Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs

Jens Groth[⋆] and Mary Maller[⋆⋆]

University College London
{j.groth, mary.maller.15}@ucl.ac.uk

Abstract. We construct a pairing-based simulation-extractable succinct non-interactive argument of knowledge (SE-SNARK) that consists of only 3 group elements and has highly efficient verification. By formally linking SE-SNARKs to signatures of knowledge, we then obtain a succinct signature of knowledge consisting of only 3 group elements.
SE-SNARKs enable a prover to give a proof that they know a witness to an instance in a manner which is: (1) *succinct* - proofs are short and verifier computation is small; (2) *zero-knowledge* - proofs do not reveal the witness; (3) *simulation-extractable* - it is only possible to prove instances to which you know a witness, even when you have already seen a number of simulated proofs.
We also prove that any pairing-based signature of knowledge or SE-SNARK must have at least 3 group elements and 2 verification equations. Since our constructions match these lower bounds, we have the smallest size signature of knowledge and the smallest size SE-SNARK possible.

Keywords. Signature of knowledge, SNARK, non-interactive zero-knowledge proof, simulation-extractability.

1 Introduction

Non-interactive zero-knowledge (NIZK) arguments enable a prover to convince a verifier that they know a witness to an instance being member of a language in NP, whilst revealing no information about this witness. Recent works have looked into building NIZK arguments that are efficient enough to use in scenarios where a large number of proofs need to be stored and where verifiers have limited computational resources. Such arguments are called succinct NIZK arguments, or zk-SNARKs (zero-knowledge succinct non-interactive arguments of knowledge). A weakness of zk-SNARKs is that they are, currently without exception, susceptible to man-in-the-middle attacks where the adversary can modify a zk-SNARK into a new one. As a result, any application intending to use zk-SNARKs where malleability is a concern has to take additional measures to ensure security e.g. signing the instance and proof. Conversely, schemes that do not require succinctness can take advantage of a primitive called signatures of knowledge (SoKs).

Signatures of knowledge [CS97,CL06] generalise signatures by replacing the public verification key with an instance in an NP-language. A signer who holds a witness for the instance can create signatures, and somebody who does not know a witness for the instance cannot sign. SoKs should not reveal the witness, since this would enable others to sign with respect to the same witness. Chase and Lysyanskaya [CL06] therefore define signatures of knowledge

[⋆] The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307937

[⋆⋆] Supported by a scholarship from Microsoft Research

to be simulatable: if you have a trapdoor associated with some public parameters, you can simulate the signature without the witness, and hence the signature cannot be disclosing information about the witness. Moreover, in the spirit of strong existential unforgeability for digital signatures, we want it to be the case that even after seeing many signatures under different instances, it should still not be possible to create a new signature unless you know a witness. Chase and Lysyanskaya capture this property through the notion of simulation-extractability where you may obtain arbitrary simulated signatures, but still not create a new signature not seen before unless you know the witness for the instance.

Both zk-SNARKs and SoKs are key building blocks in numerous cryptographic applications, including but not limited to: ring signatures, group signatures, policy based signatures, cryptocurrencies, anonymous delegatable credentials and direct anonymous attestation [DS16,BCK+14] [BF14,MGGR13,BFG13a].

Our contribution. We construct a succinct simulation-extractable NIZK argument, or an SE-SNARK. Our construction is pairing-based. Given three groups with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$, our proofs consist of only 3 group elements from the source groups: 2 from $\mathbb{G}_1$ and 1 from $\mathbb{G}_2$. The proofs also have fast verification with verifiers needing to check just 2 pairing product equations.

By exploiting the link between SoKs and SE-NIZK arguments, we show that our construction also yields a succinct SoK. We formally define the notions of succinct SoKs and SE-NIZK arguments. Then we construct SoKs from SE-NIZK arguments and also prove the reverse implication that SoKs give rise to SE-NIZK arguments. Our SoK inherits the high efficiency of our SE-NIZK argument, in particular that it consists of only 3 group elements.

We also prove a lower bound: a pairing-based SE-NIZK argument for a non-trivial language in NP must have at least 2 verification equations and 3 group elements. Due to our proof that any pairing-based SoK yields a pairing-based SE-NIZK (where the signature size equals the proof size and the number of verification equations are equal), this lower bound also applies to the signature size and the number of verification equations in SoKs. Our constructions are therefore optimal with respect to size and number of verification equations. We note that the lower bound improves on previous lower bounds on standard NIZK arguments by explicitly taking advantage of the simulation-extractability properties in the proof.

Our construction of an SE-SNARK compares well with the state of the art pairing-based zk-SNARKs. Groth [Gro16] gave a 3 element zk-SNARK, however, it is not simulation-extractable and it only has a proof of security in the generic group model. While we pay a price in computational efficiency, our simulation-extractable SNARK matches the size of Groth's zk-SNARK. We also get comparable verification complexity and unlike Groth's zk-SNARK we give a security proof based on concrete intractability assumptions instead of relying on the full generic group model. Ben-Sasson, Chiesa, Tromer, and Virza gave an 8 element zk-SNARK which is also not simulation-extractable, however they do have smaller prover computation [BSCTV14]. Compared to other pairing-based zk-SNARKs in the literature we have both the simulation-extractability property and also better efficiency. In Table 1 we give a comparison of our simulation-extractable SNARK with these prior zk-SNARKs.

Our construction of a succinct signature of knowledge is the first in any computational model. This reduces the size of the signatures, albeit at the expense of having more public parameters. For applications where the public parameters need only be generated once, such as DAA and anonymous cryptocurrencies, this can be advantageous. A comparison with the

	Groth	BCTV	This work
CRS size	$m + 2n + 3$ $\mathbb{G}_1$ $n + 3$ $\mathbb{G}_2$	$6m + n - \ell$ $\mathbb{G}_1$ m $\mathbb{G}_2$	$m + 4n + 5$ $\mathbb{G}_1$ $2n + 3$ $\mathbb{G}_2$
Proof size	2 $\mathbb{G}_1$, 1 $\mathbb{G}_2$	7 $\mathbb{G}_1$, 1 $\mathbb{G}_2$	2 $\mathbb{G}_1$, 1 $\mathbb{G}_2$
Prover computation	$m + 3n - \ell + 3$ E_1 $n + 1$ E_2	$6m + n - \ell$ E_1 m E_2	$m + 4n - \ell$ E_1 $2n$ E_2
Verifier computation	ℓ E_1, 3 P	ℓ E_1, 12 P	ℓ E_1, 5 P
Verification equations	1	5	2

Table 1. Comparison for arithmetic circuit satisfiability with ℓ element instance, m wires, n multiplication gates. Since our work uses squarings gates, we have conservatively assumed n multiplication gates translate to $2n$ squaring gates; if a circuit natively has many squaring gates our efficiency would therefore improve compared to Groth and BCTV. Units: $\mathbb{G}$ means group elements, E means exponentiations and P means pairings.

most efficient prior signature of knowledge by Bernhard, Fuchsbauer and Ghadafi [BFG13a] is given in Table 2. The BFG scheme uses standard assumptions, as opposed to ours which uses knowledge extractor assumptions. It is difficult to directly compare computational efficiency since the languages are different; our work uses arithmetic circuits whereas the BFG scheme uses satisfiability of a set of pairing product equations. Therefore, we get better efficiency for arithmetic circuits and they get better efficiency for pairing product equations. However, what is clear is that we make big efficiency gains in terms of the signature size and the number of verification equations.

	BFG	This work
Public Parameters	$10 + \lambda$	$8 + 6n + m$
Signer Computation.	$\Omega(\lvert \boldsymbol{w} \rvert + n_p)$	$m + 6n$
Signature Size	$O(\lvert \boldsymbol{w} \rvert + n_p)$	3
Verification Equations	$O(n_p)$	2

Table 2. Comparison of signatures of knowledge schemes. We use m and n for the number of wires and multiplication gates in our arithmetic curcuit, λ refers to the security parameter; $|\boldsymbol{w}|$ is the witness size and n_p is the number of pairing product equations in BFG (one can translate an arithmetic circuit to pairing product equations, in which case $n_p = n$). Size is measured in number of group elements and computation in the number of exponentiations.

Techniques and challenges. Standard definitions of signatures of knowledge [CL06] and simulation-extractable NIZK proofs [Gro06] assume the ability to encrypt the witness, which can then be decrypted using a secret extraction key. However, since we are interested in having succinct signatures and proofs, we do not have space to send a ciphertext. Instead we give new definitions that use non-black-box extraction. Roughly, the definitions say that given the signer's or prover's state it is possible to extract a witness if it succeds in creating a valid signature or proof.

To formalise the close link between SoKs and SE-NIZK arguments, we illustrate how to build a relation which includes the signature's message as part of the instance to be proved. Given an SE-NIZK for this relation, we build an SoK for the same relation only without the message encoded. This SoK is built solely from a target collision-resistant hash function and the SE-NIZK argument. The SoK is proven to be simulation-extractable directly from the definition of simulation-extractability of the NIZK argument. Once this link has been formalized, the rest of the paper focuses on how to build SE-SNARKs with optimum efficiency.

Our SE-SNARK is pairing-based. The common reference string describes a bilinear group and some group elements, the proofs consist of group elements, and the verifier checks that the exponents of the proofs satisfy quadratic equations by calculating products of pairings. The underlying relation is a square arithmetic program, which is a SNARK-friendly characterisation of arithmetic circuits. Square arithmetic programs are closely related to quadratic arithmetic programs [GGPR13], but use only squarings instead of arbitrary multiplications. As suggested by Groth [Gro16] the use of squarings give nice symmetry properties, which in our case makes it possible to check different parts of the proof against each other and hence make it intractable for an adversary to modify them without knowing a witness.

The security of our construction is based on concrete intractability assumptions. For standard knowledge soundness our strongest intractability assumption is similar to the power knowledge of exponent assumption used in [DFGK14]. To go beyond knowledge soundness to the stronger simulation-extractability property requires a stronger assumption, probably unavoidably so. We formulate the eXtended Power Knowledge of Exponent (XPKE) assumption, which assumes that an adversary cannot find elements in two source groups that have a linear relationship between each other unless it already knows what this relationship is - not even if it can query an oracle for functions of these exponents.

Finally, we rely on Groth's [Gro16] definition of pairing-based non-interactive arguments and rule out the existence of SE-NIZK arguments with 1 verification equation or 2 group elements. Groth [Gro16] already ruled out 1 element NIZK arguments by exploiting that if there is only one group element then the verification equations are linear in the exponents and easy to fool. It is an open problem from [Gro16] whether regular NIZK arguments can have 2 group element proofs, a more difficult problem since a pairing of two group elements gives rise to quadratic verification equations in the exponents. We show that in the case of SE-NIZK arguments 2 group elements is not possible by leveraging the simulation-extractability property to deal also with quadratic verification equations.

Related work. Signatures of knowledge are a core ingredient in many cryptographic protocols. For example, [CF08,GT07,BFG$^+$13b,BFG13a,YYQ$^+$15,FXC09] are DAA schemes that use SoKs. Anonymous cryptocurrencies can also be constructed using signatures of knowledge, for example Zero-Coin [MGGR13]. In order to make sufficient efficiency gains so that it could be deployed, the Zcash cryptocurrency [BCG$^+$14] instead uses zk-SNARKs. To use zk-SNARKs, Zcash has to take extra steps to avoid malleability (man-in-the-middle) attacks. Specifically, Zcash samples a key pair for a one-time signature scheme; computes MACs to tie the signing key to the identities secret keys; modifies the instance to include signature verifying key and the MACs; and finally uses the signing key to sign the transaction. However, the use of succinct SoKs for cryptocurrencies would yield the same, if not better, efficiency as the use of zk-SNARKs and the resulting models would be simpler.

NIZK proofs originated with Blum, Feldman and Micali [BDSMP91,BFM88] and there has been many works making both theoretical advances and efficiency improvements [FLS99] [SP92,KP98,DSDCP00,Dam92,GO14,Gro10,GGI$^+$15]. Groth, Ostrovsky and Sahai [GOS12] proposed the first pairing-based NIZK proofs and subsequent works [Gro06,GS12] have yielded efficient NIZK proofs that can be used in pairing-based protocols. NIZK proofs with unconditional soundness need to be linear in the witness size. However, for NIZK arguments with computational soundness it is possible to get succinct proofs that are smaller than the size of the witness [Mic00,Kil95].

The practical improvements were accompanied by theoretical works on how SNARKs compose [BSCTV14,Val08,BCCT13] and on the necessity of using strong cryptographic assumptions when building SNARKs [AF07,GW11,BCI^{+}13,BCPR13,BP14]. The latter works give methods to take SNARKs with long common reference strings and build SNARKs with common reference string size that is independent of the instance size, i.e., fully succinct SNARKs. Using these techniques on our simulation-extractable SNARK, which has a long common reference string, gives a fully succinct SE-SNARK.

Sahai [Sah99] introduced simulation-soundness of NIZK proofs as a notion to capture that even after seeing simulated proofs it is not possible to create a fake proof for a false instance unless copying a previous simulated proof. Combining this with proofs of knowledge, Groth [Gro06] defined the even stronger security notion that we should be able to extract a witness from an adversary that creates a valid new proof, even if this adversary has seen many simulated proofs for arbitrary instances. Faust, Kohlweiss, Marson, and Venturi discuss how to achieve simulation soundness in the random oracle model [FKMV12]. Kosba et al. [PVW08] discuss how to lift any zk-SNARK into a simulation-extractable one, however they do so by appending an encryption of the witness to the proof, so the result is no longer succinct.

Camenisch [CS97] coined the term signatures of knowledge to capture zero-knowledge protocols relying on techniques used in Schnorr signatures [Sch91]. Signatures of knowledge have been used in many constructions albeit without a precise security definition. Chase and Lysyanskaya [CL06] gave the first formal definition of signatures of knowledge. They also broke the tight connection with Schnorr signatures and NIZK arguments based on cyclic groups and the Fiat-shamir heuristic and instead provided a general construction from simulation-sound NIZK proofs and dense public key encryption. An alternative definition of signatures of knowledge was given by Fischlin and Onete [FO11] which requires witness indistinguishability as opposed to full zero-knowledge.

2 Definitions

2.1 Notation

We write $y \leftarrow S$ for sampling y uniformly at random from the set S. We write $y \leftarrow A(x)$ for a probabilistic algorithm on input x returning output y. When we want to be explicit about running a probabilistic algorithm on random coins r, we write $y = A(x; r)$. We use the abbreviation PPT for probabilistic polynomial time algorithms. For an algorithm $\mathcal{A}$ we define $\mathsf{trans}_{\mathcal{A}}$ to be a list containing all of $\mathcal{A}$'s inputs and outputs, including random coins.

When considering security of our cryptographic schemes, we will assume there is an adversary $\mathcal{A}$. The security of our schemes will be parameterised by a security parameter $\lambda \in \mathbb{N}$. The intuition is that the larger the security parameter, the better security we get. For functions $f, g : \mathbb{N} \rightarrow [0; 1]$ we write $f(\lambda) \approx g(\lambda)$ if $|f(\lambda) - g(\lambda)| = \lambda^{-\omega(1)}$. We say a function f is negligible if $f(\lambda) \approx 0$ and overwhelming if $f(\lambda) \approx 1$. We will always implicitly assume all participants and the adversary know the security parameter, i.e., from their input they can efficiently compute the security parameter in unary representation 1^{λ}.

We use games in security definitions and proofs. A game $\mathcal{G}$ has a number of procedures including a main procedure. The main procedure outputs either 0 or 1 depending on whether the adversary succeeds or not. $\Pr[\mathcal{G}]$ denotes the probability that this output is 1.

2.2 Relations

Let $\mathcal{R}$ be a relation generator that given a security parameter λ in unary returns a polynomial time decidable relation $R \leftarrow \mathcal{R}(1^\lambda)$. For $(\boldsymbol{\phi}, \boldsymbol{w}) \in R$ we call $\boldsymbol{\phi}$ the instance and $\boldsymbol{w}$ the witness. We define $\mathcal{R}_\lambda$ to be the set of possible relations $\mathcal{R}(1^\lambda)$ might output.

2.3 Hard Decisional Problems

A relation R is sampleable if there are two algorithms, Yes and No such that:

- Yes samples instances and witnesses in the relation.
- No samples instances outside the language L_R defined by the relation.

When proving our lower bounds for the efficiency of SE-NIZK arguments, we will assume the existence of sampleable relations where it is hard to tell whether an instance ϕ has been sampled by Yes or No.

Definition 2.1. *Let $\mathcal{R}$ a relation generator, and let Yes, No be two PPT algorithms such that for $R \leftarrow \mathcal{R}(1^\lambda)$ we have $\mathsf{Yes}(R) \to (\boldsymbol{\phi}, \boldsymbol{w}) \in R$ and $\mathsf{No}(R) \to \boldsymbol{\phi} \notin L_R$, and let $\mathcal{A}$ be an adversary. Define $\mathbf{Adv}^{DP}_{\mathcal{R},\mathsf{Yes},\mathsf{No},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{DP}_{\mathcal{R},\mathsf{Yes},\mathsf{No},\mathcal{A}}(\lambda)] - 1$ where $\mathcal{G}^{DP}_{\mathcal{R},\mathsf{Yes},\mathsf{No},\mathcal{A}}(1^\lambda)$ is given by*

```
MAIN $\mathcal{G}^{DP}_{\mathcal{R},\mathsf{Yes},\mathsf{No},\mathcal{A}}(\lambda)$
$R \leftarrow \mathcal{R}(1^\lambda)$
$\boldsymbol{\phi}_0 \leftarrow \mathsf{No}(R);\ (\boldsymbol{\phi}_1, \boldsymbol{w}) \leftarrow \mathsf{Yes}(R)$
$b \leftarrow \{0,1\}$
$b' \leftarrow \mathcal{A}(R, \boldsymbol{\phi}_b)$
return 1 if $b = b'$ and else return 0
```

We say $\mathsf{Yes}, \mathsf{No}$ provide a hard decisional problem for $\mathcal{R}$ if for all PPT adversaries $\mathcal{A}$, $\mathbf{Adv}^{DP}_{\mathcal{R},\mathsf{Yes},\mathsf{No},\mathcal{A}}(\lambda) \approx 0$.

2.4 Signatures of Knowledge

Signatures of knowledge [CL06] (SoKs) generalise digital signatures by replacing the public key with an instance in a language in NP. If you have a witness for the instance, you can sign a message. If you do not know a witness, then you cannot sign. The notion of SoKs mimic digital signatures with strong existential unforgeability; even if you have seen many signatures on arbitrary messages under arbitrary instances, you cannot create a new signature not seen before without knowing the witness for the instance.

Signatures of knowledge are closely related to simulation-extractable NIZK arguments and previous constructions have exploited the link between SoKs and NIZK proofs. In the following, we define signatures of knowledge, simulation-extractable NIZK arguments, and give a formal proof that signatures of knowledge can be constructed from simulation-extractable NIZK arguments. When we later in the article construct compact and easy to verify SE-NIZK arguments, i.e., simulation-extractable SNARKs, we will therefore automatically obtain compact and easy to verify SoKs.

For our definition of a simulation-extractable signature of knowledge, we follow the game based definitions of Chase and Lysyanskaya [CL06]. However, Chase and Lysyanskaya define their relations with respect to Turing Machines, whereas in our definitions the use of Turing Machines is implicit in the relation generator. Another more important difference is that since we want compact signatures, we give a non-black-box definition of simulation-extractability.

Definition 2.2. *Let $\mathcal{R}$ be a relation generator and let $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of message spaces. Then the quintet of efficient algorithms* $(\mathsf{SSetup}, \mathsf{SSign}, \mathsf{SVfy}, \mathsf{SSimSetup}, \mathsf{SSimSign})$ *is a signature of knowledge scheme for $\mathcal{R}$ and $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ if it is correct, simulatable and simulation-extractable (defined below) and works as follows:*

- $\boldsymbol{pp} \leftarrow \mathsf{SSetup}(R)$*: the setup algorithm is a PPT algorithm which takes as input a relation $R \in \mathcal{R}_\lambda$ and returns public parameters* $\boldsymbol{pp}$.
- $\boldsymbol{\sigma} \leftarrow \mathsf{SSign}(\boldsymbol{pp}, \boldsymbol{\phi}, \boldsymbol{w}, m)$*: the signing algorithm is a PPT algorithm which takes as input the public parameters, a pair $(\boldsymbol{\phi}, \boldsymbol{w}) \in R$ and a message $m \in \mathcal{M}_\lambda$ and returns a signature* $\boldsymbol{\sigma}$.
- $0/1 \leftarrow \mathsf{SVfy}(\boldsymbol{pp}, \boldsymbol{\phi}, m, \boldsymbol{\sigma})$*: the verification algorithm is a deterministic polynomial time algorithm, which takes as input some public parameters* $\boldsymbol{pp}$*, an instance* $\boldsymbol{\phi}$*, a message $m \in \mathcal{M}_\lambda$, and a signature $\boldsymbol{\sigma}$ and outputs either* 0 *(reject) or* 1 *(accept).*
- $(\boldsymbol{pp}, \boldsymbol{\tau}) \leftarrow \mathsf{SSimSetup}(R)$ *: the simulated setup algorithm is a PPT algorithm which takes as input a relation $R \in \mathcal{R}_\lambda$ and returns public parameters $\boldsymbol{pp}$ and a trapdoor* $\boldsymbol{\tau}$.
- $\boldsymbol{\sigma} \leftarrow \mathsf{SSimSign}(\boldsymbol{pp}, \boldsymbol{\tau}, \boldsymbol{\phi}, m)$ *: the simulated signing algorithm is a PPT algorithm which takes as input some public parameters* $\boldsymbol{pp}$*, a simulation trapdoor* $\boldsymbol{\tau}$*, and an instance $\boldsymbol{\phi}$ and returns a signature* $\boldsymbol{\sigma}$.

Perfect Correctness: A signer with a valid witness can always produce a signature that will convince the verifier.

Definition 2.3. *A signature of knowledge scheme is perfectly correct if for all $\lambda \in \mathbb{N}$, for all $R \in \mathcal{R}_\lambda$, for all $(\boldsymbol{\phi}, \boldsymbol{w}) \in R$, and for all $m \in \mathcal{M}_\lambda$*

$$\Pr[\boldsymbol{pp} \leftarrow \mathsf{SSetup}(R); \boldsymbol{\sigma} \leftarrow \mathsf{SSign}(\boldsymbol{pp}; \boldsymbol{\phi}, \boldsymbol{w}, m) : \mathsf{SVfy}(\boldsymbol{pp}, \boldsymbol{\phi}, m, \boldsymbol{\sigma}) = 1] = 1.$$

Perfect Simulatability: The verifier should learn nothing from a signature about the witness that it did not already know. The secrecy of the witness is modelled by the ability to simulate signatures without the witness. More precisely, we say the signatures of knowledge are simulatable if there is a simulator that can create good looking public parameters and signatures without the witness.

Definition 2.4. *For a signature of knowledge SoK, define* $\mathbf{Adv}^{simul}_{SoK,\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{simul}_{SoK,\mathcal{A}}(\lambda)] - 1$ *where the game $\mathcal{G}^{simul}_{SoK,\mathcal{A}}$ is defined as follows*

```
MAIN $\mathcal{G}^{simul}_{SoK,\mathcal{A}}(\lambda)$
$R \leftarrow \mathcal{R}(1^\lambda)$
$\boldsymbol{pp}_0 \leftarrow \mathsf{SSetup}(R)$
$(\boldsymbol{pp}_1, \boldsymbol{\tau}) \leftarrow \mathsf{SSimSetup}(R)$
$b \leftarrow \{0,1\}$
$b' \leftarrow \mathcal{A}^{S^b_{\boldsymbol{pp}_b,\boldsymbol{\tau}}}(\boldsymbol{pp}_b)$
return 1 if $b = b'$ and return 0 otherwise
```

```
$S^0_{\boldsymbol{pp}_0,\boldsymbol{\tau}}(\boldsymbol{\phi}_i, \boldsymbol{w}_i, m_i)$
assert $(\boldsymbol{\phi}_i, \boldsymbol{w}_i) \in R \ \wedge \ m_i \in \mathcal{M}_\lambda$
$\boldsymbol{\sigma}_i \leftarrow \mathsf{SSign}(\boldsymbol{pp}_0, \boldsymbol{\phi}, \boldsymbol{w}, m)$
return $\boldsymbol{\sigma}_i$
```

```
$S^1_{\boldsymbol{pp}_1,\boldsymbol{\tau}}(\boldsymbol{\phi}_i, \boldsymbol{w}_i, m_i)$
assert $(\boldsymbol{\phi}_i, \boldsymbol{w}_i) \in R \ \wedge \ m_i \in \mathcal{M}_\lambda$
$\boldsymbol{\sigma}_i \leftarrow \mathsf{SSimSign}(\boldsymbol{pp}_1, \boldsymbol{\tau}, \boldsymbol{\phi}, m)$
return $\boldsymbol{\sigma}_i$
```

A signature of knowledge SoK is called perfectly simulatable if for any PPT adversary $\mathcal{A}$, $\mathbf{Adv}^{simul}_{SoK,\mathcal{A}}(\lambda) = 0$.

Simulation-Extractability: An adversary should not be able to issue a new signature unless it knows a witness. This should hold even if the adversary gets to see signatures on arbitrary messages under arbitrary instances. We model this notion in a strong sense, by letting the adversary see simulated signatures for arbitrary messages and instances, which potentially includes false instances. Even under this strong attack model, we require that whenever the adversary outputs a valid signature not seen before, it is possible to extract a witness for the instance if you have access to the internal data of the adversary.

Definition 2.5. *For a signature of knowledge SoK, let $\mathbf{Adv}^{sig\text{-}ext}_{SoK,\mathcal{A},\chi_{\mathcal{A}}}(\lambda) = \Pr[\mathcal{G}^{sig\text{-}ext}_{SoK,\mathcal{A},\chi_{\mathcal{A}}}(\lambda)]$ where the game $\mathcal{G}^{sig\text{-}ext}_{SoK,\mathcal{A},\chi_{\mathcal{A}}}$ is defined as follows*

```
MAIN $\mathcal{G}^{sig\text{-}ext}_{SoK,\mathcal{A},\chi_{\mathcal{A}}}(\lambda)$
$R \leftarrow \mathcal{R}(1^\lambda);\ Q = \emptyset$
$(\boldsymbol{pp}, \boldsymbol{\tau}) \leftarrow \mathsf{SSimSetup}(R)$
$(\boldsymbol{\phi}, m, \boldsymbol{\sigma}) \leftarrow \mathcal{A}^{\mathsf{SSimSign}_{\boldsymbol{pp},\boldsymbol{\tau}}}(\boldsymbol{pp})$
$\boldsymbol{w} \leftarrow \chi_{\mathcal{A}}(\mathsf{trans}_{\mathcal{A}})$
assert $(\boldsymbol{\phi}, \boldsymbol{w}) \notin R$
assert $(\boldsymbol{\phi}, m, \boldsymbol{\sigma}) \notin Q$
return $\mathsf{SVfy}(\boldsymbol{pp}, \boldsymbol{\phi}, m, \boldsymbol{\sigma})$
```

```
$\mathsf{SSimSign}_{\boldsymbol{pp},\boldsymbol{\tau}}(\boldsymbol{\phi}_i, m_i)$
$\boldsymbol{\sigma}_i \leftarrow \mathsf{SSimSign}(\boldsymbol{pp}, \boldsymbol{\tau}, \boldsymbol{\phi}_i, m_i)$
$Q = Q \cup \{(\boldsymbol{\phi}_i, m_i, \boldsymbol{\sigma}_i)\}$
return $\boldsymbol{\sigma}_i$
```

A signature of knowledge SoK is simulation-extractable if for any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\chi_{\mathcal{A}}$ such that $\mathbf{Adv}^{sig\text{-}ext}_{SoK,\mathcal{A},\chi_{\mathcal{A}}}(\lambda) \approx 0$.

2.5 Non-interactive Zero-Knowledge Arguments of Knowledge

Definition 2.6. *Let $\mathcal{R}$ be a relation generator. A NIZK argument for $\mathcal{R}$ is a quadruple of algorithms* $(\mathsf{ZSetup}, \mathsf{ZProve}, \mathsf{ZVfy}, \mathsf{ZSimProve})$*, which is complete, zero-knowledge and knowledge sound (defined below) and works as follows:*

- $(\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R)$*: the setup algorithm is a PPT algorithm which takes as input a relation $R \in \mathcal{R}_\lambda$ and returns a common reference string* $\mathbf{crs}$ *and a simulation trapdoor $\boldsymbol{\tau}$.*
- $\boldsymbol{\pi} \leftarrow \mathsf{ZProve}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{w})$*: the prover algorithm is a PPT algorithm which takes as input a common reference string* $\mathbf{crs}$ *for a relation R and $(\boldsymbol{\phi}, \boldsymbol{w}) \in R$ and returns a proof $\boldsymbol{\pi}$.*
- $0/1 \leftarrow \mathsf{ZVfy}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{\pi})$*: the verifier algorithm is a deterministic polynomial time algorithm which takes as input a common reference string* $\mathbf{crs}$*, an instance $\boldsymbol{\phi}$ and a proof $\boldsymbol{\pi}$ and returns* 0 *(reject) or* 1 *(accept).*
- $\boldsymbol{\pi} \leftarrow \mathsf{ZSimProve}(\mathbf{crs}, \boldsymbol{\tau}, \boldsymbol{\phi})$*: the simulator is a PPT algorithm which takes as input a common reference string* $\mathbf{crs}$*, a simulation trapdoor $\boldsymbol{\tau}$ and an instance $\boldsymbol{\phi}$ and returns a proof $\boldsymbol{\pi}$.*

Perfect Completeness: Perfect completeness says that given a true statement, a prover with a witness can convince the verifier.

Definition 2.7. $(\mathsf{ZSetup}, \mathsf{ZProve}, \mathsf{ZVfy}, \mathsf{ZSimProve})$ *is a perfectly complete argument system for $\mathcal{R}$ if for all $\lambda \in \mathbb{N}$, for all $R \in \mathcal{R}_\lambda$ and for all $(\boldsymbol{\phi}, \boldsymbol{w}) \in R$:*

$$\Pr\left[(\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R); \boldsymbol{\pi} \leftarrow \mathsf{ZProve}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{w}) : \mathsf{ZVfy}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{\pi}) = 1\right] = 1.$$

Note that the simulation trapdoor $\boldsymbol{\tau}$ is kept secret and is not known to either prover or verifier in normal use of the NIZK argument, but it enables the simulation of proofs when we define zero-knowledge below.

Perfect Zero-Knowledge: An argument system has perfect zero-knowledge if it does not leak any information besides the truth of the instance. This is modelled a simulator that does not know the witness but has some trapdoor information that enables it to simulate proofs.

Definition 2.8. *For Arg =* $(\mathsf{ZSetup}, \mathsf{ZProve}, \mathsf{ZVfy}, \mathsf{ZSimProve})$ *an argument system, define* $\mathbf{Adv}^{zk}_{Arg,\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{zk}_{Arg,\mathcal{A}}(\lambda)] - 1$ *where the game* $\mathcal{G}^{zk}_{Arg,\mathcal{A}}$ *is defined as follows*

```
MAIN $\mathcal{G}^{zk}_{Arg,\mathcal{A}}(\lambda)$
$R \leftarrow \mathcal{R}(1^\lambda)$
$(\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R)$
$b \leftarrow \{0,1\}$
$b' \leftarrow \mathcal{A}^{P^b_{\mathbf{crs},\boldsymbol{\tau}}}(\mathbf{crs})$
return 1 if $b = b'$ and return 0 otherwise
```

```
$P^0_{\mathbf{crs},\boldsymbol{\tau}}(\boldsymbol{\phi}_i, \boldsymbol{w}_i)$
assert $(\boldsymbol{\phi}_i, \boldsymbol{w}_i) \in R$
$\boldsymbol{\pi}_i \leftarrow \mathsf{ZProve}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{w})$
return $\boldsymbol{\pi}_i$
```

```
$P^1_{\mathbf{crs},\boldsymbol{\tau}}(\boldsymbol{\phi}_i, \boldsymbol{w}_i)$
assert $(\boldsymbol{\phi}_i, \boldsymbol{w}_i) \in R$
$\boldsymbol{\pi}_i \leftarrow \mathsf{ZSimProve}(\mathbf{crs}, \boldsymbol{\tau}, \boldsymbol{\phi})$
return $\boldsymbol{\pi}_i$
```

The argument system Arg is perfectly zero knowledge if all PPT adversaries $\mathcal{A}$, $\mathbf{Adv}^{zk}_{Arg,\mathcal{A}}(\lambda) = 0$.

Computational Knowledge Soundness: An argument system is computationally knowledge sound if whenever somebody produces a valid argument it is possible to extract a valid witness from their internal data.

Definition 2.9. *For Arg =* $(\mathsf{ZSetup}, \mathsf{ZProve}, \mathsf{ZVfy}, \mathsf{ZSimProve})$ *an argument system, define* $\mathbf{Adv}^{sound}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{sound}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda)]$ *where the game* $\mathcal{G}^{sound}_{Arg,\mathcal{A},\chi_\mathcal{A}}$ *is defined as follows*

```
MAIN $\mathcal{G}^{sound}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda)$
$R \leftarrow \mathcal{R}(1^\lambda)$
$(\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R)$
$(\boldsymbol{\phi}, \boldsymbol{\pi}) \leftarrow \mathcal{A}(\mathbf{crs})$
$\boldsymbol{w} \leftarrow \chi_\mathcal{A}(\mathsf{trans}_\mathcal{A})$
assert $(\boldsymbol{\phi}, \boldsymbol{w}) \notin R$
return $\mathsf{ZVfy}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{\pi})$
```

An argument system Arg is computationally knowledge sound if for any PPT adversary $\mathcal{A}$, *there exists a PPT extractor* $\chi_\mathcal{A}$ *such that* $\mathbf{Adv}^{sound}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda) \approx 0$.

Simulation-Extractability: Zero-knowledge and soundness are core security properties of NIZK arguments. However, it is conceivable that an adversary that sees a simulated proof for a false instance might modify the proof into another proof for a false instance. This scenario is actually very common in security proofs for cryptographic schemes, so it is often desirable to have some form of non-malleability that prevents cheating in the presence of simulated proofs.

Traditionally, simulation-extractability is defined with respect to a decryption key associated with the common reference string that allows the extraction of a witness from a valid proof. However, in succinct NIZK arguments the proofs are too small to encode the full witness. We will therefore instead define simulation-extractable NIZK arguments using a non-black-box extractor that can deduce the witness from the internal data of the adversary.

Definition 2.10. *Let $Arg = (\mathsf{ZSetup}, \mathsf{ZProve}, \mathsf{ZVfy}, \mathsf{ZSimProve})$ be a NIZK argument for $\mathcal{R}$. Define $\mathbf{Adv}^{\textit{proof-ext}}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\textit{proof-ext}}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda)]$ where the game $\mathcal{G}^{\textit{proof-ext}}_{Arg,\mathcal{A},\chi_\mathcal{A}}$ is defined as follows*

```
MAIN $\mathcal{G}^{proof\text{-}ext}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda)$
$R \leftarrow \mathcal{R}(1^\lambda);\ Q = \emptyset$
$(\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R)$
$(\boldsymbol{\phi}, \boldsymbol{\pi}) \leftarrow \mathcal{A}^{\mathsf{ZSimProve}_{\mathbf{crs},\boldsymbol{\tau}}}(\mathbf{crs})$
$\boldsymbol{w} \leftarrow \chi_\mathcal{A}(\mathsf{trans}_\mathcal{A})$
return 1 if and only if
    $\mathsf{ZVfy}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{\pi}) = 1$
    $(\boldsymbol{\phi}, \boldsymbol{\pi}) \notin Q$
    $(\boldsymbol{\phi}, \boldsymbol{w}) \notin R$

$\mathsf{ZSimProve}_{\mathbf{crs},\boldsymbol{\tau}}(\boldsymbol{\phi}_i)$
$\boldsymbol{\pi}_i \leftarrow \mathsf{ZSimProve}(\mathbf{crs}, \boldsymbol{\tau}, \boldsymbol{\phi}_i)$
$Q = Q \cup \{(\boldsymbol{\phi}_i, \boldsymbol{\pi}_i)\}$
return $\boldsymbol{\pi}_i$
```

A NIZK argument Arg is simulation-extractable if for any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\chi_\mathcal{A}$ such that $\mathbf{Adv}^{\textit{proof-ext}}_{Arg,\mathcal{A},\chi_\mathcal{A}}(\lambda) \approx 0$.

We observe that simulation-extractability implies knowledge soundness, since knowledge soundness corresponds to simulation-extractability where the adversary is not allowed to use the simulation oracle.

Definition 2.11. *A* succinct *argument system is one in which the proof size is polynomial in the security parameter and the verifier's computation time is polynomial in the security parameter and the instance size.*

Terminology:

- A Succinct Non-interactive ARgument of Knowledge is a SNARK.
- A zk-SNARK is a zero-knowledge SNARK, or a succinct NIZK argument.
- A simulation-extractable NIZK argument is an SE-NIZK argument.
- A succinct SE-NIZK argument is an SE-SNARK.

Benign relation generators. Bitansky et al. [BCPR16] showed that indistinguishability obfuscation implies that there are potential auxiliary inputs to the adversary that allow it to create a valid proof in an obfuscated way such that it is impossible to extract the witness. Boyle and Pass [BP14] show that assuming the stronger notion of public coin differing input obfuscation there is even auxiliary inputs that defeat witness extraction for all candidate SNARKs. These counter examples, however, rely on specific input distributions for the adversary. We will therefore in the following assume the relationship generator is *benign* such that the relation (and the potential auxiliary inputs included in it) are distributed in such a way that the SNARKs we construct can be simulation extractable.

3 Signatures of Knowledge from SE-NIZKs

Signatures of knowledge and SE-NIZK arguments are closely related. We will now show how to construct a signature of knowledge scheme for messages in $\{0,1\}^*$ from an SE-NIZK argument. This means that in the rest of the article we can focus our efforts on constructing succinct SE-NIZK arguments, which is a slightly simpler notion than signatures of knowledge since it does not involve a message.

We will be using target collision-resistant hash functions, also known as universal one-way hash function. It is known that target collision-resistant hash functions exist if one-way functions exist [Rom90,NY90]. It is alternatively possible to use collision-resistant hash functions. This would simplify the construction since the key can be placed in the public parameters. However, by using target collision-resistant hash functions we provide greater generality.

Definition 3.1 (Target collision-resistant hash-function). *We say the polynomial time algorithm $H : \{0,1\}^{\ell_K(\lambda)} \times \{0,1\}^* \to \{0,1\}^{\ell_h(\lambda)}$, with ℓ_K, ℓ_h being polynomials in λ, is target collision-resistant if for all stateful PPT adversaries $\mathcal{A}$,*

$$\Pr[m_0 \leftarrow \mathcal{A}(1^\lambda); K \leftarrow \{0,1\}^{\ell_K(\lambda)}; m_1 \leftarrow \mathcal{A}(K) : m_0 \neq m_1 \ \wedge \ H_K(m_0) = H_K(m_1)] \approx 0.$$

Suppose $\mathcal{R}'$ is a relation generator which, on input of a security parameter λ, outputs a relation R'. We define a corresponding relation

$$R = \{((K, h, \boldsymbol{\phi}), \boldsymbol{w}) : K \in \{0,1\}^{\ell_K(\lambda)} \ \wedge \ h \in \{0,1\}^{\ell_h(\lambda)} \ \wedge \ (\boldsymbol{\phi}, \boldsymbol{w}) \in R'\}.$$

In the following, we let $\mathcal{R}$ be the relation generator that runs $R' \leftarrow \mathcal{R}'(1^\lambda)$ and returns R as defined above. Let H be a target collision-resistant hash function and $(\mathsf{ZSetup}, \mathsf{ZProve}, \mathsf{ZVfy}, \mathsf{ZSimProve})$ be a SE-NIZK argument for $\mathcal{R}$. Then Fig. 1 describes a signature of knowledge for $\mathcal{R}'$.

```
$\mathsf{SSetup}(R')$
$(\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R)$
return $\boldsymbol{pp} = \mathbf{crs}$

$\mathsf{SSign}(\boldsymbol{pp}, \boldsymbol{\phi}, \boldsymbol{w}, m)$
$K \leftarrow \{0,1\}^{\ell_K(\lambda)}$
$\boldsymbol{\pi} \leftarrow \mathsf{ZProve}(\mathbf{crs}, (K, H_K(m), \boldsymbol{\phi}), \boldsymbol{w})$
return $\boldsymbol{\sigma} = (K, \boldsymbol{\pi})$

$\mathsf{SVfy}(\boldsymbol{pp}, \boldsymbol{\phi}, m, \boldsymbol{\sigma})$
parse $\boldsymbol{\sigma} = (K, \boldsymbol{\pi})$
return $\mathsf{ZVfy}(\mathbf{crs}, (K, H_K(m), \boldsymbol{\phi}), \boldsymbol{\pi})$
```

```
$\mathsf{SSimSetup}(R')$
$(\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R)$
return $\boldsymbol{pp} = (\mathbf{crs}, \boldsymbol{\tau})$

$\mathsf{SSimSign}(\boldsymbol{pp}, \boldsymbol{\tau}, \boldsymbol{\phi}, m)$
$K \leftarrow \{0,1\}^{\ell_K(\lambda)}$
$\boldsymbol{\pi} \leftarrow \mathsf{ZSimProve}(\mathbf{crs}, \boldsymbol{\tau}, (K, H_K(m), \boldsymbol{\phi}))$
return $\boldsymbol{\sigma} = (K, \boldsymbol{\pi})$
```

Fig. 1. SoK scheme based on target collision-resistant hash-function and SE-NIZK argument.

Proposition 3.1. *If H is a target collision-resistant hash-function and Arg = $(\mathsf{ZSetup}, \mathsf{ZProve}, \mathsf{ZVfy}, \mathsf{ZSimProve})$ is an SE-NIZK argument for $\mathcal{R}$, then the scheme $(\mathsf{SSetup}, \mathsf{SSign}, \mathsf{SVfy}, \mathsf{SSimSetup}, \mathsf{SSimSign})$ given in Fig. 1 is a signature of knowledge for $\mathcal{R}'$ with respect to the message spaces $\mathcal{M}_\lambda = \{0,1\}^*$.*

Proof. We shall show that the signature of knowledge is perfectly correct, perfectly simulatable and that it is simulation extractable.

Perfect Correctness: Suppose that $\lambda \in \mathbb{N}$, $R' \in \mathcal{R}'_\lambda$, $(\boldsymbol{\phi}, \boldsymbol{w}) \in R'$ and $m \in \{0,1\}^*$. Running $\boldsymbol{pp} \leftarrow \mathsf{SSetup}(R')$, $\boldsymbol{\sigma} \leftarrow \mathsf{SSign}(\boldsymbol{pp}, \boldsymbol{\phi}, \boldsymbol{w}, m)$ and checking $\mathsf{SVfy}(\boldsymbol{pp}, \boldsymbol{\phi}, m, \boldsymbol{\sigma}) = 1$ corresponds to running $(\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R)$, $K \leftarrow \{0,1\}^{\ell_K(\lambda)}$, $\boldsymbol{\pi} \leftarrow \mathsf{ZProve}(\mathbf{crs}, (K, H_K(m), \boldsymbol{\phi}), \boldsymbol{w})$ and checking that $\mathsf{ZVfy}(\mathbf{crs}, (K, H_K(m), \boldsymbol{\phi}), \boldsymbol{\pi}) = 1$. As the NIZK argument is perfectly complete this check will always pass.

Perfect Simulatability: We show that for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\text{simul}}_{SoK,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\text{zk}}_{Arg,\mathcal{B}}(\lambda)$ for all $\lambda \in \mathbb{N}$. Since an SE-NIZK is perfectly zero-knowledge, this implies that $\mathbf{Adv}^{\text{simul}}_{SoK,\mathcal{A}}$ is negligible in λ, i.e., if $\mathcal{A}$ breaks simulatability for SoK then $\mathcal{B}$ breaks the zero-knowledge of Arg.

Let $\mathcal{A}$ be a PPT adversary against $\mathcal{G}^{\text{simul}}_{SoK,\mathcal{A}}$. Define the PPT adversary $\mathcal{B}$ that uses the output of $\mathcal{A}$ to attack zero-knowledge and behaves as follows:

```
$\mathcal{B}^{P^b_{\mathbf{crs},\tau}}(\mathbf{crs})$
$b' \leftarrow \mathcal{A}^{S^b_{\mathbf{crs},\tau}}(\mathbf{crs})$
return $b'$
```

```
$S^b_{\mathbf{crs},\boldsymbol{\tau}}(\boldsymbol{\phi}_i, \boldsymbol{w}_i, m_i)$
$K_i \leftarrow \{0,1\}^{\ell_K(\lambda)}$
$\boldsymbol{\pi}_i \leftarrow P^b_{\mathbf{crs},\boldsymbol{\tau}}((K_i, H_{K_i}(m_i), \boldsymbol{\phi}_i), \boldsymbol{w}_i)$
return $\boldsymbol{\sigma}_i = (K_i, \boldsymbol{\pi}_i)$
```

We argue that if $P^b_{\mathbf{crs},\boldsymbol{\tau}}$ is defined to be the oracles in $\mathcal{G}^{\text{zk}}_{Arg,\mathcal{B}}$ then $S^b_{\mathbf{crs},\boldsymbol{\tau}}$ behaves exactly as the oracles in $\mathcal{G}^{\text{simul}}_{\text{SoK},\mathcal{A}}$. To see this first note that if $(\boldsymbol{\phi}_i, \boldsymbol{w}_i) \notin R$ then S^b returns $\perp$. If $(\boldsymbol{\phi}_i, \boldsymbol{w}_i) \in R$ then the following holds.

- when $b = 0$, $P^b_{\mathbf{crs},\boldsymbol{\tau}}$ returns $\boldsymbol{\pi}_i \leftarrow \mathsf{ZProve}(\mathbf{crs}, (K_i, H_{K_i}(m_i), \boldsymbol{\phi}), \boldsymbol{w}_i)$. This corresponds exactly to sampling $\boldsymbol{\sigma}_i \leftarrow \mathsf{SSign}(\mathbf{crs}, \boldsymbol{\phi}, m_i, \boldsymbol{w}_i)$.
- when $b = 1$, $P^b_{\mathbf{crs},\boldsymbol{\tau}}$ returns $\boldsymbol{\pi}_i \leftarrow \mathsf{ZSimProve}(\mathbf{crs}, \boldsymbol{\tau}, (K_i, H_{K_i}(m_i), \boldsymbol{\phi}))$. This corresponds exactly to sampling $\boldsymbol{\sigma}_i \leftarrow \mathsf{SSimSign}(\mathbf{crs}, \boldsymbol{\tau}, \boldsymbol{\phi}_i, m_i)$.

Hence whenever $\mathcal{A}$ succeeds at $\mathcal{G}^{\text{simul}}_{\text{SoK},\mathcal{A}}$, $\mathcal{B}$ succeeds at $\mathcal{G}^{\text{zk}}_{Arg,\mathcal{B}}$ and the result holds.

Simulation-Extractability: We show that for all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that for all PPT extractors $\chi_\mathcal{B}$, there exists a PPT extractor $\chi_\mathcal{A}$ such that $\mathbf{Adv}^{sig-ext}_{\text{SoK},\mathcal{A},\chi_\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\text{proof-ext}}_{Arg,\mathcal{B},\chi_\mathcal{B}}(\lambda) + \mathbf{Adv}^{\text{hash}}_{\mathcal{B}}(\lambda)$ for all $\lambda \in \mathbb{N}$. By simulation-extractability of the SE-NIZK argument, we have that for any choice of $\mathcal{B}$, there exists a PPT $\chi_\mathcal{B}$ such that the above is negligible in λ, meaning that there exists a $\chi_\mathcal{A}$ such that $\mathbf{Adv}^{sig-ext}_{\text{SoK},\mathcal{A},\chi_\mathcal{A}}(\lambda)$ is negligible in λ. In other words, we construct an adversary $\mathcal{B}$ such that if $\mathcal{A}$ breaks simulation-extractability for SoK then $\mathcal{B}$ breaks simulation extractability for the argument.

Let $\mathcal{A}$ be a PPT adversary that on input of some public parameters outputs an instance, a message and a signature. Define the PPT adversary $\mathcal{B}$ that uses $\mathcal{A}$ to attack simulation-extractability of the argument and behaves as follows.

```
$\mathcal{B}^{\mathsf{ZSimProve}_{\mathbf{crs},\tau}}(\mathbf{crs})$
$Q' = \emptyset$
$(\boldsymbol{\phi}, m, \boldsymbol{\sigma}) \leftarrow \mathcal{A}^{\mathsf{SSimSign}_{\mathbf{crs},\tau}}(\mathbf{crs})$;
parse $\boldsymbol{\sigma} = (K, \boldsymbol{\phi})$
return $((K, H_K(m), \boldsymbol{\phi}), \boldsymbol{\pi})$
```

```
$\mathsf{SSimSign}_{\boldsymbol{pp},\boldsymbol{\tau}}(\boldsymbol{\phi}_i, m_i)$
$K_i \leftarrow \{0,1\}^{\ell_K(\lambda)}$
$\boldsymbol{\pi}_i \leftarrow \mathsf{ZSimProve}_{\mathbf{crs},\boldsymbol{\tau}}((K_i, H_{K_i}(m_i), \boldsymbol{\phi}_i))$
$Q' = Q' \cup \{(\boldsymbol{\phi}_i, m_i, (K_i, \boldsymbol{\pi}_i))\}$
return $\boldsymbol{\sigma}_i = (K_i, \boldsymbol{\pi}_i)$
```

Observe that $\mathsf{trans}_\mathcal{B}$ contains no information that cannot be calculated in polynomial time from $\mathsf{trans}_\mathcal{A}$. We need to design an extractor $\chi_\mathcal{A}$ that uses χ_B's output to break simulation-extractability of the argument. Let T be such that $\mathsf{trans}_\mathcal{B} = T(\mathsf{trans}_\mathcal{A})$. Let $\chi_\mathcal{B}$ be a PPT extractor that on input of $\mathsf{trans}_\mathcal{B}$ outputs some $\boldsymbol{w}$. Define $\chi_\mathcal{A}$ as follows.

```
$\underline{\chi_\mathcal{A}(\mathsf{trans}_\mathcal{A})}$
$\mathsf{trans}_\mathcal{B} \leftarrow T(\mathsf{trans}_\mathcal{A})$;
return $\chi_\mathcal{B}(\mathsf{trans}_\mathcal{B})$
```

For all PPT $\mathcal{A}$, if $\mathcal{B}$ is defined as above, then for all PPT $\chi_\mathcal{B}$, if $\chi_\mathcal{A}$ is defined as above, then $\mathcal{B}$ succeeds at $\mathcal{G}^{\text{prove-ext}}_{Arg,\mathcal{B},\chi_\mathcal{B}}$ whenever $\mathcal{A}$ succeeds at $\mathcal{G}^{\text{sig-ext}}_{\text{SoK},\mathcal{A},\chi_\mathcal{A}}$. To see this observe that

1. If $((K, h, \boldsymbol{\phi}), \boldsymbol{\pi}) \in Q$ then either $(\phi, m, \boldsymbol{\sigma} = (K, \boldsymbol{\pi})) \in Q'$ or $\mathcal{A}$ outputs some m such that $H_{K_i}(m) = H_{K_i}(m_i)$ for m_i one of the queried messages but $m \neq m_i$. The latter happens with negligible probability when H_K is target collision-resistant.
2. $(\boldsymbol{\phi}, \boldsymbol{w}) \in R' \iff ((K, H_K(m), \boldsymbol{\phi}), \boldsymbol{w}) \in R$.
3. $\mathsf{SVfy}(\boldsymbol{pp}, \boldsymbol{\phi}, m, \boldsymbol{\sigma}) = \mathsf{ZVfy}(\mathbf{crs}, (K, H_K(m), \boldsymbol{\phi}), \boldsymbol{\pi})$.

This completes the proof. □

In the other direction, it is easy to see that an SoK scheme can be used to construct an SE-NIZK argument by using a default message $m = 0$ assuming such a default message exists in the message spaces M_λ.

Proposition 3.2. *If an SoK scheme is simulation-extractably secure for a relation generator $\mathcal{R}$ then the NIZK argument for the relation generator $\mathcal{R}$ described in Figure 2 has perfect completeness, perfect zero-knowledge and is simulation-extractable.*

Proof. For correctness, observe that the perfect simulatability of the signature of knowledge implies real public parameters and simulated public parameters have identical distributions. The perfect completeness of the NIZK argument now follows from the perfect correctness of the signature of knowledge. Perfect zero-knowledge follows from perfect simulatability, and computational simulation-extractability follows from simulation-extractability of the SoK scheme. □

```
$\underline{\mathsf{ZSetup}(R)}$
$(\boldsymbol{pp}, \boldsymbol{\tau}) \leftarrow \mathsf{SSimSetup}(R)$
return $(\boldsymbol{pp}, \boldsymbol{\tau})$
```

```
$\underline{\mathsf{ZSimProve}(\boldsymbol{pp}, \boldsymbol{\tau}, \boldsymbol{\phi})}$
$\boldsymbol{\sigma} \leftarrow \mathsf{SSimSign}(\boldsymbol{pp}, \boldsymbol{\tau}, \boldsymbol{\phi}, 0)$
return $\boldsymbol{\sigma}$
```

```
$\underline{\mathsf{ZProve}(\boldsymbol{pp}, \boldsymbol{\phi}, \boldsymbol{w})}$
$\boldsymbol{\sigma} \leftarrow \mathsf{SSign}(\boldsymbol{pp}, \boldsymbol{\phi}, 0, \boldsymbol{w})$
return $\boldsymbol{\sigma}$
```

```
$\underline{\mathsf{ZVfy}(\boldsymbol{pp}, \boldsymbol{\phi}, \boldsymbol{\pi})}$
return $\mathsf{SVfy}(\mathbf{crs}, \boldsymbol{\phi}, 0, \boldsymbol{\pi})$
```

Fig. 2. SE-NIZK construction from an SoK.

4 Bilinear groups and Assumptions

Definition 4.1. *A* bilinear group generator $\mathcal{BG}$ *takes as input a security parameter in unary and returns a bilinear group* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathsf{aux})$ *consisting of cyclic groups* $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ *of prime order* p *and a* bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ *possibly together with some auxiliary information* aux *such that*

- *there are efficient algorithms for computing group operations, evaluating the bilinear map, deciding membership of the groups, and sampling generators of the groups;*
- *the map is bilinear, i.e., for all* $G \in \mathbb{G}_1$ *and* $H \in \mathbb{G}_2$ *and for all* $a, b \in \mathbb{Z}$ *we have* $e(G^a, H^b) = e(G, H)^{ab}$*;*
- *and the map is non-degenerate, i.e., if* $e(G, H) = 1$ *then* $G = 1$ *or* $H = 1$.

Usually bilinear groups are constructed from elliptic curves equipped with a pairing, which can be tweaked to yield a non-degenerate bilinear map. There are many ways to set up bilinear groups both as symmetric bilinear groups where $\mathbb{G}_1 = \mathbb{G}_2$ and as asymmetric bilinear groups where $\mathbb{G}_1 \neq \mathbb{G}_2$. We will be working in the asymmetric setting, in what Galbraith, Paterson and Smart [GPS08] call the Type III setting where there is no efficiently computable non-trivial homomorphism in either direction between $\mathbb{G}_1$ and $\mathbb{G}_2$. Type III bilinear groups are the most efficient type of bilinear groups and hence the most relevant for practical applications.

4.1 Intractability Assumptions

We will now specify the intractability assumptions used later to prove our pairing-based SE-SNARK secure.

The eXtended Power Knowledge of Exponent Assumption

Our strongest assumption is the extended power knowledge of exponent (XPKE) assumption, which is a knowledge extractor assumption. We consider an adversary that gets access to source group elements that have discrete logarithms that are polynomials evaluated on secret random variables. The assumption then says that the only way the adversary can produce group elements in the two source groups with matching discrete logarithms, i.e., $G^a \in \mathbb{G}_1$ and $H^b \in \mathbb{G}_2$ with $a = b$, is if it knows that b is the evaluation of a known linear combination of the polynomials.

Assumption 4.1 *Let* $\mathcal{A}$ *be an adversary and let* $\chi_{\mathcal{A}}$ *be an extractor. Define the advantage* $\mathbf{Adv}^{XPKE}_{\mathcal{BG},d(\lambda),q(\lambda),\mathcal{A},\chi_A}(\lambda) = \Pr[\mathcal{G}^{XPKE}_{\mathcal{BG},d(\lambda),q(\lambda),\mathcal{A},\chi_{\mathcal{A}}}(\lambda)]$ *where* $\mathcal{G}^{XPKE}_{\mathcal{BG},d(\lambda),q(\lambda),\mathcal{A},\chi_{\mathcal{A}}}$ *is defined as below and* Q_2 *is the set of polynomials* $h_j(X_1, \ldots, X_q)$ *queried to* $\mathcal{O}^2_{H,\boldsymbol{x}}$.

```
MAIN $\mathcal{G}^{XPKE}_{\mathcal{BG},d(\lambda),q(\lambda),\mathcal{A},\chi_{\mathcal{A}}}(\lambda)$
$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathsf{aux}) \leftarrow \mathcal{BG}(1^\lambda)$;
$G \leftarrow \mathbb{G}_1^*; H \leftarrow \mathbb{G}_2^*; \boldsymbol{x} \leftarrow (\mathbb{Z}_p^*)^q$
$(G^a, H^b) \leftarrow \mathcal{A}^{\mathcal{O}^1_{G,\boldsymbol{x}},\ \mathcal{O}^2_{H,\boldsymbol{x}}}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathsf{aux})$
$\boldsymbol{\eta} \in (\mathbb{Z}_p)^{|Q_2|} \leftarrow \chi_{\mathcal{A}}(\mathsf{trans}_{\mathcal{A}})$;
return 1 if $a = b$ and $b \neq \sum_{h_j \in Q_2} \eta_j h_j(\boldsymbol{x})$
else return 0
```

```
$\mathcal{O}^1_{G,\boldsymbol{x}}(g_i)$
assert $g_i \in \mathbb{Z}_p[X_1, \ldots X_q]$
assert $\deg(g_i) \leq d$
return $G^{g_i(\boldsymbol{x})}$
```

```
$\mathcal{O}^2_{H,\boldsymbol{x}}(h_j)$
assert $h_j \in \mathbb{Z}_p[X_1, \ldots X_q]$
assert $\deg(h_j) \leq d$
return $H^{h_j(\boldsymbol{x})}$
```

The $(d(\lambda), q(\lambda))$-XPKE assumption holds relative to $\mathcal{BG}$ if for all PPT adversaries $\mathcal{A}$, there exists a PPT algorithm $\chi_{\mathcal{A}}$ such that $\mathbf{Adv}^{XPKE}_{\mathcal{BG},d(\lambda),q(\lambda),\mathcal{A},\chi_{\mathcal{A}}}(\lambda)$ is negligible in λ.

The Computational Polynomial Assumption

The computational polynomial (Poly) assumption is related to the d-linear assumption of Escala, Herold, Kiltz, Ràfols and Villar [EHK+17]. In the univariate case, the Poly assumption says that for any $G \in \mathbb{G}_1^*$, given $G^{g_1(x)}, \ldots, G^{g_I(x)}$, an adversary cannot compute $G^{g(x)}$ for a polynomial g that is linearly independent from $g_1, \ldots, g_I$ - even if it knows $H^{g(x)}$ for $H \in \mathbb{G}_2^*$.

Assumption 4.2 *Let $\mathcal{A}$ be a PPT algorithm, and define the advantage $\mathbf{Adv}^{Poly}_{\mathcal{BG},d(\lambda),q(\lambda),\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{Poly}_{\mathcal{BG},d(\lambda),q(\lambda),\mathcal{A}}(\lambda)]$ where $\mathcal{G}^{Poly}_{\mathcal{BG},d(\lambda),q(\lambda),\mathcal{A}}$ is defined below and Q_1 is the set of polynomials $g_j(X_1, \ldots, X_q)$ queried to $\mathcal{O}^1_{G,\boldsymbol{x}}$.*

```
MAIN $\mathcal{G}^{Poly}_{\mathcal{BG},d(\lambda),q(\lambda),\mathcal{A}}(\lambda)$
$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathsf{aux}) \leftarrow \mathcal{BG}(1^\lambda)$;
$G \leftarrow \mathbb{G}_1^*; H \leftarrow \mathbb{G}_2^*; \boldsymbol{x} \leftarrow (\mathbb{Z}_p^*)^q$;
$(G^a, g(X_1, \ldots, X_q)) \leftarrow \mathcal{A}^{\mathcal{O}^1_{G,\boldsymbol{x}},\ \mathcal{O}^2_{H,\boldsymbol{x}}}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathsf{aux})$
return 1 if $a = g(\boldsymbol{x})$ and $g \notin span\{Q_1\}$
else return 0
```

```
$\mathcal{O}^1_{G,\boldsymbol{x}}(g_i)$
assert $g_i \in \mathbb{Z}_p[X_1, \ldots X_q]$
assert $\deg(g_i) \leq d$
return $G^{g_i(\boldsymbol{x})}$
```

```
$\mathcal{O}^2_{H,\boldsymbol{x}}(h_j)$
assert $h_j \in \mathbb{Z}_p[X_1, \ldots X_q]$
assert $\deg(h_j) \leq d$
return $H^{h_j(\boldsymbol{x})}$
```

The $(d(\lambda), q(\lambda))$-Poly assumption holds relative to $\mathcal{BG}$ if for all PPT adversaries $\mathcal{A}$ we have $\mathbf{Adv}^{Poly}_{Poly,d(\lambda),q(\lambda),\mathcal{A}}(\lambda)$ is negligible in λ.

Plausibility of the assumptions

To be plausible an assumption should not be trivial to break using generic group operations. There are various ways to formalize generic group models that restrict the adversary to such operations [Sho97,Nec94,MW98]. Using the framework from [BBG05] it is easy to show the following proposition.

Proposition 4.1. *The* $(d(\lambda), q(\lambda))$-XPKE *and* $(d(\lambda), q(\lambda))$-Poly *assumptions both hold in the generic group model.*

We construct a pairing-based SE-SNARK. The simulation-extractability property of the SE-SNARK will rely on the XPKE and Poly assumptions. However, it is instructive to consider the assumption requirements for the weaker notion of knowledge soundness of the SNARK first, since it illustrates that for zk-SNARKs our assumptions are on par with those used in previous zk-SNARKs and significantly simpler than the full generic group model.

To prove our SNARK has standard knowledge soundness, it suffices to consider the XPKE and Poly assumptions where the adversary has non-adaptive oracle access. We can reformulate this as the adversary having to specify all the polynomials it wants to query and then submitting all queries at once and gets the matching oracle responses.

The non-adaptive Poly assumption is a computational target assumption [GG17] and is implied by the q'-BGDHE$_1$ assumption for sufficiently large $q' = \text{Poly}(d, q)$, which says that given $\{G^{x^i}, H^{x^i}\}_{i=0}^{2q'} \setminus \{G^{x^{q'}}\}$, where $G \in \mathbb{G}_1$ and $H \in \mathbb{G}_2$, it is hard to compute $G^{x^{q'}}$.

The non-adaptive XPKE assumption bears resemblance to the power knowledge of exponent (PKE) assumption from [DFGK14]. The assumption ensures that if the response G^a and H^b has $a = b$ then it is beceause b is some known linear combination of the queried polynomials. In contrast, in the only previous 3 element zk-SNARK [Gro16] it is necessary in the proof of knowledge soundness to also consider elements where the exponent has a quadratic relationship to the queried polynomials.

To get simulation-extractability, we strengthen both the XPKE and Poly assumptions to make them interactive. We conjecture this is unavoidable; simulation-extractability is interactive in nature and we do not see how to base it on non-interactive assumptions.

5 SE-SNARK

We will now construct an SE-SNARK for square arithmetic program (SAP) generators, which we define below. Any arithmetic circuit over a finite field can be efficiently converted into an SAP over the same field, see Appendix A, so this gives us SE-SNARKs for arithmetic circuit satisfiability.

5.1 Our Techniques

Our SE-SNARK takes inspitation from that of Groth [**?**], which itself optimises standard techniques for building SE-SNARKs. First, there is a trusted party that outputs a relation specific CRS. Then the prover outputs an instance and a proof consisting of group elements. Then the verifier checks that the proof satisfies a pairing equation determined by the instance; the prover can only find verifying group elements if it knows a witness to the instance.

Let us provide some intuition as to why pairing-based zk-SNARKs are, typically speaking, not simulation-extractable. The problem is that an adversary that sees a proof is often able to modify it into a different proof for the same instance. Such modifications do not violate standard zk-SNARKs, however, for SE-SNARKs an adversary may request a simulated proof for a false instance, and then modify it into a different proof for the same false instance, which breaks simulation-extractability.

In the case of Groth's zk-SNARK, suppose for an instance ϕ that (A, B, C) are three group elements in a proof that satisfy the verification equations. The verifies equation are then given by

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

for a known polynomial f in ϕ and some secret α, β, δ.

There are two methods to generically randomise a proof A, B, C that satisfy (1). An adversary can either set

$$A' = A^r;\ B' = B^{\frac{1}{r}};\ C' = C$$

or they can set

$$A' = A;\ B' = Bh^{r\delta};\ C' = A^r C$$

for any field element r.

To neutralise the first attack we add the verification equation

$$e(A, H) = e(G, B).$$

However this still leaves the case where $r = -1$. So we further take the elements $G^{\alpha\delta}, H^{\beta\delta}$ into the quadratic constraint i.e. rather than using $e(A, B)$ in the verification equation we use $e(Ag^{\alpha\delta}, Bh^{\beta\delta})$.

To neutralise the second attack our CRS is designed to contain H^{δ}, $G^{\gamma\delta}$ and $H^{\gamma\delta}$ but not G^{δ}. That way, if the adversary sets $B' = BH^{r\delta}$, then the only possible value for A' is $AG^{r\delta}$ (which the adversary cannot compute). This is a simplication of the attack - in the full proof we are concerned about $B' = BH^{\psi(\tau)\delta}$ for ψ a polynomial in the trapdoor.

Groth's verifier does not require that $e(A, H) = e(G, B)$ and his prover sets A to depend on the $u_i(X)$ polynomials in a QAP and B to depend on the $v_i(X)$ polynomials[1]. We modify our construction to be built over a special case of QAP's where for all i, $u_i(X) = v_i(X)$.

5.2 Square Arithmetic Programs

Formally, we will be working with square arithmetic programs R that have the following description

$$R = \left(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \ell, \{u_i(X), w_i(X)\}_{i=0}^{m}, t(X)\right),$$

where the bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ defines the finite field $\mathbb{Z}_p$ we will be working over, $1 \leq \ell \leq m$, $u_i(X), w_i(X), t(X) \in \mathbb{Z}_p[X]$ and $u_i(X), w_i(X)$ have strictly lower degree than n, the degree of $t(X)$. Furthermore, we require that the set $S = \{u_i(X) : 0 \leq i \leq \ell\}$ is linearly independent and that any $u_i(X) \in S$ is also linearly independent from the set $\{u_j(X) : \ell < j \leq m\}$. A square arithmetic program with such a description defines the following binary relation, where we define $s_0 = 1$,

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

We say $\mathcal{R}$ is a bilinear group and square arithmetic program generator if it generates relations of the form given above with prime $p > 2^{\lambda-1}$.

[1] Perhaps as a result of this his security is proven in the generic group model and not over KOE assumptions.

5.3 Derivation of a Relation Dependent CRS

We require a reference string that depends on an SAP (we denote the SAP by sap). In Figure 4 we provide algorithm for generating the CRS and for verifying its structure. It is essential that the proof of knowledge algorithm (POK) does not reveal G^{δ} and we are implicitly assuming that the verifier checks that the elements in the proof of knowledge are consistent with the CRS.

```
$\underline{\mathsf{ZSetup}(1^{\lambda}, \mathsf{sap}) \mapsto (\mathsf{crs}_{\mathsf{sap}}, \rho):}$
$\alpha, \beta, \gamma, \delta, x \xleftarrow{\$} \mathbb{Z}_p$ such that $t(x) \neq 0$
$\mathsf{crs}_{\mathsf{sap}} \leftarrow \left( \begin{array}{l} G^{\alpha}, G^{\gamma}, G^{x}, G^{\alpha\delta}, G^{\gamma\delta t(x)}, G^{\gamma^2\delta t^2(x)}, G^{(\alpha+\beta)\gamma\delta t(x)}, \\ H, H^{\beta}, H^{\delta}, H^{\beta\delta}, H^{\gamma\delta t(x)}, H^{\delta^2} \\ \left\{ G^{\gamma\delta x^i}, H^{\gamma\delta x^i}, G^{\gamma^2\delta t(x) x^i} \right\}_{i=0}^{n-1}, \left\{ G^{\gamma\delta w_i(x) + \delta(\alpha+\beta)u_i(x))} \right\}_{i=0}^{\ell}, \\ \left\{ G^{\gamma^2\delta w_i(x) + \gamma\delta(\alpha+\beta)u_i(x))} \right\}_{i=\ell+1}^{m}, \end{array} \right)$
$\rho \xleftarrow{\$} \mathsf{POK}\left\{ G^{\alpha}, H^{\beta}, G^{\gamma}, H^{\delta}, G^{x} : \quad \alpha, \beta, \gamma, \delta, x \right\}$
return $(\mathsf{crs}, \rho)$

$\underline{\mathsf{UpdateVerify}(1^{\lambda}, \mathsf{sap}, \mathsf{crs}_{\mathsf{sap}}, \rho) \mapsto 0/1}$
check $\mathsf{POKVerify}(\rho)$

$\left( \begin{array}{l} G^{\alpha}, G^{\gamma}, G^{x}, G^{\alpha\delta}, G^{\gamma\delta t(x)}, G^{\gamma^2\delta t^2(x)}, G^{(\alpha+\beta)\gamma\delta t(x)}, \\ H, H^{\beta}, H^{\delta}, H^{\beta\delta}, H^{\gamma\delta t(x)}, H^{\delta^2} \\ \left\{ G^{\gamma\delta x^i}, H^{\gamma\delta x^i}, G^{\gamma^2\delta t(x) x^i} \right\}_{i=0}^{n-1}, \left\{ G^{\gamma\delta w_i(x) + \delta(\alpha+\beta)u_i(x))} \right\}_{i=0}^{\ell}, \\ \left\{ G^{\gamma^2\delta w_i(x) + \gamma\delta(\alpha+\beta)u_i(x))} \right\}_{i=\ell+1}^{m}, \end{array} \right) \leftarrow \mathsf{Parse}(\mathsf{crs}_{\mathsf{sap}})$

check that the following hold:

    $e(G^{\alpha\delta}, H) = e(G^{\alpha}, H^{\delta})$                 $e(G^{\gamma\delta}, H^{\beta}) = e(G^{\gamma}, H^{\beta\delta})$
    $e(G^{\gamma\delta}, H) = e(G^{\gamma}, H^{\delta})$                 $e(G^{\gamma\delta}, H^{\delta}) = e(G^{\gamma}, H^{\delta^2})$
    $e(G^{\gamma\delta t(x)}, H^{\gamma\delta}) = e(G^{\frac{\gamma\delta(t(x)-t_0)}{x}\gamma\delta}, H^{\gamma\delta x}) e(G^{-t_0\gamma\delta}, H^{\gamma\delta})$   $e(G^{\gamma\delta t(x)}, H^{\gamma\delta}) = e(G^{\gamma\delta}, H^{\gamma\delta t(x)})$
    $e(G^{\gamma^2\delta t(x)^2}, H^{\delta}) = e(G^{\gamma\delta t(x)}, H^{\gamma\delta t(x)})$   $e(G^{(\alpha+\beta)\gamma\delta t(x)}, H^{\delta}) = e(G^{\alpha\delta}, H^{\gamma\delta t(x)}) e(G^{\gamma\delta t(x)}, H^{\beta\delta})$
    $e(G^{\gamma}, H^{\delta}) = e(G, H^{\gamma\delta})$

    for $0 \leq i \leq n-2$ :
        $e(G^{x}, H^{\gamma x^i}) = e(G, H^{\gamma x^{i+1}})$ and $e(G^{\gamma\delta x^i}, H^{\gamma\delta x}) = e(G^{\gamma\delta x^{i+1}}, H^{\gamma\delta})$

    for $1 \leq i \leq n-1$ :
        $e(G^{\gamma^2\delta t(x) x^i}, H^{\gamma\delta}) = e(G^{\gamma\delta t(x)}, H^{\gamma\delta x^i})$

    for $0 \leq i \leq \ell$ :
        $e(G^{\gamma\delta w_i(x) + \delta(\alpha+\beta)u_i(x)}, H^{\gamma\delta}) = e(G^{\gamma\delta}, H^{\gamma\delta w_i(x)}) e(G^{\alpha\delta}, H^{\gamma\delta u_i(x)}) e(G^{\gamma\delta u_i(x)}, H^{\beta\delta})$

    for $\ell + 1 \leq i \leq m$ :
        $e(G^{\gamma^2\delta w_i(x) + \gamma\delta(\alpha+\beta)u_i(x)}, H^{\delta}) = e(G^{\gamma\delta}, H^{\gamma\delta w_i(x)}) e(G^{\alpha\delta}, H^{\gamma\delta u_i(x)}) e(G^{\gamma\delta u_i(x)}, H^{\beta\delta})$

return 1 if all checks pass, else return 0
```

Fig. 3. Algorithm for verifying the structure of the crs in our SE-SNARK construction

5.4 Our Construction

Our construction of a simulation-extractable SNARK is given in Figure 4. The prover parses the wires of the circuit as $(1, a_1, \ldots, a_m)$, and then embeds the SAP polynomials $a_i u_i(X)$ evaluated at the unknown point x into one proof element in the first source group and one proof element in the second source group. They provide a third proof element whose exponent is the product of the exponents of the first two proof elements. The verifier checks with a pairing that the first and second proof elements share common exponents. They then check, also with a pairing, that the third proof element's exponent is indeed the product of the exponents of the first two proof elements.

```
Common Input: info = $\mathsf{bp}, \mathsf{sap}, \mathsf{crs}_{\mathsf{sap}}, e(G^{\alpha\delta}, H^{\beta\delta})$

ZProve(info, $\phi, w$) $\mapsto \pi$:
$(a_0, a_1, \ldots, a_m) \leftarrow \mathsf{Parse}(1, \phi, w)$
$h(X) \leftarrow \frac{\left(\sum_{i=0}^{m} a_i u_i(X)\right)^2 - \sum_{i=0}^{m} a_i w_i(X)}{t(X)}$
$A \leftarrow G^{\gamma\delta\left(rt(x)+\sum_{i=0}^{m} a_i u_i(x)\right)}$
$B \leftarrow H^{\gamma\delta\left(rt(x)+\sum_{i=0}^{m} a_i u_i(x)\right)}$
$C \leftarrow G^{\gamma\delta\left(\sum_{i=l+1}^{m} a_i(\gamma w_i(x)+(\alpha+\beta)u_i(x))+r(\alpha+\beta)t(x)+\gamma t(x)\left[r^2 t(x)+h(x)+2r\sum_{i=0}^{m} a_i u_i(x)\right]\right)}$
return $(A, B, C)$

Verify(info, $\phi, \pi$) $\mapsto 0/1$:
$(a_0, a_1, \ldots, a_\ell) \in \mathbb{Z}_p^\ell \leftarrow \mathsf{Parse}(1, \phi)$
$(A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \leftarrow \mathsf{Parse}(\pi)$
check $e(Ag^{\alpha\delta}, Bh^{\beta\delta}) = e(G^{\alpha\delta}, H^{\beta\delta})e(G^{\sum_{i=0}^{\ell} a_i\delta(\gamma w_i(x)+(\alpha+\beta)u_i(x))}, H^{\gamma\delta})e(C, H^{\delta})$
check $e(A, H) = e(G, B)$
return 1 if both checks pass, else return 0.
```

Fig. 4. Our construction of a Simulation-Extractable SNARK.

5.5 Security Proof

Theorem 5.1. *The construction in Figure 4 has subversion zero-knowledge.*

Proof. To prove subversion zero-knowledge, we need to both show the existence of a an extractor $\chi_{\mathcal{A}}$ that can compute a trapdoor from the reference string, and describe a $\mathsf{ZSimProve}$ algorithm that produced indistinguishable proofs when provided with the extracted trapdoor. It can be seen that a adversary that outputs a verifying reference string must know $\alpha, \beta, \gamma, \delta, x$ in the exponents of $G^\alpha, H^\beta, G^\gamma, H^\delta, G^x$ because the proof of knowledge verifies. Furthermore, if each of the verifier's pairing checks verify then the adversary's outputted reference string has the same structure as one output by the setup algorithm. So the extracted τ is a valid trapdoor for the reference string.

A simulator is given a trapdoor $\tau = (\alpha, \beta, \gamma, \delta, x)$ and behaves as follows.

```
$\underline{\mathsf{ZSimProve}(\mathsf{bp}, \mathsf{sap}, \mathsf{crs}_{\mathsf{sap}}, \tau, \phi) \mapsto \pi}$
$(a_0, a_1, \ldots, a_\ell) \leftarrow \mathsf{Parse}(1, \phi)$
$\mu \overset{\$}{\leftarrow} \mathbb{Z}_p$
$A, B \leftarrow G^{\mu\delta}, H^{\mu\delta}$
$C \leftarrow G^{\left(\mu^2\delta + (\alpha+\beta)\mu\delta - \gamma\delta \sum_{i=0}^{\ell} a_i(\gamma w_i(x) + (\alpha+\beta)u_i(x))\right)}$
return $(A, B, C)$
```

To see that the simulated proofs are indistinguishable from the real proofs, first observe that the simulation procedure always produces verifying proofs. Next, observe that for a given instance and proof $\pi = (A, B, C)$ the element A uniquely determines B through the second verification equation, and the elements A, B uniquely determine C through the first verification equation. In a real proof the random choice of r makes A uniformly random, and in a simulated proof the random choice of μ makes A uniformly random. So in both cases, we get the same probability distribution over proofs with uniformly random A and the unique matching B, C.

Theorem 5.2. *The protocol in Figure 4 is simulation-extractable (implying it is knowledge sound) provided that the $(n, q+5)$-$\mathsf{XPKE}(\lambda)$ and $(n, q+5)$-$\mathsf{Poly}(\lambda)$ assumptions hold, where n is the number of squaring constraints and q the number of simulation queries the adversary asks.*

Proof. Suppose that an adversary $\mathcal{A}$ is given an crs. It accesses its simulation oracle on the instances $(\phi_1, \ldots, \phi_q)$ to obtain the responses $(\pi_1, \ldots, \pi_q)$. We show that if $\mathcal{A}$ outputs verifying (ϕ, π) then either (ϕ, π) is one of the oracle queries and responses (ϕ_j, π_j) or there exists an extractor $\chi_{\mathcal{A}}$ that outputs w such that $(\phi, w) \in R$.

From the second verification equation we have that $e(A, H) = e(G, B)$. From the $(n, q+5)$-XPKE assumption there exists an extractor that outputs

$$(\eta_0, \eta_\beta, \eta_\delta, \eta_{\beta\delta}, \eta_{\delta^2}, \eta_{\gamma,\delta,t}, \eta_{\gamma\delta}(X), \eta_{b,j})$$

such that

$$\log(B) = \eta_0 + \eta_\beta\beta + \eta_\delta\delta + \eta_{\beta\delta}\beta\delta + \eta_{\delta^2}\delta^2 + \eta_{\gamma\delta t}\gamma\delta t(x) + \gamma\delta\eta_{\gamma\delta}(x) + \delta\sum_j \eta_{b,j}\mu_j.$$

Taking the adversary and the extractor together, we can see them as a combined algorithm that outputs A, B, C and the formal polynomial $\eta(X, X_\beta, X_\gamma, X_x, X_\delta, X_{\mu_1}, \ldots, X_{\mu_q})$ such that $A = G^{\eta(x,\beta,\gamma,\delta,\mu_1,\ldots,\mu_q)}$. By the $(n, q+5)$-Poly assumption this has negligible probability unless η is in the span of

$$\begin{gathered} X,\ X_\alpha,\ X_\gamma,\ X_x,\ X_\alpha X_\delta,\ X_\gamma X_\delta t(X),\ X_\gamma X_\delta (t(X))^2,\ (X_\alpha + X_\beta)X_\gamma X_\delta t(X), \\ \left\{X_\gamma X_\delta X^i,\ X_\gamma^2 X_\delta t(X) X^i\right\}_{i=0}^{n-1}, \{X_\gamma X_\delta w_i(X) + X_\delta(X_\alpha + X_\beta)u_i(X)\}_{i=0}^{\ell}, \\ \left\{X_\gamma^2 X_\delta w_i(X) + X_\gamma X_\delta (X_\alpha + X_\beta) u_i(X)\right\}_{i=\ell+1}^{m}, \ \left\{X_{\mu_j} X_\delta\right\}_{j=1}^{q} \\ \left\{\left(X_{\mu_j}^2 X_\delta + (X_\alpha + X_\beta)X_{\mu_j}X_\delta - X_\gamma X_\delta \textstyle\sum_{i=0}^{\ell} a_{\mu_j,i}(X_\gamma w_i(X) + (X_\alpha + X_\beta)u_i(X))\right)\right\}_{j=1}^{q}. \end{gathered} \tag{2}$$

This means that

$$\eta(x, \beta, \gamma, \delta, \mu_1, \ldots, \mu_q) = \eta_0 + \eta_{\gamma\delta t}\gamma\delta t(x) + \gamma\delta\eta_{\gamma\delta}(x) + \delta\sum \eta_{b,j}\mu_j.$$

From the first verification equation we get that $C = G^{f(x,\beta,\gamma,\delta,\mu_1,\dots,\mu_q)}$ where f is given by

$$\frac{1}{\delta}\left(\eta(x,\beta,\gamma,\delta,\mu_1,\dots,\mu_q)+\alpha\delta\right)\cdot\left(\eta(x,\beta,\gamma,\delta,\mu_1,\dots,\mu_q)+\beta\delta\right) \\ -\alpha\beta\delta-\sum_{i=0}^{\ell}a_i\gamma\delta(\gamma w_i(x)+(\alpha+\beta)u_i(x))$$

By the $(n,q+5)$-Poly assumption this means that

$$\frac{1}{X_\delta}\left(\eta(X,X_\beta,X_\gamma,X_\delta,X_{\mu_1},\dots,X_{\mu_q})+X_\alpha X_\delta\right)\cdot\left(\eta(X,X_\beta,X_\gamma,X_\delta,X_{\mu_1},\dots,X_{\mu_q})+X_\beta X_\delta\right) \\ -X_\alpha X_\beta X_\delta-\sum_{i=0}^{\ell}a_iX_\gamma X_\delta(X_\gamma w_i(X)+(X_\alpha+X_\beta)u_i(X))$$

also belongs to the span in (2).

The span has no polynomials of the form $\frac{1}{\delta}$ thus $\eta_0 = 0$. The span has no polynomials of the form $X_\delta X_{\mu_j} X_{\mu_k}$ for $j \neq k$ thus at most one $\eta_{b,j}$ is uncancelled. Suppose without loss of generality that $\eta_{b,j}$ are cancelled for $j \geq 2$ and rename $\eta_{b,1}$ by η_μ and μ_1 by μ.

We are now left with

$$\delta\left(\eta_{\gamma\delta t}\gamma t(x)+\gamma\eta_{\gamma\delta}(x)+\eta_\mu\mu+\alpha\right)\cdot\left(\eta_{\gamma\delta t}\gamma t(x)+\gamma\eta_{\gamma\delta}(x)+\eta_\mu\mu+\beta\right) \\ =\alpha\beta\delta+\sum_{i=0}^{\ell}a_i\gamma\delta(\gamma w_i(x)+(\alpha+\beta)u_i(x))+\log(C).$$

If $\eta_\mu \neq 0$ then $\eta_{\gamma,\delta,t}$ and $\eta_{\gamma,\delta}(X)$ both cancel because the span has no polynomials of the form $X_{\mu_{b,1}}X_\gamma X_\delta$. As a result, the polynomial $f(x,\beta,\gamma,\delta,\mu_1,\dots,\mu_q)$ extracted from C is given by

$$(\eta_\mu\mu)^2+\eta_\mu\mu(\alpha+\beta)-\sum_{i=0}^{\ell}a_i\gamma\delta(\gamma w_i(x)+(\alpha+\beta)u_i(x)).$$

The only way to obtain the $(\eta_\mu\mu)^2$ term and the $\eta_\mu\mu(\alpha+\beta)$ term is if f contains a non-trivial linear combination of the term

$$\left(X_{\mu_1}^2X_\delta+(X_\alpha+X_\beta)X_{\mu_1}X_\delta-X_\gamma X_\delta\sum_{i=0}^{\ell}a_{\mu_1,i}(X_\gamma w_i(X)+(X_\alpha+X_\beta)u_i(X))\right).$$

Thus $\eta_\mu^2 = \eta_\mu$ i.e. $\eta_{b,1} = 1$ and f contains exactly one of the above term. There are no polynomials in the span that can be used to balance $\gamma\delta(\alpha+\beta)u_i(x)$ because: there are no $X_\beta X_\gamma X_\delta X^i$ terms in $\nu(\boldsymbol{X})$; $t(X)$ has degree n which is strictly greater than the degree of the other polynomials; and the set $S = \{u_i(X) : 0 \leq i \leq \ell\}$ is linearly independent from the set $\{u_j(X) : \ell < j \leq m\}$. Hence $a_{\mu_1,i} = a_i$, i.e. $(\phi,\pi) = (\phi_1,\pi_1)$ and the adversary has regurgitated a simulated proof.

We are now left with

$$\delta\left(\eta_{\gamma\delta t}\gamma t(x)+\gamma\eta_{\gamma\delta}(x)+\alpha\right)\cdot\left(\eta_{\gamma\delta t}\gamma t(x)+\gamma\eta_{\gamma\delta}(x)+\beta\right) \\ =\alpha\beta\delta+\sum_{i=0}^{\ell}a_i\gamma\delta(\gamma w_i(x)+(\alpha+\beta)u_i(x))+\log(C).$$

Looking at the terms involving α, we get that

$$\eta_{\gamma\delta t}\alpha\gamma\delta t(x)+\alpha\gamma\delta\eta_{\gamma\delta}(x)=\sum_{i=0}^{\ell}a_i\gamma\delta\alpha u_i(x)+\sum_{i=\ell+1}^{m}a_i\gamma\delta\alpha u_i(x)+a_{(\alpha+\beta)\gamma\delta t}\alpha\gamma\delta t(x)$$

where $a_{\ell+1},\dots,a_m$ are the coefficients in f relating to the terms

$$\left\{X_\gamma^2X_\delta w_i(X)+X_\gamma X_\delta(X_\alpha+X_\beta)u_i(X)\right\}_{i=\ell+1}^{m}$$

and $a_{(\alpha+\beta)\gamma\delta t}$ relates to the term $(X_\alpha+X_\beta)X_\gamma X_\delta t(X)$. We see that $\eta_{\gamma\delta t}=a_{(\alpha+\beta)\gamma\delta t}$ because the degree of $t(X)$ is strictly greater than the degree of the other polynomials.

Looking at the terms involving $\gamma^2\delta$, we get that

$$\eta_{\gamma\delta t}\gamma^2\delta t^2(x)+\gamma^2\delta\eta_{\gamma\delta}^2(x)=\sum_{i=0}^{\ell}a_i\gamma^2\delta w_i(x)+\sum_{i=\ell+1}^{m}a_i\gamma^2\delta w_i(X)+\gamma^2\delta h(x)t(x)+a_{\gamma^2\delta t}t^2(X).$$

for some polynomial $h(X)$ relating to the coefficients in f that refer to the terms $X_\gamma^2X_\delta t(X)X^i$ and where $a_{\gamma^2\delta t}$ relates to the term $X_\gamma^2X_\delta t^2(X)$. We see that $\eta_{\gamma\delta t}=a_{\gamma^2\delta t}$ because the degree of $t^2(X)$ is strictly greater than the degree of the other polynomials.

Putting these two expressions for $\eta_{\gamma\delta}(X)$ together gives us that

$$\left(\sum_{i=0}^{m}a_iu_i(X)\right)^2=\sum_{i=0}^{m}a_iw_i(X)+h(X)t(X)$$

which gives us that $a_{\ell+1},\dots,a_m$ is a valid witness for ϕ, completing our proof.

5.6 Efficiency

The proof size is 2 elements in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$. The reference string contains $m+2n+5$ elements in $\mathbb{G}_1$ and $n+3$ elements in $\mathbb{G}_2$.

The verifier can work with a reduced reference string that only contains $\ell+2$ elements from $\mathbb{G}_1$, 3 elements from $\mathbb{G}_2$, and 1 element from $\mathbb{G}_T$ of the form

$$\left(G^{\alpha\delta},H^{\delta},H^{\beta\delta},H^{\gamma\delta},\{G^{\delta(\gamma w_i(x)+(\alpha+\beta)u_i(x))}\}_{i=0}^{\ell},e(G^{\alpha\delta},H^{\beta\delta})\right).$$

The verification consists of checking that the proof contains 3 appropriate group elements and checking 2 pairing product equations. The verifier's computation is dominated by a multi-exponentiation $\mathbb{G}_1$ to ℓ exponents (noting that $a_0=1$) and 5 pairings (assuming $e(G^\alpha,H^\beta)$ is precomputed).

The prover has to compute the polynomial $h(X)$ and it depends on the relation how long this computation takes. If we construct the SAP from an arithmetic circuit where each multiplication gate connects to a constant number of wires, there is a set of distinct points $r_1,\dots,r_n$ where the polynomials are non-zero only in a few places. In this case we can use fast polynomial manipulation techniques to compute $h(X)$ in $\tilde{O}(n)$ operations in $\mathbb{Z}_p$. The prover also computes the coefficients of $\sum_{i=0}^{m}a_iu_i(X)$, which again can be done in $\tilde{O}(n)$ operations in $\mathbb{Z}_p$ for polynomials arising from arithmetic circuits where each multiplication gate connects to a constant number of wires. Having all the coefficients of relevant polynomials, the prover's cost is dominated by $m+2n-\ell$ exponentiations in $\mathbb{G}_1$ and n exponentiations in $\mathbb{G}_2$.

6 Lower Bounds

We now show that our pairing-based simulation-extractable SNARK construction in Section 5.1 is optimal in the number of group elements and verification equations. First, we prove that it is impossible to have a pairing-based SE-NIZK argument with just one verification equation. Afterwards, we prove that it is impossible to have a pairing-based SE-NIZK argument with just two group elements. Consequently, it is impossible to have a pairing-based SE-NIZK argument or SoK with one verification equation or with 2 group elements. This stands in contrast to standard knowledge sound NIZK arguments, for which there are constructions consisting of just one verification equation.

6.1 Pairing-Based NIZK Arguments

We will in this subsection define pairing-based arguments. Before giving the definition, let us introduce some useful notation. For a row vector $\mathbf{a} = (a_1, \ldots, a_m) \in \mathbb{Z}_p^m$ and a group element G we define $G^{\mathbf{a}} = (G_1^{a_1}, \ldots, G^{a_m})$. For a matrix $\hat{B} \in \mathbb{Z}_p^{m \times n}$ we define $(G^{\mathbf{a}})^{\hat{B}} = G^{\mathbf{a}\hat{B}}$, which can be computed using generic group operations. For two vectors $\mathbf{G} = (G_1, \ldots, G_n)$ and $\mathbf{H} = (H_1, \ldots, H_n)$ we define $e(\mathbf{G}, \mathbf{H}) = \prod_{i=1}^{n} e(G_i, H_i)$. Following Groth [Gro16] quite closely we say an argument is pairing-based if it works as follows:

$(\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R)$**:** The relation contains a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. The setup algorithm runs an internal algorithm $(G, H, \mathbf{u}, \mathbf{v}, \mathbf{z}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}'(R)$ yielding $G \in \mathbb{G}_1^*, H \in \mathbb{G}_2^*$ and $(\mathbf{u}, \mathbf{v}, \mathbf{z}) \in \mathbb{Z}_p^{m_1} \times \mathbb{Z}_p^{m_2} \times \mathbb{Z}_p^{m_T}$. It returns simulation trapdoor $\boldsymbol{\tau}$ and the common reference string

$$\mathbf{crs} = (R, G^{\mathbf{u}}, H^{\mathbf{v}}, e(G, H)^{\mathbf{z}}).$$

We can without loss of generality assume $u_1 = v_1 = 1$ such that G, H are included in the CRS, and will in the following use them as the base when computing discrete logarithms.

$\pi \leftarrow \mathsf{ZProve}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{w})$**:** The prover runs an internal algorithm $(\hat{P}, \hat{Q}, \hat{T}) \leftarrow \mathsf{ZProve}'(R, \boldsymbol{\phi}, \boldsymbol{w})$ to get matrices $\hat{P} \in \mathbb{Z}_p^{m_1 \times n_1}, \hat{Q} \in \mathbb{Z}_p^{m_2 \times n_2}$ and $\hat{T} \in \mathbb{Z}_p^{m_T \times n_T}$. The prover returns the proof

$$\pi = (G^{\mathbf{u}\hat{P}}, H^{\mathbf{v}\hat{Q}}, e(G, H)^{\mathbf{z}\hat{T}}).$$

$0/1 \leftarrow \mathsf{ZVfy}(\mathbf{crs}, \boldsymbol{\phi}, \pi)$**:** The verifier runs an internal algorithm $\{(\hat{A}_i, \hat{B}_i, \hat{C}_i, \hat{D}_i, \hat{E}_i, \hat{F}_i)\}_{i=1}^{\eta} \leftarrow \mathsf{ZVfy}'(R, \boldsymbol{\phi})$ giving matrices $\hat{A}_i \in \mathbb{Z}_p^{n_1 \times n_2}, \hat{B}_i \in \mathbb{Z}_p^{n_1 \times m_2}, \hat{C}_i \in \mathbb{Z}_p^{m_1 \times n_2}, \hat{D}_i \in \mathbb{Z}_p^{m_1 \times m_2}, \hat{E}_i \in \mathbb{Z}_p^{n_T \times 1}, \hat{F}_i \in \mathbb{Z}_p^{m_T \times 1}$. The verifier then asserts $\pi = (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2, \boldsymbol{\Pi}_T) \in \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2} \times \mathbb{G}_T^{n_T}$ and checks the verification equations

$$e(\boldsymbol{\Pi}_1^{\hat{A}_i}, \boldsymbol{\Pi}_2) \cdot e(\boldsymbol{\Pi}_1^{\hat{B}_i}, H^{\mathbf{v}}) \cdot e((G^{\mathbf{u}})^{\hat{C}_i}, \boldsymbol{\Pi}_2) \cdot e((G^{\mathbf{u}})^{\hat{D}_i}, H^{\mathbf{v}}) = \boldsymbol{\Pi}_T^{\hat{E}_i} \cdot (e(G, H)^{\mathbf{z}})^{\hat{F}_i}$$

for $i = 1, \ldots, \eta$.

The dimensions $m_1, m_2, m_T, n_1, n_2, n_T$ and η are constants implicitly determined by the relation R in the argument system.

Our definition of pairing-based arguments captures all existing pre-processing SNARKs based on Type III pairings. Essentially, the definition captures that using generic group operations we should expect common reference strings that are computed as group elements raised to known exponents, proofs should be generated by computing known linear combinations of group elements in the common reference string, and verification of proofs should be done by multiplying and pairing group elements in the common reference string and in the proof. Bitansky et al. [BCI$^+$13] abstracted the action taking place in the exponents as linear interactive proofs, and for our concrete Type III pairing setting we can think of the definition above as saying the pairing-based argument is built by executing a split non-interactive linear interactive proof [Gro16] in the exponent.

6.2 Disclosure-Freeness

We want to avoid that the prover can learn non-trivial information about the discrete logarithms in the common reference string using generic bilinear group operations. An example of such a pathological case is a common reference string with group elements G, G^b, where b is a bit. The prover can easily recover the bit b by guessing it and verifying the guess with generic group operations. This would make it possible to embed and subliminally communicate a non-pairing-based common reference string to the prover, who could then proceed in a non-pairing way to construct a proof. We therefore restrict the argument to being disclosure-free, which means that a given pairing product equation will always evaluate to the same over a randomly sampled common reference string, which in turn means that it is not leaking non-trivial information about the discrete logarithms.

Definition 6.1. *We say a pairing-based argument is disclosure-free if for all adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{c} R \leftarrow \mathcal{R}(\lambda);\ (R, G^{\mathbf{u}}, H^{\mathbf{v}}, e(G,H)^{\mathbf{z}}) \leftarrow \mathsf{ZSetup}(R); \\ (R, G^{\mathbf{u}'}, H^{\mathbf{v}'}, e(G,H)^{\mathbf{z}'}) \leftarrow \mathsf{ZSetup}(R);\ (\hat{P}, \hat{T}) \leftarrow \mathcal{A}(R)\ : \\ \\ e(G^{\mathbf{u}\hat{P}}, H^{\mathbf{v}}) = e(G,H)^{\mathbf{z}\hat{T}} \text{ and } e(G^{(\mathbf{u}')\hat{P}}, H^{\mathbf{v}'}) \neq e(G,H)^{(\mathbf{z}')\hat{T}} \end{array}\right] \approx 1.$$

6.3 Lower Bounds for Pairing-Based Simulation-Sound NIZK Arguments

Here we prove that any simulation-extractable NIZK argument must have at least 2 verification equations and 3 group elements. Our theorems extend to the weaker properties of computational completeness (an honest prover can convince an honest verifier with all but negligible probability), computational zero-knowledge (a PPT adversary has negligible advantage in discerning a real proof from a simulated one) and simulation soundness. Simulation-soundness is akin to simulation-extractability but weaker in not being a proof of knowledge. In simulation-soundness the adversary has access to a simulation oracle but wins by returning a false instance and an accepting proof for it, under the condition that the instance-proof pair is not one of the simulations. We call a simulation-sound NIZK argument an SS-NIZK argument.

Theorem 6.1. *If* $(\mathsf{ZSetup}, \mathsf{ZProve}, \mathsf{ZVfy}, \mathsf{ZSimProve})$ *is a disclosure-free pairing-based SS-NIZK argument for relation generator $\mathcal{R}$ with a hard decisional problem* $(\mathsf{Yes}, \mathsf{No})$*, then it must have at least 2 verification equations and at least 3 group elements in the proofs.*

Proof. As in Groth [Gro16] it can be shown that the verification equations cannot be linear. More precisely, in a disclosure-free pairing-based NIZK argument we must have negligible probability of $\hat{A}_1 = \cdots = \hat{A}_\eta = 0$ when ϕ is generated by Yes or No. Groth observed that this means $n_1 \geq 1$ and $n_2 \geq 1$ since at least one pairing of source group elements in the proof must take place in the verification equations

Assuming without loss of generality $\hat{A}_1 \neq 0$ we now get from Theorem 6.2 that there cannot be a single verification equation, i.e., $\eta \neq 1$. Moreover, again assuming $\hat{A}_1 \neq 0$, Theorem 6.3 then shows that it is impossible to have $n_1 = n_2 = 1$ and $n_T = 0$ for a simulation-sound NIZK argument. □

6.4 Number of Verification Equations

We will now prove that there is no pairing-based SS-NIZK argument with a single verification equation, i.e. $\eta = 1$. The idea is that a single verification equation corresponds to a quadratic equation in the discrete logarithms of the group elements of the proof. The adversary will simulate to get a proof, and then try to find a second solution to the quadratic equation. If the instance is true, then usually this is possible. If the instance is false, then by simulation-soundness this should be impossible. To formalize this idea, we first write what a generic verification equation has to look like. We then translate this into a corresponding matrix equation in the discrete logarithms and use techniques from linear algebra to find a second solution.

Theorem 6.2. *If* $(\mathsf{ZSetup}, \mathsf{ZProve}, \mathsf{ZVfy}, \mathsf{ZSimProve})$ *is a disclosure-free pairing-based SS-NIZK argument for* $\mathcal{R}$ *with a hard decision problem* $\mathsf{Yes}, \mathsf{No}$*, then the verification procedure must use at least* 2 *verification equations.*

Proof. We show the contrapositive, i.e. that if $(\mathsf{ZSetup}, \mathsf{ZProve}, \mathsf{ZVfy}, \mathsf{ZSimProve})$ is an SS-NIZK argument for the relation generator $\mathcal{R}$ where ZVfy generates a single verification equation, then $\mathsf{Yes}, \mathsf{No}$ can be efficiently distinguished. To achieve this, we exploit the form of the pairing-based verification equation in order to construct a PPT adversary $\mathcal{A}$ that can construct new, valid proofs from old simulated proofs. Whenever $\boldsymbol{\phi} \in L_R$, $\mathcal{A}$ outputs a different verifying proof for $\boldsymbol{\phi}$. Whenever $\phi \notin L_R$, by simulation-soundness $\mathcal{A}$ cannot output a (new) verifying proof for $\boldsymbol{\phi}$. We can then use $\mathcal{A}$ to test membership in the language.

Form of pairing-based verification equations:
Suppose that $(\mathsf{ZSetup}, \mathsf{ZProve}, \mathsf{ZVfy}, \mathsf{ZSimProve})$ is a pairing-based SS-NIZK argument for $\mathcal{R}$ such that ZVfy consists of just one verification equation. First, observe that if there is a proof component $\boldsymbol{\Pi}_T$ in the target group that is used in a non-trivial way, then it is trivial to satisfy the verification equation. By soundness of the argument system, we must therefore have with overwhelming probability over the choice of $R \leftarrow \mathcal{R}(1^\lambda)$ that $m_T = 0$.

The CRS's consist of the relation including the bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ as well as group elements $\{\{G^{\mathbf{u}}\}, \{H^{\mathbf{v}}\}, \{e(G, H)^{\mathbf{z}}\}\}$ for some $G \in \mathbb{G}_1^*$, $H \in \mathbb{G}_2^*$, and $\mathbf{u}$, $\mathbf{v}$, $\mathbf{z}$ vectors with entries in $\mathbb{Z}_p$. The algorithms ZProve and $\mathsf{ZSimProve}$ output proof vectors $\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2$ where $\boldsymbol{\Pi}_i$ has entries in $\mathbb{G}_i$.

The verification equation can be rewritten

$$e(\boldsymbol{\Pi}_1^{\hat{A}}, \boldsymbol{\Pi}_2) \cdot e(\boldsymbol{\Pi}_1^{\hat{B}}, H^{\mathbf{v}}) \cdot e((G^{\mathbf{u}})^{\hat{C}}, \boldsymbol{\Pi}_2) = Z \tag{3}$$

where $Z = e((G^{\mathbf{u}})^{-\hat{D}}, H^{\mathbf{v}}) \cdot (e(G, H)^{\mathbf{z}})^{\hat{F}}$.

Linear algebra trick:
For ease of notation, set

$$N = \mathsf{rank}(\hat{A}).$$

Consider the matrix $\hat{\Delta} \in \mathbb{Z}_p^{n_1 \times n_2}$ that has entries $\hat{\Delta}_{ii} = 1$ for $1 \leq i \leq N$ and zeros everywhere else.

$$\hat{\Delta} = \begin{pmatrix} \hat{I}_N & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{4}$$

Then $\mathsf{rank}(\hat{\Delta}) = N$ and we can pick invertible matrices $\hat{P}$, $\hat{Q}$ such that

$$\hat{\Delta} = \hat{P}\hat{A}\hat{Q}. \tag{5}$$

PPT algorithm to generate new proofs:
We define a PPT algorithm $\mathcal{A}$ that given an instance $\boldsymbol{\phi}$ and a verifying proof $(\boldsymbol{\varPi}_1, \boldsymbol{\varPi}_2)$ outputs a new valid proof.

```
$\underline{\mathcal{A}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{\varPi}_1, \boldsymbol{\varPi}_2)}$
$\hat{A}, \hat{B}, \hat{C} \leftarrow$ matrices in the verification equation defined by $\mathbf{crs}$ and $\boldsymbol{\phi}$
$G^{\boldsymbol{u}}, H^{\boldsymbol{v}}, e(G,H)^z \leftarrow$ vectors from $\mathbf{crs}$
$N = rank(\hat{A})$;  $\hat{\Delta} =$ matrix defined by (4)
$\hat{P}, \hat{Q} \leftarrow$ matrices that satisfy (5)
$\boldsymbol{\varPi}_1' \leftarrow \boldsymbol{\varPi}_1 \boldsymbol{\varPi}_1^{-2\hat{P}^{-1}\hat{\Delta}\hat{P}} (G^{\boldsymbol{u}})^{-2\hat{C}\hat{Q}\hat{\Delta}\hat{P}}$
$\boldsymbol{\varPi}_2' \leftarrow \boldsymbol{\varPi}_2 \boldsymbol{\varPi}_2^{-2(\hat{Q}^{-1})^T\hat{\Delta}\hat{Q}^T} (H^{\boldsymbol{v}})^{-2\hat{B}^T\hat{P}^T\hat{\Delta}\hat{Q}^T}$
return $(\boldsymbol{\varPi}_1', \boldsymbol{\varPi}_2')$
```

Let $\boldsymbol{\pi}_1 = \log(\boldsymbol{\varPi}_1)$ and $\boldsymbol{\pi}_2 = \log(\boldsymbol{\varPi}_2)$. Then plugging $\boldsymbol{\varPi}_1'$ and $\boldsymbol{\varPi}_2'$ into the left hand side of the rewritten verification equation (3) leads to the following value in the discrete logs.

$$\begin{aligned}
&\left(\boldsymbol{\pi}_1 - 2(\boldsymbol{\pi}_1\hat{P}^{-1} + \mathbf{u}\hat{C}\hat{Q})\hat{\Delta}\hat{P}\right)\hat{A}\left(\boldsymbol{\pi}_2 - 2\left(\boldsymbol{\pi}_2(\hat{Q}^{-1})^T + \mathbf{v}\hat{B}^T\hat{P}^T\right)\hat{\Delta}\hat{Q}^T\right)^T \\
&+ \left(\boldsymbol{\pi}_1 - 2(\boldsymbol{\pi}_1\hat{P}^{-1} + \mathbf{u}\hat{C}\hat{Q})\hat{\Delta}\hat{P}\right)\hat{B}\mathbf{v}^T + \mathbf{u}\hat{C}\left(\boldsymbol{\pi}_2 - 2(\boldsymbol{\pi}_2\hat{Q}^{-T} + \mathbf{v}\hat{B}^T\hat{P}^T)\hat{\Delta}\hat{Q}^T\right)^T \\
= \quad & \boldsymbol{\pi}_1\hat{A}\boldsymbol{\pi}_2^T + \boldsymbol{\pi}_1\hat{A}\left(-2\hat{Q}\hat{\Delta}\left(\hat{Q}^{-1}\boldsymbol{\pi}_2^T + \hat{P}\hat{B}\mathbf{v}^T\right)\right) + \left(-2(\boldsymbol{\pi}_1\hat{P}^{-1} + \mathbf{u}\hat{C}\hat{Q})\hat{\Delta}\hat{P}\right)\hat{A}\boldsymbol{\pi}_2^T \\
&\left(-2(\boldsymbol{\pi}_1\hat{P}^{-1} + \mathbf{u}\hat{C}\hat{Q})\hat{\Delta}\hat{P}\right)\hat{A}\left(-2\hat{Q}\hat{\Delta}\left(\hat{Q}^{-1}\boldsymbol{\pi}_2^T + \hat{P}\hat{B}\mathbf{v}^T\right)\right) \\
&+ \boldsymbol{\pi}_1\hat{B}\mathbf{v}^T + \left(-2(\boldsymbol{\pi}_1\hat{P}^{-1} + \mathbf{u}\hat{C}\hat{Q})\hat{\Delta}\hat{P}\right)\hat{B}\mathbf{v}^T + \mathbf{u}\hat{C}\boldsymbol{\pi}_2^T + \mathbf{u}\hat{C}\left(-2\hat{Q}\hat{\Delta}(\hat{Q}^{-1}\boldsymbol{\pi}_2^T + \hat{P}\hat{B}\mathbf{v}^T)\right).
\end{aligned}$$

Rearranging, we see that this is equal to

$$\begin{aligned}
&\boldsymbol{\pi}_1\hat{A}\boldsymbol{\pi}_2^T + \boldsymbol{\pi}_1\hat{B}\mathbf{v}^T + \mathbf{u}\hat{C}\boldsymbol{\pi}_2^T \\
&+ \boldsymbol{\pi}_1\left(-2\hat{A}\hat{Q}\hat{\Delta}\hat{Q}^{-1} - 2\hat{P}^{-1}\hat{\Delta}\hat{P}\hat{A} + 4\hat{P}^{-1}\hat{\Delta}\hat{P}\hat{A}\hat{Q}\hat{\Delta}\hat{Q}^{-1}\right)\boldsymbol{\pi}_2^T \\
&+ \boldsymbol{\pi}_1\left(-2\hat{A}\hat{Q}\hat{\Delta}\hat{P}\hat{B} + 4\hat{P}^{-1}\hat{\Delta}\hat{P}\hat{A}\hat{Q}\hat{\Delta}\hat{P}\hat{B} - 2\hat{P}^{-1}\hat{\Delta}\hat{P}\hat{B}\right)\mathbf{v}^T \\
&+ \mathbf{u}\left(-2\hat{C}\hat{Q}\hat{\Delta}\hat{P}\hat{A} + 4\hat{C}\hat{Q}\hat{\Delta}\hat{P}\hat{A}\hat{Q}\hat{\Delta}\hat{Q}^{-1} - 2\hat{C}\hat{Q}\hat{\Delta}\hat{Q}^{-1}\right)\boldsymbol{\pi}_2^T \\
&+ \mathbf{u}\left(4\hat{C}\hat{Q}\hat{\Delta}\hat{P}\hat{A}\hat{Q}\hat{\Delta}\hat{P}\hat{B} - 2\hat{C}\hat{Q}\hat{\Delta}\hat{P}\hat{B} - 2\hat{C}\hat{Q}\hat{\Delta}\hat{P}\hat{B}\right)\mathbf{v}^T.
\end{aligned}$$

Using that $\hat{\Delta} = \hat{\Delta}\hat{\Delta}$, $\hat{\Delta} = \hat{P}\hat{A}\hat{Q}$, $\hat{P}^{-1}\hat{\Delta} = \hat{P}^{-1}\hat{\Delta}\hat{\Delta} = \hat{A}\hat{Q}\hat{\Delta}$ and that $\hat{\Delta}\hat{Q}^{-1} = \hat{\Delta}\hat{P}\hat{A}$ we see that this is equal to

$$\begin{aligned}
&\boldsymbol{\pi}_1\hat{A}\boldsymbol{\pi}_2^T + \boldsymbol{\pi}_1\hat{B}\mathbf{v}^T + \mathbf{u}\hat{C}\boldsymbol{\pi}_2^T \\
&+ \boldsymbol{\pi}_1\left(-2\hat{A}\hat{Q}\hat{\Delta}\hat{P}\hat{A} - 2\hat{A}\hat{Q}\hat{\Delta}\hat{P}\hat{A} + 4\hat{A}\hat{Q}\hat{\Delta}\hat{P}\hat{A}\right)\boldsymbol{\pi}_2^T \\
&+ \boldsymbol{\pi}_1\left(-2\hat{A}\hat{Q}\hat{\Delta}\hat{P}\hat{B} + 4\hat{A}\hat{Q}\hat{\Delta}\hat{P}\hat{B} - 2\hat{A}\hat{Q}\hat{\Delta}\hat{P}\hat{B}\right)\mathbf{v}^T \\
&+ \mathbf{u}\left(-2\hat{C}\hat{Q}\hat{\Delta}\hat{P}\hat{A} + 4\hat{C}\hat{Q}\hat{\Delta}\hat{P}\hat{A} - 2\hat{C}\hat{Q}\hat{\Delta}\hat{P}\hat{A}\right)\boldsymbol{\pi}_2^T \\
&+ \mathbf{u}\left(4\hat{C}\hat{Q}\hat{\Delta}\hat{P}\hat{B} - 2\hat{C}\hat{Q}\hat{\Delta}\hat{P}\hat{B} - 2\hat{C}\hat{Q}\hat{\Delta}\hat{P}\hat{B}\right)\mathbf{v}^T.
\end{aligned}$$

Cancelling terms leaves

$$\boldsymbol{\pi}_1\hat{A}\boldsymbol{\pi}_2^T + \boldsymbol{\pi}_1\hat{B}\mathbf{v}^T + \mathbf{u}\hat{C}\boldsymbol{\pi}_2^T,$$

which we know matches the right hand side of (3).

Analysis yielding contradiction:
By simulation-soundness we have on $\boldsymbol{\phi} \leftarrow \mathsf{No}(R)$ that there is negligible chance of $\boldsymbol{\Pi}_1' \neq \boldsymbol{\Pi}_1$ or $\boldsymbol{\Pi}_2' \neq \boldsymbol{\Pi}_2$.

$$\Pr\left[\begin{array}{c} R \leftarrow \mathcal{R}(1^\lambda);\ \boldsymbol{\phi} \leftarrow \mathsf{No}(R);\ (\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R); \\ (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \leftarrow \mathsf{ZSimProve}(\mathbf{crs}, \boldsymbol{\tau}, \boldsymbol{\phi})\ : \\ \\ \mathcal{A}(\mathsf{crs}, \boldsymbol{\phi}, \boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \neq (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \end{array}\right] \approx 0.$$

Since $\mathsf{Yes}, \mathsf{No}$ is a hard decisional problem, we get the same for $\boldsymbol{\phi} \leftarrow \mathsf{Yes}(R)$.

$$\Pr\left[\begin{array}{c} R \leftarrow \mathcal{R}(1^\lambda);\ (\boldsymbol{\phi}, \boldsymbol{w}) \leftarrow \mathsf{Yes}(R);\ (\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R); \\ (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \leftarrow \mathsf{ZSimProve}(\mathbf{crs}, \boldsymbol{\tau}, \boldsymbol{\phi})\ : \\ \\ \mathcal{A}(\mathsf{crs}, \boldsymbol{\phi}, \boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \neq (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \end{array}\right] \approx 0.$$

By zero-knowledge, this also holds for real proofs.

$$\Pr\left[\begin{array}{c} R \leftarrow \mathcal{R}(1^\lambda);\ (\boldsymbol{\phi}, \boldsymbol{w}) \leftarrow \mathsf{Yes}(R);\ (\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R); \\ (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \leftarrow \mathsf{ZProve}(\mathbf{crs}, \boldsymbol{\tau}, \boldsymbol{w})\ : \\ \\ \mathcal{A}(\mathsf{crs}, \boldsymbol{\phi}, \boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \neq (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \end{array}\right] \approx 0.$$

So both on Yes-instances and No-instances, we have overwhelming probability $\boldsymbol{\Pi}_1' = \boldsymbol{\Pi}_1$. Taking discrete logarithms this means

$$\boldsymbol{\pi}_1 = \boldsymbol{\pi}_1 - 2(\boldsymbol{\pi}_1\hat{P}^{-1} + \mathbf{u}\hat{C}\hat{Q})\hat{\Delta}\hat{P}.$$

Since $\hat{P}$ is invertible this means

$$\boldsymbol{\pi}_1\hat{P}^{-1}\hat{\Delta} = -\mathbf{u}\hat{C}\hat{Q}\hat{\Delta}.$$

This means that

$$\boldsymbol{\pi}_1\hat{A}\boldsymbol{\pi}_2^T = \boldsymbol{\pi}_1\hat{P}^{-1}\hat{\Delta}\hat{Q}^{-1}\boldsymbol{\pi}_2^T = -\mathbf{u}\hat{C}\hat{Q}\hat{\Delta}\hat{Q}^{-1}\boldsymbol{\pi}_2^T.$$

We could therefore replace the first component $e(\boldsymbol{\Pi}_1^{\hat{A}}, \boldsymbol{\Pi}_2)$ of the verification equation by $e((G^{\mathbf{u}})^{\hat{C}\hat{Q}\hat{\Delta}\hat{Q}^{-1}}, \boldsymbol{\Pi}_2)$ and with overwhelming probability get the same result of the verification.

On a Yes-instance, we know that the prover can find matrices that can be applied to the group elements in the common reference string to yield a proof $(\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2)$. The proof matrices are chosen independently of the common reference string, so if we sample many common reference strings, by completeness they have overwhelming probability of yielding verifying proofs on all of the common reference strings. After replacing the quadratic component in the verification equations, they correspond to a large system of linear equations, and we know the discrete logarithms of all the common reference strings. This means we can solve a system of linear equations to find one or more pairs of suitable proof matrices that yield verifying proofs with high probability.

On a No-instance on the other hand, soundness of the argument means that there is negligible chance of computing a valid proof and hence no such proof matrices exist. We can therefore distinguish Yes-instances and No-instances. □

6.5 Number of Group Elements

We will now show that that there is no pairing-based SS-NIZK argument where proofs only have two group elements for hard to decide relations. Our strategy is to show that given an SS-NIZK argument for a relation generator $\mathcal{R}$ with PPT sampling algorithms Yes and No it is trivial to distinguish whether instances have been output by Yes or No. Hence the decisional problems behind the SS-NIZK are not hard.

Theorem 6.3. *If* (ZSetup, ZProve, ZVfy, ZSimProve) *is a pairing-based SS-NIZK argument for relation generator $\mathcal{R}$ with a hard decisional problem* Yes, No *then the proofs must contain at least 3 group elements.*

Proof. Since the verification equations must involve a pairing between source group elements in the proof, we can without loss of generality assume $\hat{A}_1 \neq 0$. This means we have $n_1 \geq 1$ and $n_2 \geq 1$, so the case we need to rule out is $n_1 = n_2 = 1$ and $n_T = 0$.

Let $\hat{A}_i, \ldots, \hat{F}_i$ be the matrices output by ZVfy′. Note that $\hat{A}_i$ are single field elements as there is only one proof element on each side of the pairing. By scaling the first verification equation by the exponent $\frac{1}{\hat{A}_1}$ we can without loss of generality assume $\hat{A}_1 = 1$. Moreover, by dividing subsequent verification equations with appropriately scaled versions of the first verification equation, we can without loss of generality also assume $\hat{A}_2 = \ldots = \hat{A}_\eta = 0$. Two linearly independent equalities in two variables is uniquely solvable, so if the first verification with $\hat{A}_1 = 1$ is non-redundant then the remaining verification equations can impose at most one linear constraint on the proof. This means that they can be combined to a single linear constraint. Theorem 6.2 showed there must be two non-trivial verification equations, which means that we can without loss of generality consider only the case where $\eta = 2$, $\hat{A}_1 = (1)$ and $\hat{A}_2 = (0)$.

We will now show that the existence of a pairing-based SS-NIZK argument with $\eta = 2$, $\hat{A}_1 = (1)$ and $\hat{A}_2 = (0)$ implies a decision procedure for determining whether an instance $\boldsymbol{\phi}$ has been sampled by Yes or No. We break the analysis into two cases depending on whether the simulation is likely to always produce the same proofs or whether it returns randomized proofs (noting that by the structure of verification equations there are at most two acceptable proofs).

Case I: The simulator always outputs the same proof in repeated executions.
In Case I we assume

$$\Pr\left[\begin{array}{c} R \leftarrow \mathcal{R}(1^\lambda);\ (\mathbf{crs}, \boldsymbol{\tau})) \leftarrow \mathsf{ZSetup}(R);\ (\boldsymbol{\phi}, \boldsymbol{w}) \leftarrow \mathsf{Yes}(R) : \\ \mathsf{ZProve}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{w}) = \mathsf{ZProve}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{w}) \end{array}\right] \approx 1.$$

We construct a PPT decision algorithm $\mathcal{A}$ for the hard decisional problem.

```
$\underline{\mathcal{A}(R, \boldsymbol{\phi})}$
$m = \max\{m_1, m_2\}$
for $i = 0, .., m$ :
    $(\mathbf{u}_i, \mathbf{v}_i, \mathbf{z}_i, \boldsymbol{\tau}_i) = \mathsf{ZSetup}'(R)$
    $(G^{s_i}, H^{t_i}) \leftarrow \mathsf{ZSimProve}(\mathbf{crs}_i, \boldsymbol{\tau}_i, \boldsymbol{\phi})$
    $(\hat{A}_1, \ldots, \hat{F}_2) \leftarrow \mathsf{ZVfy}'(R, \boldsymbol{\phi})$
    $s_i, t_i \leftarrow$ compute from verification equations
return 1 if and only if it is possible to find $\hat{P}, \hat{Q}$ such that
for all $i = 0, \ldots, m$ we have $s_i = \mathbf{u}_i\hat{P}, t_i = \mathbf{v}_i\hat{Q}$
```

Let us analyse the performance of $\mathcal{A}$ when working on $\boldsymbol{\phi}$ generated by Yes and No, respectively. First we consider the Yes case and define another PPT algorithm $\mathcal{B}$ that is similar to $\mathcal{A}$ but instead uses real proofs.

```
$\underline{\mathcal{B}(R, \boldsymbol{\phi}, \boldsymbol{w})}$
$(\hat{P}', \hat{Q}') \leftarrow \mathsf{ZProve}'(R, \boldsymbol{\phi}, \boldsymbol{w})$
$m = \max\{m_1, m_2\}$
for $i = 0, .., m$ :
    $(\mathbf{u}_i, \mathbf{v}_i, \mathbf{z}_i, \boldsymbol{\tau}_i) = \mathsf{ZSetup}'(R)$
    $s_i = \mathbf{u}_i\hat{P}', t_i = \mathbf{v}_i\hat{Q}'$
return 1 if and only if it is possible to find $\hat{P}, \hat{Q}$ such that
for all $i = 0, \ldots, m$ we have $s_i = \mathbf{u}_i\hat{P}, t_i = \mathbf{v}_i\hat{Q}$
```

Since we are in Case I, when running $\mathcal{B}$ we would in each iteration $i = 0, \ldots, m$ get the same (s_i, t_i) pairs even if running the prover algorithm several times. By the zero-knowledge property real proofs should match these unique (s_i, t_i) pairs, so $\mathcal{A}$ generates similar s_i, t_i values in its execution on a Yes-instance.

$$\Pr[R \leftarrow \mathcal{R}(1^\lambda);\ (\boldsymbol{\phi}, \boldsymbol{w}) \leftarrow \mathsf{Yes}(R) : \mathcal{A}(R, \boldsymbol{\phi}) = 1] \\ \approx \Pr[R \leftarrow \mathcal{R}(1^\lambda);\ (\boldsymbol{\phi}, \boldsymbol{w}) \leftarrow \mathsf{Yes}(R) : \mathcal{B}(R, \boldsymbol{\phi}, \boldsymbol{w}) = 1].$$

When running $\mathcal{B}$ there exists at least one solution $\hat{P} = \hat{P}'$ and $\hat{Q} = \hat{Q}'$ for the matrices, so Gaussian elimination will provide some $\hat{P}$ and $\hat{Q}$ solution. This means

$$\Pr[R \leftarrow \mathcal{R}(1^\lambda);\ (\boldsymbol{\phi}, \boldsymbol{w}) \leftarrow \mathsf{Yes}(R) : \mathcal{B}(R, \boldsymbol{\phi}, \boldsymbol{w}) = 1] \approx 1.$$

Next, we consider the No case. Also here it must be true that the simulator produces unique (s_i, t_i) pairs, since otherwise we could use the deviation in simulated proofs to determine we had sampled $\boldsymbol{\phi} \leftarrow \mathsf{No}(R)$. Now, if $\mathcal{A}$ returns 1 it means it found matrices $\hat{P}, \hat{Q}$ that explain all the simulated verification equation solutions (s_i, t_i). The dimensions of $\hat{P}, \hat{Q}$ are $m_1 \times 1$ and $m_2 \times 1$, so we have more equations in (s_i, t_i) than variables. Some of the equations must therefore be linearly dependent. This means, if we were to sample yet another pair of common reference string and verification equations, then there is a chance that it too would be linearly dependent on previous verification equations and hence the matrices $\hat{P}, \hat{Q}$ would yield accepting proofs. However, by soundness such matrices $\hat{P}, \hat{Q}$ cannot exist, since it would mean the forgery of a proof for a false instance. We therefore get

$$\Pr[R \leftarrow \mathcal{R}(1^\lambda);\ \boldsymbol{\phi} \leftarrow \mathsf{No}(R) : \mathcal{A}(R, \boldsymbol{\phi}) = 1] \approx 0,$$

which contradicts that $\mathsf{Yes}, \mathsf{No}$ was a hard to decision problem for $\mathcal{R}$.

Remark 6.1. It is tempting to use the same analysis also when the proofs sometimes differ. Just let $\mathcal{A}$ use both posisble proofs. The issue is that the adversary does not know which proof element, s_i or s_i' to use in the system of equations, so it has to run through up to 2^{m+1} combinations in order to try all combinations to find the proof vector $\hat{P}$ and the same for finding $\hat{Q}$. If the common reference string is large then this adversary is not polynomial time. Nonetheless, it does show that any computationally sound NIZK argument must either have a long common reference string or at least 3 group elements in the proofs. In the next part, we leverage simulation soundness to show that on any common reference string whether short or long we cannot have two group element proofs.

Case II: The simulator sometimes produces different proofs in repeated executions.

Let

$$\epsilon(\lambda) = \Pr\left[\begin{array}{c} R \leftarrow \mathcal{R}(1^\lambda);\ (\mathbf{crs}, \boldsymbol{\tau})) \leftarrow \mathsf{ZSetup}(R);\ (\boldsymbol{\phi}, \boldsymbol{w}) \leftarrow \mathsf{Yes}(R) : \\ \mathsf{ZProve}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{w}) \neq \mathsf{ZProve}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{w}) \end{array}\right].$$

In Case II we have that $\epsilon(\lambda) \not\approx 0$, so there is a constant $c > 0$ such that $\epsilon(\lambda) \geq \frac{1}{\lambda^c}$ for infinitely many choices of $\lambda \in \mathbb{N}$.

We construct a PPT algorithm $\mathcal{A}$ that whenever there is two pairs of proof matrices takes as input a CRS and an instance $\boldsymbol{\phi} \in L_R$ and matrices that can be used to compute the difference between the two proofs.

```
$\underline{\mathcal{A}(\mathbf{crs}, \boldsymbol{\phi})}$
$m = \max\{m_1, m_2\}$
for $i = 1, .., m$ :
    $(\mathbf{u}_i, \mathbf{v}_i, \mathbf{z}_i, \boldsymbol{\tau}_i) = \mathsf{ZSetup}'(R)$
    $(s_i, t_i), (s_i', t_i') \leftarrow$ compute two solutions to verification equation
    $(x_i, y_i) = (s_i + s_i', t_i + t_i')$
solve for two matrices $\hat{S}, \hat{T}$ such that for all $i = 1, \ldots, m$
    $s_i + s_i' = \mathbf{u}_i \hat{S}$ and $t_i + t_i' = \mathbf{v}_i \hat{T}$
return $(\hat{S}, \hat{T})$, and else abort
```

Now, observe that if the prover is run twice and produces two different proofs for the same statement, then it must have used two different proof matrices. By disclosure-freeness, there is then overwhelming probability that these two proof matrices would also produce distinct proofs on another freshly drawn CRS. Moreover, since the verification equations have exactly two different solutions, these two proofs are uniquely defined. The algorithm $\mathcal{A}$ would in this case succeed in finding $\hat{S}, \hat{T}$. Written formally, we get

$$\Pr\left[\begin{array}{c} R \leftarrow \mathcal{R}(1^\lambda);\ (\boldsymbol{\phi}, \boldsymbol{w}) \leftarrow \mathsf{Yes}(R);\ (\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R); \\ (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \leftarrow \mathsf{ZProve}(\mathbf{crs}, \boldsymbol{\phi}, \boldsymbol{w});\ (\hat{S}, \hat{T}) \leftarrow \mathcal{A}(R, \boldsymbol{\phi}); \\ \boldsymbol{\Pi}_1' \leftarrow (G^{\mathbf{u}})^{\hat{S}} \boldsymbol{\Pi}^{-1};\ \boldsymbol{\Pi}_1' \leftarrow (G^{\mathbf{u}})^{\hat{S}} \boldsymbol{\Pi}^{-1} \ : \\ \\ (\boldsymbol{\Pi}_1', \boldsymbol{\Pi}_2') \neq (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \text{ and } \mathsf{ZVfy}(\mathbf{crs}, \boldsymbol{\phi}, (\boldsymbol{\Pi}_1', \boldsymbol{\Pi}_2')) = 1 \end{array}\right] \approx \epsilon(\lambda).$$

From the zero-knowledge property we then get

$$\Pr\left[\begin{array}{c} R \leftarrow \mathcal{R}(1^\lambda);\ (\boldsymbol{\phi}, \boldsymbol{w}) \leftarrow \mathsf{Yes}(R);\ (\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R); \\ (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \leftarrow \mathsf{ZSimProve}(\mathbf{crs}, \boldsymbol{\tau}, \boldsymbol{\phi});\ (\hat{S}, \hat{T}) \leftarrow \mathcal{A}(R, \boldsymbol{\phi}); \\ \boldsymbol{\Pi}'_1 \leftarrow (G^{\mathbf{u}})^{\hat{S}} \boldsymbol{\Pi}^{-1};\ \boldsymbol{\Pi}'_1 \leftarrow (G^{\mathbf{u}})^{\hat{S}} \boldsymbol{\Pi}^{-1} : \\ \\ (\boldsymbol{\Pi}'_1, \boldsymbol{\Pi}'_2) \neq (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \text{ and } \mathsf{ZVfy}(\mathbf{crs}, \boldsymbol{\phi}, (\boldsymbol{\Pi}'_1, \boldsymbol{\Pi}'_2)) = 1 \end{array}\right] \approx \epsilon(\lambda).$$

The hardness of deciding whether $\boldsymbol{\phi}$ is drawn as a Yes-instance or a No-instance, now implies

$$\Pr\left[\begin{array}{c} R \leftarrow \mathcal{R}(1^\lambda);\ \boldsymbol{\phi} \leftarrow \mathsf{No}(R);\ (\mathbf{crs}, \boldsymbol{\tau}) \leftarrow \mathsf{ZSetup}(R); \\ (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \leftarrow \mathsf{ZSimProve}(\mathbf{crs}, \boldsymbol{\tau}, \boldsymbol{\phi});\ (\hat{S}, \hat{T}) \leftarrow \mathcal{A}(R, \boldsymbol{\phi}); \\ \boldsymbol{\Pi}'_1 \leftarrow (G^{\mathbf{u}})^{\hat{S}} \boldsymbol{\Pi}^{-1};\ \boldsymbol{\Pi}'_1 \leftarrow (G^{\mathbf{u}})^{\hat{S}} \boldsymbol{\Pi}^{-1} : \\ \\ (\boldsymbol{\Pi}'_1, \boldsymbol{\Pi}'_2) \neq (\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \text{ and } \mathsf{ZVfy}(\mathbf{crs}, \boldsymbol{\phi}, (\boldsymbol{\Pi}'_1, \boldsymbol{\Pi}'_2)) = 1 \end{array}\right] \approx \epsilon(\lambda).$$

However, this contradicts simulation soundness. □

A Square Arithmetic Programs

We defined Square Arithmetic Program (SAP) relations in Section 5.1. We will now show how any arithmetic circuit with fan-in 2 gates over a finite field $\mathbb{Z}_p$ can be expressed as a SAP over the same finite field. The conversion is largely the same as the conversions described in [GGPR13] and [DFGK14], however we also discuss how to instantiate the trick of replacing multiplications with squarings that was mentioned in [Gro16].

Gennaro, Gentry, Parno and Raykova [GGPR13] introduced quadratic span programs (QSPs) and quadratic arithmetic programs (QAPs). These define NP-complete languages specified by a quadratic equation over polynomials. QSPs characterise boolean circuits and QAPs characterise arithmetic circuits in a natural way. Danezis, Fournet, Groth and Kohlweiss [DFGK14] noticed that by replacing each of the constraints in QSPs with 2 other constaints, it is possible to design a square span program (SSP), which is a QSP in which the two sets of polynomials involved in the quadratic term are identical. Using this technique they were able to reduce the number of proof elements and verification equations (at the cost of a circuit with twice as many gates) by having the quadratic proof components on each side of the pairing be replicas. We use SAPs in a similar way to make the group elements A and B in our proof symmetric.

An arithmetic circuit can be described as a set of arithmetic constraints over the wires $s_1, \ldots, s_m$. We fix the constant $s_0 = 1$, use $s_1, \ldots, s_\ell \in \mathbb{Z}_p$ to describe the instance, and the rest of the wires $s_{\ell+1}, \ldots, s_m$ can be viewed as the witness. Generalising from the addition and multiplication constraints that arise in an arithmetic circuit, e.g., $s_i \cdot s_j = s_k$ we consider a set of n multiplication constraints of the form

$$\sum_{i=0}^{m} s_i u_{i,q} \cdot \sum_{i=0}^{m} s_i v_{i,q} = \sum_{i=0}^{m} s_i w_{i,q},$$

where $u_{i,q}, v_{i,q}, w_{i,q}$ are constants in $\mathbb{Z}_p$ specifying the qth equation.

Our SAP is based on a simplification of systems of arithmetic constraints, where all multiplications are replaced with squarings. As suggested by [Gro16], we can write a product

$ab = \frac{(a+b)^2-(a-b)^2}{4}$. A system with n multiplication constraints can therefore be rewritten as a system with at most $2n$ squaring constraints. To be precise, for each multiplication constraint $\sum_{i=0}^m s_i u_{i,q} \cdot \sum_{i=0}^m s_i v_{i,q} = \sum_{i=0}^m s_i w_{i,q}$ we introduce a new variable s_{m+q} and replace the multiplication constraint with two squaring constraints

1. $\left(\sum_{i=0}^m s_i(u_{i,q}+v_{i,q})\right)^2 = 4\sum_{i=0}^m s_i w_{i,q} + s_{m+q}$;
2. $\left(\sum_{i=0}^m s_i(u_{i,q}-v_{i,q})\right)^2 = s_{m+q}$.

Consider now n squaring constraints $\left\{\left(\sum_{i=0}^m s_i u_{i,q}\right)^2 = \sum_{i=0}^m s_i w_{i,q}\right\}_{q=1}^n$. Let us pick arbitrary distinct $r_1, \ldots, r_n \in \mathbb{Z}_p$ and define $t(x) = \prod_{q=1}^n (x - r_q)$. Furthermore, let $u_i(x), w_i(x)$ be degree $n-1$ polynomials such that

$$u_i(r_q) = u_{i,q} \quad \text{and} \quad w_i(r_q) = w_{i,q} \qquad \text{for} \quad i = 0, \ldots, m, q = 1, \ldots, n.$$

We now have that $s_0 = 1$ and the variables $s_1, \ldots, s_m \in \mathbb{Z}_p$ satisfy the n constraints if and only if in each point $r_1, \ldots, r_q$

$$\left(\sum_{i=0}^m s_i u_i(r_q)\right)^2 = \sum_{i=0}^m s_i w_i(r_q).$$

Since $t(X)$ is the lowest degree monomial with $t(r_q) = 0$ in each point, we can reformulate this condition as

$$\left(\sum_{i=0}^m s_i u_i(X)\right)^2 \equiv \sum_{i=0}^m s_i w_i(X) \mod t(X).$$

This gives rise to a SAP as defined in Section 5.1.

Formally, we work with square arithmetic programs R that have the following description

$$R = \left(\mathbb{Z}_p, \text{aux}, \ell, \{u_i(X), w_i(X)\}_{i=0}^m, t(X)\right),$$

where p is a prime, aux is some auxiliary information, $1 \le \ell \le m$, $u_i(X), w_i(X), t(X) \in \mathbb{Z}_p[X]$ and $u_i(X), w_i(X)$ have strictly lower degree than n, the degree of $t(X)$. A square arithmetic program with such a description defines the following binary relation, where we let $s_0 = 1$,

$$R = \left\{ (\phi, w) \left| \begin{array}{l} \phi = (s_1, \ldots, s_\ell) \in \mathbb{Z}_p^\ell \\ w = (s_{\ell+1}, \ldots, s_m) \in \mathbb{Z}_p^{m-\ell} \\ \\ \quad \left(\sum_{i=0}^m s_i u_i(X)\right)^2 \equiv \sum_{i=0}^m s_i w_i(X) \mod t(X) \end{array} \right. \right\}.$$

We say $\mathcal{R}$ is a square arithmetic program generator if it generates relations of the form given above with $p > 2^{\lambda-1}$.

Relations can arise in many different ways in practice. It may be that the relationship generator is deterministic or it may be that it is randomized. It may be that first the prime p is generated and then the rest of the relation is built on top of the $\mathbb{Z}_p$. Or it may be that the polynomials are specified first and then a random field $\mathbb{Z}_p$ is chosen. To get maximal flexibility we have chosen our definitions to be agnostic with respect to the exact way the field and the relation is generated, the different options can all be modelled by appropriate choices of relation generators.

In our pairing-based NIZK arguments the auxiliary information aux specifies a bilinear group over $\mathbb{Z}_p$. It may seem a bit surprising to make the choice of bilinear group part of the relation generator but this provides a better model of settings where the relation is built on top of an already existing bilinear group. Again, there is no loss of generality in this choice, one can think of a traditional setting where the relation is chosen first and then the bilinear group is chosen at random as the special case where the relation generator works in two steps, first choosing the relation and then picking a random bilinear group. Of course letting the relation generator pick the bilinear group is another good reason that we need to assume it is benign; an appropriate choice of bilinear group is essential for security.

We use in the security proofs that the polynomials $u_0(X), \ldots, u_\ell(X)$ are linearly independent from each other and are also linearly independent from $u_{\ell+1}(X), \ldots, u_m(X)$. When constructing the square arithmetic program from squaring constraints, we can achieve this by adding a new variable s_i' and a squaring constraints $s_i^2 = s_i'$ for any $u_i(X)$ that is not already linearly independent. This makes $u_i(X)$ linearly independent from all the other $u_j(X)$ since $u_i(r_{n+i}) = 1$, while $u_j(r_{n+i}) = 0$ for all $j \neq i$.

Finally, we note that given a square arithmetic program defining a relation R', we can formulate a square arithmetic program defining

$$R = \{((K, h, \boldsymbol{\phi}), \boldsymbol{w}) : K \in \{0,1\}^{\ell_K(\lambda)} \ \wedge \ h \in \{0,1\}^{\ell_h(\lambda)} \ \wedge \ (\boldsymbol{\phi}, \boldsymbol{w}) \in R'\},$$

by thinking of $\{0,1\}^{\ell_K(\lambda)} \times \{0,1\}^{\ell_h(\lambda)}$ as embedded in $\mathbb{Z}_p^d$ in a natural way, adding new variables $s_1', s_1'', \ldots, s_d', s_d''$ to the instance, and adding squaring constraints $(s_i')^2 = s_i''$. This means that from a simulation-extractable NIZK for relation R, we can build a simulation extractable signature of knowledge as in Section 3.

Acknowledgments

We thank Vasilios Mavroudis and Markulf Kohlweiss for helpful discussions.

References

[AF07] Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 118–136. Springer, 2007.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 440–456. Springer, 2005.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 111–120. ACM, 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, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474. IEEE, 2014.

[BCI$^+$13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 315–333. Springer, 2013.

[BCK$^+$14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 551–572. Springer, 2014.

[BCPR13] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. Indistinguishability obfuscation vs. auxiliary-input extractable functions: One must fall. *IACR Cryptology ePrint Archive*, 2013:641, 2013.

[BCPR16] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. *SIAM Journal on Computing*, 45(5):1910–1952, 2016.

[BDSMP91] M Blum, A De Santis, S Micali, and G Persiano. Non-interactive zero-knowledge proof systems. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[BF14] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 520–537. Springer, 2014.

[BFG13a] David Bernhard, Georg Fuchsbauer, and Essam Ghadafi. Efficient signatures of knowledge and DAA in the standard model. In *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, pages 518–533. Springer, 2013.

[BFG$^+$13b] David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Anonymous attestation with user-controlled linkability. *Int. J. Inf. Sec.*, 12(3):219–249, 2013.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 103–112. ACM, 1988.

[BP14] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 236–261. Springer, 2014.

[BSCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In *International Cryptology Conference*, pages 276–294. Springer, 2014.

[CF08] Xiaofeng Chen and Dengguo Feng. A new direct anonymous attestation scheme from bilinear maps. In *Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS 2008, Zhang Jia Jie, Hunan, China, November 18-21, 2008*, pages 2308–2313. Springer, 2008.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pages 78–96, 2006.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 410–424. Springer, 1997.

[Dam92] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 341–355. Springer, 1992.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 532–550. Springer, 2014.

[DS16] David Derler and Daniel Slamanig. Fully-anonymous short dynamic group signatures without encryption. *IACR Cryptology ePrint Archive*, 2016:154, 2016.

[DSDCP00] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Necessary and sufficient assumptions for non-interactive zero-knowledge proofs of knowledge for all np relations. In *International Colloquium on Automata, Languages, and Programming*, pages 451–462. Springer, 2000.

[EHK$^+$17] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Rafols, and Jorge Villar. An algebraic framework for diffie–hellman assumptions. *Journal of Cryptology*, 30(1):242–288, 2017.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 60–79. Springer, 2012.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999.

[FO11] Marc Fischlin and Cristina Onete. Relaxed security notions for signatures of knowledge. In *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*, pages 309–326, 2011.

[FXC09] Deng-Guo Feng, Jing Xu, and Xiao-Feng Chen. An efficient direct anonymous attestation scheme with forward security. *WSEAS TRANSACTIONS on COMMUNICATIONS*, 8(10):1076–1085, 2009.

[GG17] Essam Ghadafi and Jens Groth. Towards a classification of non-interactive computational assumptions in cyclic groups. Cryptology ePrint Archive, Report 2017/343, 2017.

[GGI$^+$15] Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam D. Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *Journal of cryptology*, 28(4):820–843, 2015.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645. Springer, 2013.

[GO14] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. *Journal of Cryptology*, 27(3):506–543, 2014.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *Journal of the ACM (JACM)*, 59(3):11, 2012.

[GPS08] Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, pages 444–459. Springer, 2006.

[Gro10] Jens Groth. Short non-interactive zero-knowledge proofs. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 341–358. Springer, 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 305–326. Springer, 2016.

[GS12] Jens Groth and Amit Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM Journal on Computing*, 41(5):1193–1232, 2012.

[GT07] He Ge and Stephen R. Tate. A direct anonymous attestation scheme for embedded devices. In *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, pages 16–30. Springer, 2007.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 99–108. ACM, 2011.

[Kil95] Joe Kilian. Improved efficient arguments (preliminary version). In *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, pages 311–324. Springer, 1995.

[KP98] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for np with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 397–411. IEEE, 2013.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[MW98] Ueli M. Maurer and Stefan Wolf. Lower bounds on generic algorithms in groups. In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, pages 72–84. Springer, 1998.

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

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 554–571. Springer, 2008.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394, 1990.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 543–553. IEEE, 1999.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 256–266. Springer, 1997.

[SP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 427–436. IEEE, 1992.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, pages 1–18. Springer, 2008.

[YYQ$^+$15] Bo Yang, Kang Yang, Yu Qin, Zhenfeng Zhang, and Dengguo Feng. DAA-TZ: an efficient DAA scheme for mobile devices using ARM trustzone. In *Trust and Trustworthy Computing - 8th International Conference, TRUST 2015, Heraklion, Greece, August 24-26, 2015, Proceedings*, pages 209–227. Springer, 2015.