{\$} \mathcal{S},\ s' \leftarrow \mathcal{A}^{f_s(\cdot)}(f_s(\beta), \beta) \colon f_{s'}(\beta) = f_s(\beta)\Big] \approx_\lambda 0.$$

Extending the public key with the PRF evaluation at β and proving its well-formedness is then sufficient to "commit" to the PRF key.[9]

For C∅C∅, we can apply the same idea: we replace the commitment to the PRF key with the evaluation of the PRF at β and adapt the language accordingly. That is, for the construction depicted in Fig. 1[10], let the language $\mathcal{L}'$ be such that $\{(\mathtt{x}, \mu, \mathsf{pk}_{\mathsf{OT}}, \rho, \beta), (\mathtt{w}, s_0)\} \in \mathcal{R}_{\mathcal{L}'}$ if and only if:

$$\left\{(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}}\ \vee (\mu = f_{s_0}(\mathsf{pk}_{\mathsf{OT}})\ \wedge\ \rho = f_{s_0}(\beta))\right\}.$$

We denote the C∅C∅ framework using the language $\mathcal{L}'$ as Optimized C∅C∅, or OC∅C∅ for short. For the security proofs (Theorem 2 and Theorem 3 in [Bag19]), we note for each game change based on computational hiding of the commitment, we now use the PRF property and replace them with the evaluation of a random function (Lemma 4). For the step relying on the commitment scheme's binding property (Lemma 2), one can now argue the uniqueness of the PRF key using the fixed-value-key-binding property of the PRF. One obtains the following:

Corollary 1. *If the underlying NIZK scheme satisfies perfect completeness, knowledge soundness, subversion zero-knowledge, the PRF is secure and computationally fixed-value-key-binding, and the one-time signature is* sEUF-CMA *secure, then OC∅C∅ is a zero-knowledge proof system satisfying perfect completeness, subversion zero-knowledge, and strong simulation extractability.*

Instantiating the OC∅C∅ Framework. When instantiating the original C∅C∅ framework or OC∅C∅, SHA-256 as well as any other variant of the SHA2 family or the SHA3 family are non-optimal choices from a efficiency point of view. Indeed, representing the SHA-256 compression function as R1CS requires 22,272 constraints [CGGN17]. The permutation used in SHA3 is even more expensive with 38,400 constraints [AGR+16]. Recent lines of work specifically designed block ciphers and hash functions that work especially well in the context of SNARKs. These include MiMC [AGR+16], GMiMC [AGP+19], Poseidon [GKK+19], Friday [AD18], Vision and Rescue [AABS+19], which all were

[9] Similarly, [DRS18] employs the same idea to commit to a PRF key.

[10] Now, one will use the non black-box extractor of the underlying NIZK instead of the black-box extractor Ext from Fig. 1.

specifically designed with SNARK/STARK-based applications in mind. We however want to note that these designs are all relatively young and were not available at the time C∅C∅ was proposed.

Since those designs are all very recent, their cryptanalysis is still ongoing. Friday suffers from a Gröbner-basis attack [ACG+19], the key schedule of some variants of MiMC can be attacked using an interpolation attack [LP19] and they also suffer from a collision attack [Bon19], which can also be applied to some variants of GMiMC. Notably, the designs also received some interest as part of a hash collision challenge for STARK-friendly designs,[11] where collisions have been found for low-security instances already. Therefore, we will only include instances in our evaluation that – to the best of our knowledge – have not been broken so far.

Even though these symmetric primitives are designed for SNARKs, they often run into practical problems. For instance, one of the popular choices for instantiating SNARKs is the pairing-friendly BLS12-381[12] curve. However, its group order q does not match MiMC's and GMiMC's requirement coming from the choice of $x \mapsto x^3$ as Sbox that $\gcd(q-1, 3) = 1$. Additionally, MiMC operates in large prime fields, requiring one to emulate the required fields on top of $\mathbb{F}_q$. The latter issue is solved by GMiMC working over smaller fields, but the order requirement is still an issue. Poseidon, which allows one to choose $x \mapsto x^5$ as Sbox meaning that $\gcd(q-1, 5) = 1$ needs to be satisfied, fixes both problems and can be directly implemented in $\mathbb{F}_q$ arithmetic. Similarly, Rescue faces similar issues as the Sboxes used there are $x \mapsto x^\alpha$ and $x \mapsto x^{1/\alpha}$. Hence, for the specific choice of BLS12-381 this would imply $\alpha = 5$. Vision, on the other hand, is specified over a binary field and can thus also not be directly implemented in $\mathbb{F}_q$ arithmetic.

Additionally, the signature scheme Picnic [CDG+17] demonstrated that LowMC [ARS+15], initially designed for the application in secure multiparty computation and fully homomorphic encryption, performs well enough in the context of NIZKs. We consider LowMC in our evaluation as the conservative choice of SNARK-friendly primitives, since it has seen some rounds of cryptanalysis [DEM16, DLMW15] and corresponding updates to the round formula [RST18], and additionally gained attention in terms of efficient implementations [DKP+19].

Evaluation. In Table 1 we evaluate a variety of SNARK-friendly primitives together with the SHA2 and SHA3 families of hash functions. Our evaluation focuses on the provable secure version using fixed-value-key-binding PRFs as discussed above with a PRF having 256 bit keys, inputs and outputs. The number of constraints are computed according to the formulas given in the respective works. We consider MiMC-(N, R), GMiMC-(N, t, R) with the expanding round function (ERF) construction, Poseidon-(N, t, R_f, R_p) with $x \mapsto x^5$ as SBox, Rescue-(N, t, R) with $x \mapsto x^5$ and $x \mapsto x^{1/5}$, Vision-(N, t, R), and LowMC-(N, k, m, R), where N denotes the block size, t the number of branches, R the number of rounds, R_f and R_p the number of full and partial rounds, k the key size and m the number of Sboxes.

[11] https://starkware.co/hash-challenge/

[12] https://electriccoin.co/blog/new-snark-curve/

Table 1. Number of constraints required for C∅C∅ and OC∅C∅.

Framework	Symmetric primitive	PRF / Commitment	Provably secure	# of constraints PRF / Com.	$\sum$
C∅C∅	SHA256	HMAC PRF + hash com.	✗	111360 + 44544	244992
OC∅C∅	SHA256	HMAC PRF	✓	111360	222720
		TLS 1.2 PRF	✓	230400	460800
	SHAKE256	Sponge PRF	✓	38400	76800
	MiMC-$(1025, 646)$	Sponge PRF	✓	646	1292
	GMiMC-$(1024, 4, 332)$	Sponge PRF	✓	999	1998
	Poseidon-$(1536, 2, 10, 114)$	Sponge PRF	✓	402	804
	Vision-$(1778, 14, 10)$	Sponge PRF	✓	1400	2800
	Rescue-$(1750, 14, 10)$	Sponge PRF	✓	840	1680
	LowMC-$(256, 256, 1, 537)$	feed-forward PRF	✓	1074	2148
	LowMC-$(1024, 256, 1, 1027)$	Sponge PRF	✓	2144	4288

Where possible, we selected instances compatible with the field induced by BLS12-381, i.e., for Poseidon and Rescue. The table also provides various different PRF constructions. Where possible, we use a Sponge-based approach [BDPV08] akin to SHAKE256. For LowMC, we also consider a feed-forward PRF built as $f_s(x) = \mathcal{E}(s, x) \oplus x$ where $\mathcal{E}$ denotes the encryption of a block. In the case of SHA256, we consider three variants that can partly also be observed in practice – directly using the HMAC output as PRF and the one from TLS 1.2 [DR08]. Regardless of the concrete choice, even the rather expensive SHAKE256 PRF is a better choice than any of the SHA256-based ones.

We stress that the numbers in Table 1 should be treated as lower bounds. One the one hand, as the security analysis of these primitives evolves, the rather aggressive choice of round numbers may need to be increased. Considering that the STARK-friendly hash challenge was almost immediately solved for the low security instances of MiMC, GMiMC and Poseidon, we expect those numbers to grow. On the other hand, for some of the instantiations it might not be immediately clear if they actually provide the fixed-value-key-binding property. For a very conservative instantiation, one could fallback to the tree-based approach by Fischlin [Fis99], which would be even more expensive, since then every PRF evaluation would internally call the PRF multiple times.

Other Important Remarks. Furthermore, beside more efficient instantiations than within the original CØCØ framework, the approach based on fixed-value-key-binding PRFs also circumvents another issue in concrete instantiations. Hash commitments can only be proven secure in the ROM, which would require to prove preimages of a random oracle. Hence, the construction is impossible to properly instantiate with provable security guarantees. In any case, the choice of a commitment based on symmetric primitive comes with other drawbacks as well. Since such a commitment lacks any useful algebraic structure, it is not obvious how to obtain SE updatable SNARKs.

Regarding the choice of strongly unforgeable one-time signature schemes, Groth's sOTS (as discussed in [ARS20]) or Boneh-Boyen signatures [BB04] (as proposed in other instantiations of CØCØ [AB19, Bag19]) would be natural choices especially when considering the underlying SNARKs already rely on discrete logarithm assumptions (in bilinear groups). Alternatively, any strong EUF-CMA secure signature such as EC-Schnorr would fit as well. We note however, while this choice would avoid the need for a pairing evaluation for signature verification (in the case of Boneh-Boyen) and the proof overhead would be slightly smaller, EC-Schnorr provides the necessary security guarantees only in the ROM.[13]

[13] In private communication, A. Kosba confirmed that their implementation used a non-malleable variant of ECDSA for benchmarking. To the best of our knowledge, this variant is only suspected to be strongly unforgeable without proof so far. Thus we consider EC-Schnorr as candidate. The performance and overhead is expected to be the same.

Putting everything together, instantiating the C∅C∅ or OC∅C∅ framework with concrete symmetric primitives is non-trivial and comes with some limitations. Subsequently, we will propose an alternative framework Lamassu, which comes with the same cost as the most aggressive choice of symmetric-key primitive and in contrast to C∅C∅ also provides SE updatable SNARKs.

3.3 The Lamassu Framework

Now, we present the Lamassu framework, which builds upon the recent compiler to obtain SE-NIZK proposed in [DS19]. However, we want to stress that we cannot directly use their compiler in order to construct SE updatable SNARKs and this requires non-trivial changes. The ingredients of their construction is to use a combination of an EUF-CMA secure adaptable key-homomorphic signature scheme Σ (EC-Schnorr or ECDSA are good candidate for pairing based SNARKs) and a strongly unforgeable one-time signature (sOTS) scheme Σ_{OT} (Groth's sOTS under the discrete logarithm assumption is a good candidate) to add the required non-malleability guarantees to the underlying knowledge sound NIZK proof system Π together with the folklore OR-trick to add simulation soundness. The distinguishing feature of this transformation is that in the proof computation one computes a signature to certify a public key of OTS using freshly sampled signing key sk of Σ in plain and thus does not need to encrypt a signature and prove that it verifies with a verification key in the CRS (e.g., as done in [Gro06]). Consequently, in the OR part of the proof one just needs to prove that one knows the shift csk (which is the trapdoor of the CRS) to adapt signatures from pk to ones valid under verification key cpk in the CRS. As it turns out, this feature lays the foundation for being able to support updatability. Now, given any language $\mathcal{L}$ with NP relation $\mathcal{R}_{\mathcal{L}}$, the language obtained via the compiler is $\mathcal{L}'$ s.t. $\{(\mathtt{x}, \mathsf{cpk}, \mathsf{pk}), (\mathtt{w}, \mathsf{csk} - \mathsf{sk})\} \in \mathcal{R}_{\mathcal{L}'}$ iff:

$$\{(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}} \vee \mathsf{cpk} = \mathsf{pk} \cdot \mu(\mathsf{csk} - \mathsf{sk})\}\,.$$

More precisely, in every proof computation one uses Σ to "certify" the public key of a newly generated key pair of Σ_{OT}. The secret key of Σ_{OT} is then used to sign the parts of the proof which must be non-malleable. Adaptability of Σ makes it possible to also use newly generated keys of Σ upon each proof computation. In particular, the relation associated to $\mathcal{L}'$ is designed so that the additional clause introduced via the OR-trick is the "shift amount" required to shift such signatures to signatures under a key cpk of Σ in the CRS. A proof for $\mathtt{x} \in \mathcal{L}$ is easy to compute when given $\mathtt{w}$ such that $(\mathtt{x}, \mathtt{w}) \in \mathcal{L}_{\pi}$. One does not need a satisfying assignment for the second clause in the OR statement, and can thus compute all signatures under newly generated keys. To simulate proofs, however, one can set up CRS in a way that we know csk corresponding to cpk, compute the "shift amount" and use it as a satisfying witness for the other clause in the OR statement. We recall the construction in Fig. 2 and for completeness recall the Theorem given in [DS19] below.[14] We note that for non black-box extraction as it is the case with SNARKs, the trapdoor $\mathtt{tc}_{\mathsf{ext}} = \bot$ and one simply uses the non black-box extractor of the underlying SNARK.

Theorem 2 ([DS19]). *Let Π be a complete, witness indistinguishable non-interactive argument of knowledge system for the language $\mathcal{L}$, let Σ be an EUF-CMA secure signature scheme that adapts signatures, and let Σ_{OT} be a strongly unforgeable one-time signature scheme, then the argument system Π' is a complete and strong simulation extractable argument system for language $\mathcal{L}'$.*

Note that the theorem clearly applies to any proof system that is zero-knowledge, as this implies the weaker notion of witness-indistinguishability.

Applying [DS19] to NIZKs without knowledge soundness. We now argue that, although we do not require it in context of SNARKs, analogous to the folklore compiler used in [KZM+15], we can also start form any NIZK that is only sound instead of knowledge sound. Then, using the compiler in [DS19] we still can obtain SE-NIZK when starting from any conventional NIZK. More precisely, the by now folklore compiler [DP92] to obtain knowledge soundness for any sound NIZK is to put a public key $\mathsf{pk_e}$ of any perfectly correct IND-CPA secure public key encryption scheme into the CRS, where the extraction trapdoor $\mathtt{tc}_{\mathsf{Ext}}$ is the corresponding secret key, and extend the language such that it contains an encryption of the witness of the original language. We will capture this in the following corollary, where starting from a NIZK for $\mathcal{L}$ with NP relation $\mathcal{R}_{\mathcal{L}}$, we obtain a knowledge sound NIZK by extending the language to $\mathcal{L}'$ such that $\{(\mathtt{x}, c), (\mathtt{w}, \omega)\} \in \mathcal{R}_{\mathcal{L}'}$ iff:

$$\{(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}} \wedge c = \mathsf{Enc}(\mathsf{pk_e}, \mathtt{w}; \omega)\}.$$

Corollary 2. *Let NIZK for language $\mathcal{L}$ be complete, sound and zero-knowledge, the public key encryption scheme be perfectly correct and IND-CPA secure, then the NIZK for language $\mathcal{L}'$ is complete, knowledge-sound and zero-knowledge.*

The proof exactly follows the argumentation in [KZM+15] and is thus omitted. We stress that if one bases the compiler of [DS19] on a NIZK that is based on standard or falsifiable assumptions that is only sound, then one requires this additional encryption of the witness $\mathtt{w}$. However, when one relies on knowledge assumptions, as it is the case within SNARKs, then one does not need the language extension in Corollary 2 and can simply use the non black-box extractor of the underlying SNARK.

[14] We note that what is called simulation sound extractable in [DS19] is called strong simulation extractable in this paper in order to be aligned with the notation used in the CØCØ framework.

$\mathsf{KGen_{crs}}(1^\lambda)$

- $(\mathtt{crs}_\Pi, \mathtt{tc}_\Pi, \mathtt{tc}_{\mathsf{ext}}) \leftarrow \Pi.\mathsf{KGen}(1^\lambda)$;
- $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \Sigma.\mathsf{KGen}(1^\kappa)$;
- $\mathtt{crs} := (\mathtt{crs}_\Pi, \mathsf{cpk}), \mathtt{tc} := (\mathtt{tc}_\Pi, \mathsf{csk})$; **return** $\mathtt{crs}$.

$\mathsf{P}(\mathtt{crs}, \mathtt{x}, \mathtt{w})$

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KGen}(1^\kappa)$;
- $(\mathsf{sk_{OT}}, \mathsf{pk_{OT}}) \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{KGen}(1^\kappa)$;
- $\pi_\Pi \leftarrow \Pi.\mathsf{P}(\mathtt{crs}, \mathtt{x}, (\mathtt{w}, \bot)); \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk_{OT}}))$;
- $\sigma_{\mathsf{OT}} \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{Sign}(\mathsf{sk_{OT}}, \pi_\Pi || \mathtt{x} || \mathsf{pk} || \sigma)$;

return $\pi := (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk_{OT}}, \sigma_{\mathsf{OT}})$.

$\mathsf{V}(\mathtt{crs}, \mathtt{x}, \pi)$

- Parse π as $(\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk_{OT}}, \sigma_{\mathsf{OT}})$;
- **if** $\Pi.\mathsf{V}(\mathtt{crs}_\Pi, \mathtt{x}, \pi_\Pi) = 0$

$\vee\ \Sigma.\mathsf{Verify}(\mathsf{pk}, \mathsf{pk_{OT}}, \sigma) = 0$

$\vee\ \Sigma_{\mathsf{OT}}.\mathsf{Verify}(\mathsf{pk_{OT}}, \pi_\Pi || \mathtt{x} || \mathsf{pk} || \sigma, \sigma_{\mathsf{OT}}) = 0$ **then return** 0;

else return 1.

$\mathsf{Sim}(\mathtt{crs}, \mathtt{x}, \mathtt{tc})$

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KGen}(1^\kappa)$; $(\mathsf{sk_{OT}}, \mathsf{pk_{OT}}) \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{KGen}(1^\kappa)$;
- $\pi_\Pi \leftarrow \Pi.\mathsf{P}(\mathtt{crs}, \mathtt{x}, (\bot, \mathsf{csk} - \mathsf{sk})); \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk_{OT}})$;
- $\sigma_{\mathsf{OT}} \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{Sign}(\mathsf{sk_{OT}}, \pi_\Pi || \mathtt{x} || \mathsf{pk} || \sigma)$;
- **return** $\pi := (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk_{OT}}, \sigma_{\mathsf{OT}})$.

$\mathsf{Ext}(\mathtt{crs}, \mathtt{x}, \pi, \mathtt{tc}_{\mathsf{ext}})$

- $(\mathtt{w}, \bot) \leftarrow \Pi.\mathsf{Ext}(\mathtt{crs}, \mathtt{x}, \pi, \mathtt{tc}_{\mathsf{ext}})$; **return** $\mathtt{w}$.

Fig. 2. The generic SE-NIZK compiler from [DS19].

Updatable SNARKs from LAMASSU. In the following we present the generic construction using the definitional framework in [GKM+18] for updatable SNARKs and refer to [ARS20] for subversion version of the compiler. Roughly speaking, in the updatable CRS definition, Groth et. al relaxed the CRS model by allowing the adversary to either fully generate the CRS itself, or at least contribute to its computation as one of the parties performing updates. In other words, we can think of this as having the adversary interact with the $\mathsf{KGen_{crs}}$ algorithm. An updatable SNARK has the following additional PPT algorithms on top of $(\mathsf{KGen_{crs}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$. After running $(\mathtt{crs}, \mathtt{tc}, \zeta) \leftarrow \mathsf{KGen_{crs}}$, where ζ is a proof of correctness of $\mathtt{crs}$.

$\mathsf{Ucrs}(1^\lambda, \mathtt{crs}, \{\zeta_i\}_{i=1}^{i=n})$**.** Takes as input the security parameter λ, a CRS $\mathtt{crs}$, and a list of update proofs for the CRS. It outputs an updated CRS $\mathtt{crs_{up}}$ and a proof ζ_{up} of the correctness of the update.

$\mathsf{Vcrs}(1^\lambda, \texttt{crs}, \{\zeta_i\}_{i=1}^{i=n})$. Given the security parameter λ, a CRS $\texttt{crs}$, and a list of proofs ζ_i. It outputs a bit indicating accept ($b = 1$), or reject ($b = 0$).

The standard trusted setup can be considered as an updatable setup with $\zeta = \epsilon$ as the update proof and where the verification algorithm accepts anything. For a subversion resistant setup (Sub-zk-SNARKs), the proof ζ could be added as extra elements into the CRS solely to make the CRS verifiable.

We present the full construction of SE updatable SNARKs in Fig. 3. Notice that in the Fig. 3, the subverter Z could be either the algorithms ($\Pi.\mathsf{KGen}$, $\Sigma.\mathsf{KGen}$) or the updater Ucrs.

Theorem 3 ([ARS20]). *Let the underlying updatable SNARK scheme satisfy perfect completeness, updatable zero-knowledge, and updatable knowledge soundness. Let Σ be an EUF-CMA secure adaptable and updatable signature scheme and Σ_{OT} is a strongly unforgeable one-time signature scheme. Then, the SE updatable SNARKs argument system from Fig. 3, is (i) perfectly complete, (ii) updatable zero-knowledge, and (iii) updatable strong simulation extractable.*

Instantiation. As an example instantiation, by taking updatable Schnorr signatures as presented in Section 2.5, using the Lamassu framework we can now obtain an SE updatable SNARK by lifting the updatable SNARK in [GKM+18]. This, for instance, results in an overhead of $1\mathbb{G}_1 + 1\mathbb{G}_2$ elements in the CRS and $2\mathbb{G}_1 + 2\mathbb{G}_2 + 2\mathbb{Z}_q$ elements in the proofs (cf. Table 2 for a comparison of different instantiations and existing ad-hoc approaches).

4 Evaluation

For the evaluation of OC∅C∅ and Lamassu, we focus on SNARKs built from the pairing-friendly elliptic curve BLS12-381. In this case, we can leverage the Jubjub curve [HBHW19] used by Zcash for fast elliptic-curve arithmetic in the circuit. The Jubjub curve is a twisted Edwards curve defined over $\mathbb{F}_r$ with r being the prime order of BLS12-381. Twisted Edwards curves enjoy complete addition laws and they naturally fit the requirements of Schnorr signatures.

$\mathsf{KGen}_{\mathsf{crs}}(1^\lambda)$

- $(\mathtt{crs}_\Pi, \mathtt{tc}_\Pi, \zeta_\Pi) \leftarrow \Pi.\mathsf{KGen}(1^\lambda)$;
- $(\mathsf{csk}, \mathsf{cpk}, \zeta_{\mathsf{cpk}}) \leftarrow \Sigma.\mathsf{KGen}(1^\kappa)$;
- $\mathtt{crs} := (\mathtt{crs}_\Pi, \mathsf{cpk}), \mathtt{tc} := (\mathtt{tc}_\Pi, \mathsf{csk});$ **return** $\mathtt{crs}$.

$\mathsf{Ucrs}(1^\lambda, \mathtt{crs}, \{\zeta_i\}_{i=1}^{i=n})$

- $(\mathtt{crs}_{\Pi,\mathsf{up}}, \zeta_{\Pi,\mathsf{up}}) \leftarrow \Pi.\mathsf{Ucrs}(1^\lambda, \mathtt{crs}_\Pi, \{\zeta_{\Pi,i}\}_{i=1}^{i=n})$;
- $(\mathsf{cpk}_{\mathsf{up}}, \zeta_{\mathsf{cpk},\mathsf{up}}) \leftarrow \Sigma.\mathsf{Ucrs}(\mathsf{cpk}, \{\zeta_{\mathsf{cpk},i}\}_{i=1}^{i=n})$;
- **return** $(\mathtt{crs}_{\mathsf{up}} = (\mathtt{crs}_{\Pi,\mathsf{up}}, \mathsf{cpk}_{\mathsf{up}}), \zeta_{\mathsf{up}} = (\zeta_{\Pi,\mathsf{up}}, \zeta_{\mathsf{cpk},\mathsf{up}}))$

$\mathsf{Vcrs}(1^\lambda, \mathtt{crs}, \{\zeta_i\}_{i=1}^{i=n})$

- Parse ζ_i as $(\zeta_{\Pi,i}, \zeta_{\mathsf{cpk},i})$;
- **if** $\mathsf{Vcrs}_\Pi(1^\lambda, \mathtt{crs}, \{\zeta_{\Pi,i}\}_{i=1}^{i=n}) = 1 \wedge$
- $\Sigma.\mathsf{Vpk}(\mathsf{pk}, \mathsf{cpk}, \{\zeta_{\mathsf{cpk},i}\}_{i=1}^{i=n})) = 1$

then return 1; **else return** 0.

$\mathsf{P}(\mathtt{crs}_{\mathsf{up}}, \mathtt{x}, \mathtt{w})$

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KGen}(1^\kappa)$;
- $(\mathsf{sk}_{\mathsf{OT}}, \mathsf{pk}_{\mathsf{OT}}) \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{KGen}(1^\kappa)$;
- $\pi_\Pi \leftarrow \Pi.\mathsf{P}(\mathtt{crs}_{\mathsf{up}}, \mathtt{x}, (\mathtt{w}, \perp), \perp); \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{OT}})$;
- $\sigma_{\mathsf{OT}} \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{OT}}, \pi_\Pi || \mathtt{x} || \mathsf{pk} || \sigma)$;

return $\pi := (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{OT}}, \sigma_{\mathsf{OT}})$.

$\mathsf{V}(\mathtt{crs}_{\mathsf{up}}, \mathtt{x}, \pi)$

- Parse π as $(\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{OT}}, \sigma_{\mathsf{OT}})$;
- **if** $\Pi.\mathsf{V}(\mathtt{crs}_{\Pi,\mathsf{up}}, \mathtt{x}, \pi_\Pi) = 0 \vee \Sigma.\mathsf{Verify}(\mathsf{pk}, \mathsf{pk}_{\mathsf{OT}}, \sigma) = 0$
$\vee\ \Sigma_{\mathsf{OT}}.\mathsf{Verify}(\mathsf{pk}_{\mathsf{OT}}, \pi_\Pi || \mathtt{x} || \mathsf{pk} || \sigma, \sigma_{\mathsf{OT}}) = 0$ **then return** 0;
else return 1.

$\mathsf{Sim}(\mathtt{crs}_{\mathsf{up}}, \mathtt{x}, \mathtt{tc})$

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KGen}(1^\kappa)$; $(\mathsf{sk}_{\mathsf{OT}}, \mathsf{pk}_{\mathsf{OT}}) \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{KGen}(1^\kappa)$;
- $\pi_{\mathsf{Sim}} \leftarrow \Pi.\mathsf{Sim}(\mathtt{crs}_{\Pi,\mathsf{up}}, \mathtt{x}, (\perp, \mathtt{tc}_\Pi), \perp)$;
- $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{OT}})$;
- $\sigma_{\mathsf{OT}} \leftarrow \Sigma_{\mathsf{OT}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{OT}}, \pi_{\mathsf{Sim}} || \mathtt{x} || \mathsf{pk} || \sigma)$;

return $\pi := (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{OT}}, \sigma_{\mathsf{OT}})$.

$\mathsf{Ext}_{\mathsf{Z}}(1^\lambda, \mathtt{crs}, \mathtt{crs}_{\mathsf{up}}, \omega_{\mathsf{Z}})$

- $\mathtt{tc} \leftarrow \Pi.\mathsf{Ext}(1^\lambda, \mathtt{crs}, \mathtt{crs}_{\mathsf{up}}, \omega_{\mathsf{Z}});$ **return** $\mathtt{tc}$.

Fig. 3. The SE updatable SNARKs from LAMASSU.

Table 2. Comparison of SE-SNARKs. The given sizes for the CRS and proofs as well as the number of operations are overheads compared to the underlying SNARKs. For ad-hoc constructions the overhead is relative to Groth's SNARK. n denotes the number of multiplication gates.

	Features			Overhead				
	generic	subversion	updatable	`crs`	bits	π	bits	V
C∅C∅ [KZM+15]‡	✓	✓*	✗	1λ	256	$1\mathbb{G}, 2\mathbb{Z}_q, 1\lambda$	1016	$2\mathsf{E}_{\mathbb{G}}$
OC∅C∅[S]	✓	✓*	✗	2λ	512	$1\mathbb{G}, 2\mathbb{Z}_q, 1\lambda$	1016	$2\mathsf{E}_{\mathbb{G}}$
OC∅C∅[G]	✓	✓*	✗	2λ	512	$3\mathbb{G}, 2\mathbb{Z}_q, 1\lambda$	1528	$3\mathsf{E}_{\mathbb{G}}$
Lamassu[S,S]	✓	✓	✗	$1\mathbb{G}$	256	$2\mathbb{G}, 4\mathbb{Z}_q$	1520	$4\mathsf{E}_{\mathbb{G}}$
Lamassu[S,S]	✓	✓	✓	$1\mathbb{G}_1, 1\mathbb{G}_2$	1145	$1\mathbb{G}_1, 1\mathbb{G}_2, 1\mathbb{G}, 4\mathbb{Z}_q$	2415	$2\mathsf{E}_{\mathbb{G}_1}, 2\mathsf{E}_{\mathbb{G}}$
Lamassu[S,G]	✓	✓	✗	$1\mathbb{G}$	256	$4\mathbb{G}, 4\mathbb{Z}_q$	2032	$5\mathsf{E}_{\mathbb{G}}$
Lamassu[S,G]	✓	✓	✓	$1\mathbb{G}_1, 1\mathbb{G}_2$	1145	$1\mathbb{G}_1, 1\mathbb{G}_2, 3\mathbb{G}, 4\mathbb{Z}_q$	2927	$2\mathsf{E}_{\mathbb{G}_1}, 3\mathsf{E}_{\mathbb{G}}$
Lamassu[S,BB]	✓	✓	✗	$1\mathbb{G}$	256	$1\mathbb{G}_1, 1\mathbb{G}_2, 1\mathbb{G}, 2\mathbb{Z}_q$	1905	$2\mathsf{E}_{\mathbb{G}}, 1\mathsf{P}$
Lamassu[S,BB]	✓	✓	✓	$1\mathbb{G}_1, 1\mathbb{G}_2$	1145	$2\mathbb{G}_1, 2\mathbb{G}_2, 2\mathbb{Z}_q$	2800	$2\mathsf{E}_{\mathbb{G}_1}, 1\mathsf{P}$
Groth-Maller [GM17]	✗	✗	✗	$(2n+5)\mathbb{G}_1, n\mathbb{G}_2$	$1910 + 1527n$	–	0	2P
Bowe-Gabizon [BG18]	✗	✗	✗	–†	0	$1\mathbb{G}_1, 1\mathbb{G}_2$	1146	2P
Lipmaa ($\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$) [Lip19]	✗	✓	✗	$n\mathbb{G}_1$	$382n$	$1\mathbb{G}_1$	382	2P
Kim-Lee-Oh [KLO19]	✗	✓	✓	$n\mathbb{G}_1$	$382n$	–	0	–
Atapoor-Baghery [AB19]‡	✗	✗	✗	1λ	256	$1\mathbb{G}_1, 1\mathbb{G}_2, 1\lambda$	1401	1P
Baghery [Bag19]‡	✗	✓	✗	1λ	256	$1\mathbb{G}_1, 1\mathbb{G}_2, 1\lambda$	1401	1P

‡ Proves statements with respect to the evaluation of a random oracle (cf. Section 3.2).
† Achieves no `crs` overhead by additionally requiring random oracles.
* With the non-black box extractor, C∅C∅ retains the subversion resistance of the underlying SNARK [Bag19].

The Sapling protocol uses the Jubjub curve to prove relations of the form $\mathsf{rk} = \mathsf{ak} \cdot g^{\alpha}$ and checking that α is in the correct range for the witness α. The first part of the relation can be expressed with 756 constraints, whereas the latter can be expressed with 252 constraints, so a total of 1008 constraints [HBHW19, Section A.4]. For LAMASSU, we extend the relation with a proof of the statement $\mathsf{cpk} = \mathsf{pk} \cdot \mu(\mathsf{csk} - \mathsf{sk})$ with the witness $\mathsf{csk} - \mathsf{sk}$. For Schnorr signatures (cf. Appendix A.4), but also other DLOG-based signature schemes such as Schnorr, the public key is a group element of the form g^{sk} and similarly μ simply maps scalars to the corresponding group element, i.e., $\mu(x) = g^x$. Hence, the circuit for this relation also requires 1008 constraints. Compared to the OC∅C∅ framework instantiations (cf. Table 1), LAMASSU needs only 200 constraints more than the most aggressive choice using Poseidon and beats all others in the number of constraints. Considering that Schnorr signatures are well established primitives, and that the confidence in their security is far bigger than all the recent SNARK/STARK-friendly primitives, this additional confidence and the updatability feature come at a very small cost for the prover.

In terms of bandwidth overhead, we only need to compare the overhead induced by $\mathsf{cpk} = \mathsf{pk} \cdot \mu(\mathsf{csk} - \mathsf{sk})$ together with the signature and one-time signature in LAMASSU, and $\mu = f_{s_0}(\mathsf{pk}_s) \wedge \rho = f_{s_0}(\beta_0)$ and the one-time signature in the case of OC∅C∅. We start with LAMASSU. The CRS is extended with a public key cpk of signature scheme Σ, i.e., when using Schnorr (or ECDSA) a point on the Jubjub curve which requires 510 bits without or 256 bits with point compression. For each proof, new Σ and Σ_{OT} keys are sampled. The proof then includes a Σ public key and signature, as well as as Σ_{OT} public key and signature. The former amounts to 256 bits for the public key and 504 bits for the signature (2 integers modulo the group order), and the latter – when instantiated as Groth's sOTS over Jubjub (or a curve of similar size) – amounts to 768 bits for the public key (3 group elements) and 504 bits for the signature (2 integers modulo the group order). In total, the size of the proof is increased by 2032 bits. The updatable version is similar, but Schnorr is performed in $\mathbb{G}_1$ with additional public key and update in $\mathbb{G}_2$.

For C∅C∅, the CRS is extended with a SHA256 commitment. The proofs are extended with a freshly generated Σ_{OT} public key and a signature together with the evaluation of a PRF also instantiated with SHA256. Hence, the CRS grows by 256 bits and each proof grows by 1016 bits assuming the use of Schnorr over Jubjub. For OC∅C∅, the CRS is extended with ρ and β, both 256 bits each. Each proof additionally contains μ as well as freshly generated Σ_{OT} public key and signature. Using Groth's sOTS, the proof grows by 1528 bits in total.

In Table 2 we present a comparison of SE-SNARKs including OC∅C∅ using Groth's OTS, OC∅C∅[G], and Schnorr, OC∅C∅[S], LAMASSU using Schnorr, LAMASSU[S,S], Groth's OTS, LAMASSU[S,G], and Boneh-Boyen signatures [BB04], LAMASSU[S,BB], both as non-updatable and updatable variant. The overhead is relative to the underlying SNARK (for the generic constructions) or the SNARK they are based on, e.g., relative to [Gro16]. In the table, n denotes the number of multiplication gates in the circuit, $\mathbb{G}_1$ and $\mathbb{G}_2$ the two source groups of a bilinear group, $\mathbb{G}$ a group with prime order q, and λ the sizes of commitments and PRF evaluations. For concrete numbers, we followed the above choice of curves, namely Jubjub ($\mathbb{G}$) and BLS12-381 as bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g})$, respectively. For commitments and PRF images, we assume that they are 256 bits. For the verifier overhead, we consider the most expensive operations. $\mathsf{E}_{\mathbb{G}}$ denotes an exponentiation in $\mathbb{G}$ and P a pairing evaluation. Thereby, P is a factor 10 slower than $\mathsf{E}_{\mathbb{G}}$.

Compared to the ad-hoc constructions, the generic frameworks C∅C∅, OC∅C∅, and LAMASSU offer a trade-off between the size of the CRS, proof sizes and verifier overhead. Especially when comparing to Kim-Lee-Oh [KLO19], which only extends the CRS, this trade-off becomes apparent. When comparing to the others, the verifier overhead is smaller than the ones observed for Groth-Maller [GM17], Bowe-Gabizon [BG18] and Lipmaa [Lip19] and is comparable to the constructions of Atappoor-Baghery [AB19] and Baghery [Bag19], yet LAMASSU offers more features.

Acknowledgements. We thank Ahmed Kosba for valuable feedback on the choice of the one-time signature scheme in the C∅C∅ framework. We also like to thank the anonymous reviewers. This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreements n°830929 (CyberSec4Europe) and n°871473 (KRAKEN), by the Austrian Science Fund (FWF) and netidee SCIENCE under grant agreement P31621-N38 (PROFET) and the Estonian Research Council grant PRG49.

References

AABS+19. Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. Cryptology ePrint Archive, Report 2019/426, 2019. https://eprint.iacr.org/2019/426.

AB19. Shahla Atapoor and Karim Baghery. Simulation extractability in groth's zk-snark. In Cristina Pérez-Solà, Guillermo Navarro-Arribas, Alex Biryukov, and Joaquín García-Alfaro, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26-27, 2019, Proceedings*, volume 11737 of *LNCS*, pages 336–354. Springer, 2019.

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

ACG+19. Martin R. Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. Algebraic cryptanalysis of STARK-friendly designs: Application to MARVELlous and MiMC. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 371–397. Springer, Heidelberg, December 2019.

ACJT00. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270. Springer, Heidelberg, August 2000.

AD18. Tomer Ashur and Siemen Dhooghe. Marvellous: a stark-friendly family of cryptographic primitives. Cryptology ePrint Archive, Report 2018/1098, 2018. https://eprint.iacr.org/2018/1098.

AGP+19. Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel structures for MPC, and more. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *ESORICS 2019, Part II*, volume 11736 of *LNCS*, pages 151–171. Springer, Heidelberg, September 2019.

AGR+16. Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 191–219. Springer, Heidelberg, December 2016.

AMV15. Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signature schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 364–375. ACM Press, October 2015.

ARS+15. Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Heidelberg, April 2015.

ARS20. Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable snarks generically. *IACR Cryptology ePrint Archive*, 2020:62, 2020.

Bag19. 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.

BB04. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Heidelberg, May 2004.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BBC+18. Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 669–699. Springer, Heidelberg, August 2018.

BBHR19. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732. Springer, Heidelberg, August 2019.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

BBUV19. Ward Beullens, Tim Beyne, Aleksei Udovenko, and Giuseppe Vitto. Cryptanalysis of the legendre prf and generalizations. Cryptology ePrint Archive, Report 2019/1357, 2019. https://eprint.iacr.org/2019/1357.

BCC04. Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 132–145. ACM Press, October 2004.

BCC$^+$09. Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2009.

BCC$^+$16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 117–136. Springer, Heidelberg, June 2016.

BCG$^+$13. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013.

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.

BCG$^+$15. Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE Computer Society Press, May 2015.

BCG$^+$18. Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 595–626. Springer, Heidelberg, December 2018.

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.

BDD$^+$15. Achiya Bar-On, Itai Dinur, Orr Dunkelman, Virginie Lallemand, Nathan Keller, and Boaz Tsaban. Cryptanalysis of SP networks with partial non-linear layers. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 315–342. Springer, Heidelberg, April 2015.

BDPV08. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Heidelberg, April 2008.

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.

BFS16. Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016.

BG90. Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211. Springer, Heidelberg, August 1990.

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.

BGG19. Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *FC 2018 Workshops*, volume 10958 of *LNCS*, pages 64–77. Springer, Heidelberg, March 2019.

BL09. Ernie Brickell and Jiangtao Li. Enhanced privacy ID from bilinear pairing. Cryptology ePrint Archive, Report 2009/095, 2009. http://eprint.iacr.org/2009/095.

Bon19. Xavier Bonnetain. Collisions on feistel-mimc and univariate gmimc. Cryptology ePrint Archive, Report 2019/951, 2019. https://eprint.iacr.org/2019/951.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

CCD+17. Jan Camenisch, Liqun Chen, Manu Drijvers, Anja Lehmann, David Novick, and Rainer Urian. One TPM to bind them all: Fixing TPM 2.0 for provably secure anonymous attestation. In *2017 IEEE Symposium on Security and Privacy*, pages 901–920. IEEE Computer Society Press, May 2017.

CDD17. Jan Camenisch, Manu Drijvers, and Maria Dubovitskaya. Practical UC-secure delegatable credentials with attributes and their application to blockchain. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 683–699. ACM Press, October / November 2017.

CDG+17. Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1825–1842. ACM Press, October / November 2017.

CFQ19. Matteo Campanelli, Dario Fiore, and AnaÃŕs Querol. Legosnark: Modular design and composition of succinct zero-knowledge proofs. Cryptology ePrint Archive, Report 2019/142, 2019. https://eprint.iacr.org/2019/142, to appear at CCS'19.

CGGN17. Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 229–243. ACM Press, October / November 2017.

CGL+17. Alessandro Chiesa, Matthew Green, Jingcheng Liu, Peihan Miao, Ian Miers, and Pratyush Mishra. Decentralized anonymous micropayments. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 609–642. Springer, Heidelberg, April / May 2017.

Cha86. David Chaum. Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In Franz Pichler, editor, *EUROCRYPT'85*, volume 219 of *LNCS*, pages 241–244. Springer, Heidelberg, April 1986.

CHM+19. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. Cryptology ePrint Archive, Report 2019/1047, 2019. https://eprint.iacr.org/2019/1047.

CKL+16. Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. Formal treatment of privacy-enhancing credential systems. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015*, volume 9566 of *LNCS*, pages 3–24. Springer, Heidelberg, August 2016.

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001.

CL03. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Heidelberg, September 2003.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

CMR98. Ran Canetti, Daniele Micciancio, and Omer Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *30th ACM STOC*, pages 131–140. ACM Press, May 1998.

Cv91. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991.

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.

DEM16. Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Higher-order cryptanalysis of LowMC. In Soonhak Kwon and Aaram Yun, editors, *ICISC 15*, volume 9558 of *LNCS*, pages 87–101. Springer, Heidelberg, November 2016.

DFGK14. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014.

DGS03. Ivan Damgård, Jens Groth, and Gorm Salomonsen. The theory and implementation of an electronic voting system. In Dimitris Gritzalis, editor, *Secure Electronic Voting*, volume 7 of *Advances in Information Security*, pages 77–98. Springer, 2003.

DKP+19. Itai Dinur, Daniel Kales, Angela Promitzer, Sebastian Ramacher, and Christian Rechberger. Linear equivalence of block ciphers with partial non-linear layers: Application to LowMC. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 343–372. Springer, Heidelberg, May 2019.

DLMW15. Itai Dinur, Yunwen Liu, Willi Meier, and Qingju Wang. Optimized interpolation attacks on LowMC. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 535–560. Springer, Heidelberg, November / December 2015.

DOR+16. David Derler, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, and Daniel Slamanig. Digital signatures from symmetric-key primitives. Cryptology ePrint Archive, Report 2016/1085, 2016. http://eprint.iacr.org/2016/1085.

DP92. Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *33rd FOCS*, pages 427–436. IEEE Computer Society Press, October 1992.

DP06. Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06*, volume 4341 of *LNCS*, pages 193–210. Springer, Heidelberg, September 2006.

DR08. Tim Dierks and Eric Rescorla. The transport layer security (TLS) protocol version 1.2. *RFC*, 5246:1–104, 2008.

DRS18. David Derler, Sebastian Ramacher, and Daniel Slamanig. Generic double-authentication preventing signatures and a post-quantum instantiation. In Joonsang Baek, Willy Susilo, and Jongkil Kim, editors, *ProvSec 2018*, volume 11192 of *LNCS*, pages 258–276. Springer, Heidelberg, October 2018.

DS18. David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *ASIACCS 18*, pages 551–565. ACM Press, April 2018.

DS19. David Derler and Daniel Slamanig. Key-homomorphic signatures: definitions and applications to multiparty signatures and non-interactive zero-knowledge. *Des. Codes Cryptogr.*, 87(6):1373–1413, 2019.

FHS19. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

Fis99. Marc Fischlin. Pseudorandom function tribe ensembles based on one-way permutations: Improvements and applications. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 432–445. Springer, Heidelberg, May 1999.

FMMO19. Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 649–678. Springer, Heidelberg, December 2019.

FPS+18. Jonathan Frankle, Sunoo Park, Daniel Shaar, Shafi Goldwasser, and Daniel J. Weitzner. Practical accountability of secret processes. In William Enck and Adrienne Porter Felt, editors, *USENIX Security 2018*, pages 657–674. USENIX Association, August 2018.

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

Fuc19. Georg Fuchsbauer. WI is not enough: Zero-knowledge contingent (service) payments revisited. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 49–62. ACM Press, November 2019.

GGP10. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Heidelberg, August 2010.

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.

GKK$^+$19. Lorenzo Grassi, Daniel Kales, Dmitry Khovratovich, Arnab Roy, Christian Rechberger, and Markus Schofnegger. Starkad and poseidon: New hash functions for zero knowledge proof systems. Cryptology ePrint Archive, Report 2019/458, 2019. https://eprint.iacr.org/2019/458.

GKM$^+$18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018.

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.

GMNO18. Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-SNARKs from square span programs. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 556–573. ACM Press, October 2018.

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

GR19. Alonso González and Carla Ràfols. Sublinear pairing-based arguments with updatable crs and weaker assumptions. *IACR Cryptology ePrint Archive*, 2019:326, 2019.

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.

Gro10a. 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.

Gro10b. Jens Groth. A verifiable secret shuffle of homomorphic encryptions. *Journal of Cryptology*, 23(4):546–579, October 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.

HBHW19. Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification: Version 2019.0.6 [overwinter+sapling], 2019.

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 2016*, pages 283–295. ACM Press, October 2016.

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.

KLO19. Jihye Kim, Jiwon Lee, and Hyunok Oh. Updatable crs simulation-extractable zk-snarks with a single verification. Cryptology ePrint Archive, Report 2019/586, 2019. https://eprint.iacr.org/2019/586.

KMP16. Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Heidelberg, August 2016.

KMS+16. Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858. IEEE Computer Society Press, May 2016.

KZM+15. Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093, 2015. https://eprint.iacr.org/2015/1093.

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.

Lip13. Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, December 2013.

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

LP19. Chaoyun Li and Bart Preneel. Improved interpolation attacks on cryptographic primitives of low algebraic degree. Cryptology ePrint Archive, Report 2019/812, 2019. https://eprint.iacr.org/2019/812, to appear at SAC'19.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

MGGM18. Alexander Mühle, Andreas Grüner, Tatiana Gayvoronskaya, and Christoph Meinel. A survey on essential components of a self-sovereign identity. *Computer Science Review*, 30:80–86, 2018.

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.

PS96. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Heidelberg, May 1996.

RST18. Christian Rechberger, Hadi Soleimany, and Tyge Tiessen. Cryptanalysis of low-data instances of full LowMCv2. *IACR Trans. Symm. Cryptol.*, 2018(3):163–181, 2018.

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

Sah01. Amit Sahai. Simulation-sound non-interactive zero knowledge. Technical report, IBM RESEARCH REPORT RZ 3076, 2001.

Sch90. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

SK95. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *EUROCRYPT'95*, volume 921 of *LNCS*, pages 393–403. Springer, Heidelberg, May 1995.

A Omitted Primitives

A.1 Non-Interactive Zero-Knowledge

Let RGen be a relation generator, such that $\mathsf{RGen}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathcal{R} = \{(\mathtt{x}, \mathtt{w})\}$. Here, $\mathtt{x}$ is the statement and $\mathtt{w}$ is the witness. We assume that λ is explicitly deducible from the description of $\mathcal{R}$. The relation generator also outputs auxiliary information $\mathsf{aux}_\mathcal{R}$ that will be given to the honest parties and the adversary. Let $\mathcal{L}_\mathcal{R} = \{\mathtt{x} : \exists \mathtt{w}, (\mathtt{x}, \mathtt{w}) \in \mathcal{R}\}$ be an NP-language. Non-interactive zero-knowledge (NIZK) proofs and arguments in the CRS model consist of algorithms $(\mathsf{KGen}_{\mathsf{crs}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$, and satisfy the following properties: completeness (for all common reference strings $\mathtt{crs}$ generated by $\mathsf{KGen}_{\mathsf{crs}}$ and $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$, we have that $\mathsf{V}(\mathtt{crs}, \mathtt{x}, \mathsf{P}(\mathtt{crs}, \mathtt{x}, \mathtt{w})) = 1$), zero-knowledge (there exists a simulator Sim that outputs a simulated proof such that an adversary cannot distinguish it from proofs computed by $\mathsf{P}(\mathtt{crs}, \mathtt{x}, \mathtt{w})$), soundness (an adversary cannot output a proof π and an instance $\mathtt{x} \notin \mathcal{L}_\mathcal{R}$ such that $\mathsf{V}(\mathtt{crs}, \mathtt{x}, \pi) = 1$. Moreover, knowledge soundness steps further and says that for any prover generating a valid proof there is an extractor Ext that can extract a valid witness.

A.2 QAPs and R1CS

Quadratic Arithmetic Programs (QAPs) have been introduced by Gennaro et al. [GGPR13] as a language where for an input $\mathtt{x}$ and witness $\mathtt{w}$, $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$ can be verified by using a parallel quadratic check, and that has an efficient reduction from a well-known language (either Boolean or Arithmetic) Circuit-SAT.

Definition 14 (QAP). *A quadratic arithmetic program over a field $\mathbb{F}$ is a tuple of the form*

$$\Big(\mathbb{F}, n, \{\boldsymbol{A}_i(X), \boldsymbol{B}_i(X), \boldsymbol{C}_i(X)\}_{i=0}^{i=m}; \boldsymbol{D}(X)\Big)$$

where $\boldsymbol{A}_i(X), \boldsymbol{B}_i(X), \boldsymbol{C}_i(X), \boldsymbol{D}(X) \in \mathbb{F}[X]$*, define a language of statements* $(s_1, \ldots, s_n) \in \mathbb{F}$ *and witnesses* $(s_{n+1}, \ldots, s_m) \in \mathbb{F}^{m-n}$ *such that*

$$\left(\sum_{i=0}^{m} s_i \boldsymbol{A}_i(X)\right) \cdot \left(\sum_{i=0}^{m} s_i \boldsymbol{B}_i(X)\right) = \left(\sum_{i=0}^{m} s_i \boldsymbol{C}_i(X)\right) + \boldsymbol{H}(X) \cdot \boldsymbol{D}(X) \qquad (2)$$

where $s_0 = 1$ *and for some degree-*$(d-2)$ *quotient polynomial* $\boldsymbol{H}(X)$*, where* d *is the degree of* $\boldsymbol{D}(X)$*. Let the degrees of all* $\boldsymbol{A}_i(X), \boldsymbol{B}_i(X)$ *and* $\boldsymbol{C}_i(X)$ *are at most* $d-1$*.*

We note that all the considered SNARK constructions are for QAPs defined over a bilinear group. Thus we consider relation generators RGen of the following form:

Definition 15 (QAP relation). *A QAP relation generator* RGen *is a PT algorithm that on input* λ *returns a relation description* $\mathcal{R} = (\mathsf{pars}, n, (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) \in \mathbb{F}^{(d-1)}[X]^{m-1}, \boldsymbol{D} \in \mathbb{F}^{(d)}[X])$ *where* pars *is a bilinear group whose order* p *defines* $\mathbb{F} := \mathbb{Z}_p$ *and* $n \leq m$*. Fix* $\mathtt{x} \in \mathbb{F}^n$ *and* $\mathtt{w} \in \mathbb{F}^{m-n}$*, we define* $\mathcal{R}(\mathtt{x}, \mathtt{w}) = 1$ *if there exists* $\boldsymbol{H}(X) \in \mathbb{F}[X]$ *so that Eq. (2) holds for* $\mathtt{x} = (s_1, \ldots, s_n)$ *and* $\mathtt{w} = (s_{n+1}, \ldots s_m)$*.*

For reducing arithmetic circuits to QAP relations, circuits can first be transformed to a system of rank-1 quadratic equations (R1CS) which is latter transformed into a QAP [BCG+13]. The R1CS relation over a field $\mathbb{F}$ consists of instance-witness pairs $((A, B, C, \boldsymbol{v}), \boldsymbol{w})$ with matrices $A, B, C \in \mathbb{F}^{n \times m}$ and vectors $\boldsymbol{v}, \boldsymbol{w}$ such that $(A\boldsymbol{z}) \circ (B\boldsymbol{z}) = C\boldsymbol{z}$ with $\boldsymbol{z} = (1, \boldsymbol{v}, \boldsymbol{w}) \in \mathbb{F}^m$ where $\circ$ denotes the entry-wise product. For capturing arithmetic circuit satisfaction, A, B, C represent the gates, v the public inputs, and w the private inputs and wire values.

A.3 Public-key Encryption

Definition 16. *A public key encryption scheme* $\Omega = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *consists of the following PPT algorithms:*

$\mathsf{KGen}(1^\lambda)$: *Given a security parameter* λ *it outputs the secret key* sk *and public key* pk *with message space* $\mathcal{M}$*.*

$\mathsf{Enc}(\mathsf{pk}, m)$: *Given a public key* pk *and a message* $m \in \mathcal{M}$ *it outputs a ciphertext* c*.*

$\mathsf{Dec}(\mathsf{sk}, C)$: *Given a secret key* sk *and a ciphertext* c *it outputs a message* $m \in \mathcal{M} \cup \{\perp\}$*.*

We say that an encryption scheme Ω is perfectly correct if for all $\kappa \in \mathbb{N}$, for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$ and for all $m \in \mathcal{M}$ it holds that $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m$. Below, we recall the standard notion of indistinguishability under chosen plaintext attacks ($\mathsf{IND\text{-}CPA}$ security).

Definition 17 (IND-CPA). *A public key encryption scheme Ω is* IND-CPA *secure, if for all PPT adversaries $\mathcal{A}$ it holds that*

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda), b \leftarrow_{\$} \{0,1\}, \\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pk}), b^* \leftarrow \mathcal{A}(\mathsf{Enc}(\mathsf{pk}, m_b), \mathsf{st}) : \\ b = b^*\end{array}\right] \approx_\lambda \frac{1}{2}.$$

A.4 Schnorr Signatures

We recall the Schnorr signature scheme [Sch90] together with the required Adapt algorithm (cf. [DS19]) in Fig. 4. It can be shown to provide EUF-CMA security in the random oracle model (ROM) under the DLP in $\mathbb{G}$ by using the now popular rewinding technique [PS96] (cf. also [KMP16] for a recent treatment on tightness and optimality of such reductions). In the following we present Schnorr signatures with respect to a common setup, i.e., $\mathsf{PP} \leftarrow \mathsf{PGen}(1^\lambda)$ are given to all instances of KGen and let GGen be a group generator that on input 1^λ outputs the description of a prime order group $\mathcal{G} = (\mathbb{G}, g, p)$ with order p s.t. $\lambda = \log_2 p$ and generator g. Recall, that in addition Schnorr requires a collision resistant hash function $H : \mathbb{G} \times \mathcal{M} \to \mathbb{Z}_p$ (formally sampled uniformly at random from a family $\{H_k\}_{k \in \mathcal{K}}$ of hash functions) and thus we have $\mathsf{PP} := (\mathcal{G}, H)$ (which we assume to be an implicit input to all algorithms). We recall a lemma from [DS19] showing that Schnorr signatures using the Adapt algorithm in Fig. 4 satisfies the signature adaption notion in Definition 8.

Lemma 1 (**[DS19]**). *Schnorr signatures are adaptable according to Definition 8.*

A.5 Groth's Strong One-Time Signatures

In Fig. 5 we recall the strong one-time signature scheme from Groth [Gro06] and its security below:

Theorem 4 (**[Gro06]**). *Assuming hardness of computing discrete logarithms and collision-resistance of the hash function, the scheme* ($\mathsf{PGen}_{\mathsf{ots}}$, $\mathsf{KGen}_{\mathsf{ots}}$, $\mathsf{Sign}_{\mathsf{ots}}$, $\mathsf{Verify}_{\mathsf{ots}}$) *described in Fig. 5 is a strong one-time signature scheme for signing messages $m \in \{0,1\}^*$ with perfect correctness.*

A.6 BDH Knowledge Assumption

Let BGen be a PPT algorithm that, on input a security parameter λ, outputs $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g})$ for generators g and $\hat{g}$ of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $\Theta(\lambda)$-bit prime p.

Assumption 1 (BDH-**Knowledge Assumption [ABLZ17]**) *We say that* BGen *is* BDH-KE secure *for $\mathcal{R}$ if for any λ, $(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}) \in \mathrm{im}(\mathcal{R}(1^\lambda))$, and PPT adversary $\mathcal{A}$ there exists a PPT extractor $\mathsf{Ext}^{\mathsf{BDH}}_{\mathcal{A}}$, such that*

$$\Pr\left[\begin{array}{l}r \leftarrow_r \mathsf{RND}(\mathcal{A}), \\ (V, \hat{V} || a) \leftarrow (\mathcal{A} || \mathsf{Ext}^{\mathsf{BDH}}_{\mathcal{A}})(\mathcal{R}, \mathsf{aux}_{\mathcal{R}}; \omega_{\mathcal{A}}) : \\ e(V, \hat{g}) = e(g, \hat{V}) \wedge g^a \neq V\end{array}\right] \approx_\lambda 0.$$

$\mathsf{PGen}(1^\lambda)$

- $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$; $H \leftarrow_{\$} \{H_k\}_{k \in \mathcal{K}}$;
- **return** $\mathsf{PP} := (\mathcal{G}, H)$;

$\mathsf{KGen}(\mathsf{PP})$:

- Parse $\mathsf{PP} = ((\mathbb{G}, g, p), H)$;
- $x \leftarrow_{\$} \mathbb{Z}_p$;
- **return** $(\mathsf{sk}, \mathsf{pk}) := (x, g^x)$.

$\mathsf{Sign}(\mathsf{sk}, m)$:

- Parse $\mathsf{sk} = x$;
- $r \leftarrow_{\$} \mathbb{Z}_p$; $R := g^r$; $c := H(R \| m)$; $y := r + x \cdot c \bmod p$
- **return** $\sigma := (c, y)$.

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$:

- Parse $\mathsf{pk} = g^x; \sigma = (c, y)$;
- **if** $c = H((g^x)^{-c} g^y, m)$ **return** 1 **else return** 0.

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$:

- Parse $\mathsf{pk} = g^x$; $\sigma = (c, y)$; $\Delta \in \mathbb{Z}_p$;
- $\mathsf{pk}' := g^x \cdot g^\Delta$; $y' := y + c \cdot \Delta \bmod p$;
- **return** $\sigma' := (c, y')$.

Fig. 4. Schnorr signatures.

Note that the BDH assumption can be considered as a simple case of the PKE assumption of [DFGK14] (where $\mathcal{A}$ is given as an input the tuple $\{(g^{x^i}, \hat{g}^{x^i})\}_{i=0}^n$ for some $n \geq 0$, and assumed that if $\mathcal{A}$ outputs $(V, \hat{V})$ then she knows $(a_0, a_1, \ldots, a_n)$, such that $V = g^{\sum_{i=0}^n a_i x^i}$ as used in the case of asymmetric pairings in [DFGK14]. Thus, BDH can be seen as an asymmetric-pairing version of the original and by now well established KoE assumption due to Damgård [Dam92].

$\mathsf{PGen}_{\mathsf{ots}}(1^\lambda)$

- $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda);\ H \leftarrow_{\$} \{H_k\}_{k \in \mathcal{K}};$
- **return** $\mathsf{PP} := (\mathcal{G}, H);$

$\mathsf{KGen}_{\mathsf{ots}}(\textsc{pp})$:

- Parse $\mathsf{PP} = ((\mathbb{G}, g, p), H);$
- $x_s, y_s, r_s, s_s \leftarrow_{\$} \mathbb{Z}_p;$
- $f_s := g^{x_s};\ h_s := g^{y_s};\ c_s := g_s^{r_s} \cdot h_s^{s_s};$
- **return** $(\mathsf{sk}, \mathsf{pk}) := ((x_s, y_s, r_s, s_s), (f_s, h_s, c_s)).$

$\mathsf{Sign}_{\mathsf{ots}}(\mathsf{sk}, m)$:

- Parse $\mathsf{sk} = (x_s, y_s);$
- $r \leftarrow_{\$} \mathbb{Z}_p;\ z := x_s(r_s - r) + y_s \cdot s_s - H(m) \cdot y_s^{-1} \bmod p$
- **return** $\sigma := (r, z).$

$\mathsf{Verify}_{\mathsf{ots}}(\mathsf{pk}, m, \sigma)$:

- Parse $\mathsf{pk} = (f_s, h_s, c_s); \sigma = (r, z);$
- **if** $c_s = g^{H(m)} \cdot f_s^r \cdot h_s^s$ **return** 1 **else return** 0.

Fig. 5. Groth's strong one-time signature scheme.