On the Size of Pairing-based Non-interactive Arguments*

Jens Groth**

University College London, UK
j.groth@ucl.ac.uk

Abstract. Non-interactive arguments enable a prover to convince a verifier that a statement is true. Recently there has been a lot of progress both in theory and practice on constructing highly efficient non-interactive arguments with small size and low verification complexity, so-called succinct non-interactive arguments (SNARGs) and succinct non-interactive arguments of knowledge (SNARKs).
Many constructions of SNARGs rely on pairing-based cryptography. In these constructions a proof consists of a number of group elements and the verification consists of checking a number of pairing product equations. The question we address in this article is how efficient pairing-based SNARGs can be.
Our first contribution is a pairing-based (preprocessing) SNARK for arithmetic circuit satisfiability, which is an NP-complete language. In our SNARK we work with asymmetric pairings for higher efficiency, a proof is only 3 group elements, and verification consists of checking a single pairing product equations using 3 pairings in total. Our SNARK is zero-knowledge and does not reveal anything about the witness the prover uses to make the proof.
As our second contribution we answer an open question of Bitansky, Chiesa, Ishai, Ostrovsky and Paneth (TCC 2013) by showing that linear interactive proofs cannot have a linear decision procedure. It follows from this that SNARGs where the prover and verifier use generic asymmetric bilinear group operations cannot consist of a single group element. This gives the first lower bound for pairing-based SNARGs. It remains an intriguing open problem whether this lower bound can be extended to rule out 2 group element SNARGs, which would prove optimality of our 3 element construction.

Keywords: SNARKs, non-interactive zero-knowledge arguments, linear interactive proofs, quadratic arithmetic programs, bilinear groups.

1 Introduction

Goldwasser, Micali and Rackoff [GMR89] introduced zero-knowledge proofs that enable a prover to convince a verifier that a statement is true without revealing anything else. They have three core properties:

Completeness: Given a statement and a witness, the prover can convince the verifier.
Soundness: A malicious prover cannot convince the verifier of a false statement.
Zero-knowledge: The proof does not reveal anything but the truth of the statement, in particular it does not reveal the prover's witness.

** 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 and the Engineering and Physical Sciences Research Council grant EP/J009520/1. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

Blum, Feldman and Micali [BFM88] extended the notion to *non-interactive* zero-knowledge (NIZK) proofs in the common reference string model. NIZK proofs are useful in the construction of non-interactive cryptographic schemes, e.g., digital signatures and CCA-secure public key encryption.

The amount of communication is an important performance parameter for zero-knowledge proofs. Kilian [Kil92] gave the first sublinear communication zero-knowledge argument that sends fewer bits than the size of the statement to be proved. Micali [Mic00] proposed sublinear size arguments by letting the prover in a communication efficient argument compute the verifier's challenges using a cryptographic function, and as remarked in Kilian [Kil95] this leads to sublinear size NIZK proofs when the interactive argument is public coin and zero-knowledge.

Groth, Ostrovsky and Sahai [GOS12,GOS06,Gro06,GS12] introduced pairing-based NIZK proofs, yielding the first linear size proofs based on standard assumptions. Groth [Gro10] combined these techniques with ideas from interactive zero-knowledge arguments [Gro09] to give the first constant size NIZK arguments. Lipmaa [Lip12] used an alternative construction based on progression-free sets to reduce the size of the common reference string.

Groth's constant size NIZK argument is based on constructing a set of polynomial equations and using pairings to efficiently verify these equations. Gennaro, Gentry, Parno and Raykova [GGPR13] found an insightful construction of polynomial equations based on Lagrange interpolation polynomials yielding a pairing-based NIZK argument with a common reference string size proportional to the size of the statement and witness. They gave two types of polynomial equations: quadratic span programs for proving boolean circuit satisfiability and quadratic arithmetic programs for proving arithmetic circuit satisfiability. Lipmaa [Lip13] suggested more efficient quadratic span programs using error correcting codes, and Danezis, Fournet, Groth and Kohlweiss [DFGK14] refined quadratic span programs to square span programs that give NIZK arguments consisting of 4 group elements for boolean circuit satisfiability.

Following these theoretical advances there has been exciting work on building concrete implementations of SNARKs [PHGR13,BCG$^+$13,BCTV14b,CTV15,CFH$^+$15]. Most efficient implementations refine the quadratic arithmetic program approach of Gennaro et al. [GGPR13] and combine it with a compiler producing a suitable quadratic arithmetic program that is equivalent to the statement to be proven; libsnark [BCTV14b,BSCG$^+$14] also includes a SNARK based on [DFGK14].

One powerful motivation for building efficient non-interactive arguments is verifiable computation. A client can outsource a complicated computational task to a server in the cloud and get back the results. To convince the client that the computation is correct the server may include a non-interactive argument of correctness with the result. However, since the verifier does not have many computational resources this only makes sense if the argument is compact and computationally light to verify, i.e., it is a succinct non-interactive argument (SNARG) or a succinct non-interactive argument of knowledge (SNARK). While pairing-based SNARGs are efficient for the verifier, the computational overhead for the prover is still orders of magnitude too high to warrant use in outsourced computation [Wal15] and further efficiency improvements are needed. In their current state, SNARKs that are zero-knowledge already have uses when proving statements

about private data though. Zero-knowledge SNARKs are for instance key ingredients in the virtual currency proposals Pinocchio coin [DFKP13] and Zerocash [BCG+14].

In parallel with developments in pairing-based NIZK arguments there has been interesting work on understanding SNARKs. Gentry and Wichs [GW11] showed that SNARGs must necessarily rely on non-falsifiable assumptions, and Bitansky et al. [BCCT12] proved designated verifier SNARKs exist if and only if extractable collision-resistant hash functions exist. Of particular interest in terms of efficiency is a series of works studying how SNARKs compose [Val08,BCCT13,BCTV14a]. They show among other things that a preprocessing SNARK with a long common reference string can be used to build a fully succinct SNARK with a short common reference string.

Bitansky et al. [BCI+13] give an abstract model of SNARKs that rely on linear encodings of field elements. Their information theoretic framework called linear interactive proofs (LIPs) capture proof systems where the prover is restricted to using linear operations in computing her messages. Given a LIP it can be converted to a publicly verifiable SNARK using pairing-based techniques or to a designated verifier using additively homomorphic encryption techniques.

1.1 Our contribution

Succinct NIZK. We construct a NIZK argument for arithmetic circuit satisfiability where a proof consists of only 3 group elements. In addition to being small, the proof is also easy to verify. The verifier just needs to compute a number of exponentiations proportional to the statement size and check a single pairing product equation, which only has 3 pairings. Our construction can be instantiated with any type of pairings including Type III pairings, which are the most efficient pairings.

The argument has perfect completeness and perfect zero-knowledge. For soundness we take an aggressive stance and rely on a security proof in the generic bilinear group model in order to get optimal performance. This stance is partly justified by Gentry and Wichs [GW11] that rule out SNARGs based on standard falsifiable assumptions. However, following Abe, Groth, Ohkubo and Tibouchi [AGOT14] we do provide a hedge against cryptanalysis by proving our construction secure in the symmetric pairing setting. For optimal efficiency it makes sense to use our NIZK argument in the asymmetric setting, however, by providing a security proof in the symmetric setting we get additional security: even if cryptanalytic advances yield a hitherto unknown efficiently computable isomorphism between the source groups this does not necessarily lead to a break of our scheme. We therefore have a unified NIZK argument that can be instantiated with any type of pairing, yielding both optimal efficiency and optimal generic bilinear group resilience.

We give a performance comparison for boolean circuit satisfiability in Table 1 and for arithmetic circuit satisfiability in Table 2 of the size of the common reference string (CRS), the size of the proof, the prover's computation, the verifier's computation, and the number of pairing product equations used to verify a proof. We perform better than the state of the art on all efficiency parameters.

In both comparisons the number of wires exceeds the number of gates, $m \geq n$, since each gate has an output wire. We expect for typical cases that the statement size ℓ will be small compared to m and n. In both tables, we have excluded the size of representing

	CRS size	Proof size	Prover comp.	Verifier comp.	PPE
[DFGK14]	$2m+n-2\ell$ $\mathbb{G}_1$, $m+n-\ell$ $\mathbb{G}_2$	3 $\mathbb{G}_1$, 1 $\mathbb{G}_2$	$m+n-\ell$ E_1	ℓ M_1 , 6 P	3
This work	$3m+n$ $\mathbb{G}_1$, m $\mathbb{G}_2$	2 $\mathbb{G}_1$, 1 $\mathbb{G}_2$	n E_1	ℓ M_1 , 3 P	1

Table 1. Comparison for boolean circuit satisfiability with ℓ-bit statement, m wires and n fan-in 2 logic gates. Notation: $\mathbb{G}$ means group elements, M means multiplications, E means exponentiations and P means pairings with subscripts indicating the relevant group. It is possible to get a CRS size of $m+2n$ elements in $\mathbb{G}_1$ and n elements in $\mathbb{G}_2$ but we have chosen to include some precomputed values in the CRS to reduce the prover's computation, see Sect. 3.2.

	CRS size	Proof size	Prover comp.	Verifier comp.	PPE
[PHGR13]	$7m+n-2\ell$ $\mathbb{G}$	8 $\mathbb{G}$	$7m+n-2\ell$ E	ℓ E , 11 P	5
This work	$m+2n$ $\mathbb{G}$	3 $\mathbb{G}$	$m+3n-\ell$ E	ℓ E , 3 P	1
[BCTV14a]	$6m+n+\ell$ $\mathbb{G}_1$, m $\mathbb{G}_2$	7 $\mathbb{G}_1$, 1 $\mathbb{G}_2$	$6m+n-\ell$ E_1 , m E_2	ℓ E_1 , 12 P	5
This work	$m+2n$ $\mathbb{G}_1$, n $\mathbb{G}_2$	2 $\mathbb{G}_1$, 1 $\mathbb{G}_2$	$m+3n-\ell$ E_1 , n E_2	ℓ E_1 , 3 P	1

Table 2. Comparison for arithmetic circuit satisfiability with ℓ-element statement, m wires, n multiplication gates. Notation: $\mathbb{G}$ means group elements, E means exponentiations and P means pairings. We compare symmetric pairings in the first two rows and asymmetric pairings in the last two rows.

the relation for which we give proofs. In the boolean circuit satisfiability case, we are considering arbitrary fan-in 2 logic gates. In the arithmetic circuit satisfiability case we work with fan-in 2 multiplication gates where each input factor can be a weigthed sum of other wires. We assume each multiplication gate input depends on a constant number of wires; otherwise the cost of evaluating the relation itself may exceed the cost of the subsequent proof generation.

We note that [PHGR13] uses symmetric bilinear groups where $\mathbb{G}_1 = \mathbb{G}_2$ and we are therefore comparing with a symmetric bilinear group instantiation of our scheme, which saves n elements in the common reference string. However, in the implementation of their system, called Pinocchio, asymmetric pairings are used for better efficiency. The switch to asymmetric pairings only requires minor modifications, see e.g. [BCTV14a] for a specification of such a SNARK, which has been implemented in the libsnark library.

Size matters. While the reduction in proof size to 3 group elements and the reduction in verification time is nice in itself, we would like to highlight that it is particularly important when composing SNARKs. [BCCT13,BCTV14a] show that preprocessing SNARKs with a long CRS can be composed to yield fully succinct SNARKs with a short CRS.[1] The transformations split the statement into smaller pieces, prove each piece is correct by itself, and recursively construct proofs of knowledge of other proofs that jointly show the pieces are correct and fit together. In the recursive construction of proofs, it is extra beneficial when the proofs are small and easy to verify since the resulting statements "there exists a proof satisfying the verification equation..." become small themselves. So we gain both from the prover's lower computation and from the

[1] We remark that soundness against generic adversaries is not preserved under composition (an issue that also appears in [Val08]), since the composition needs a concrete instantiation of the bilinear groups when writing out the statements corresponding to verification of another SNARK. What we do claim is that if our SNARK is knowledge sound in the standard model, then we get secure composition.

fact that the statements in the recursive composition are smaller since we have a more efficient verification procedure for our SNARK. Chiesa and Virza [CV16] report a factor 4-5 speedup from using our SNARKs in the implementation of [BCTV14a].

Technique. All pairing-based SNARKs in the literature follow a common paradigm where the prover computes a number of group elements using generic group operations and the verifier checks the proof using a number of pairing product equations. Bitansky et al. [BCI+13] formalize this paradigm through the definition of linear interactive proofs (LIPs). A linear interactive proof works over a finite field and the prover's and verifier's messages consist of vectors of field elements. It furthermore requires that the prover computes her messages using only linear operations. Once we have the LIP, it can then be compiled into a SNARK by executing the equations "in the exponent" using pairing-based cryptography. One source of our efficiency gain is that we design a LIP system for arithmetic circuits where the prover only sends 3 field elements. In comparison, the quadratic arithmetic programs by [GGPR13,PHGR13] correspond to LIPs where the prover sends 4 field elements.

A second source of efficiency gain compared to previous work is a more aggressive compilation of the LIP. Bitansky et al. [BCI+13] propose a transformation in the symmetric bilinear group setting, where each field element gets compiled into two group elements. They then use a knowledge of exponent assumption to argue that the prover knows the relevant field elements. A less conservative choice would be to compile each field element into a single group element. This improves efficiency but security requires stronger assumptions since we the scheme may be secure in the generic group model [Sho97,BBS04] but we can no longer use the knowledge of exponent assumption. It is also possible to make a choice between these two extremes, Parno et al. [PHGR13] for instance have a LIP with 4 field elements, which gets compiled into 7 group elements. In this paper we have opted for maximal efficiency and compile each field element in the LIP into a single group element and argue security in the generic group model.

We prefer to work with asymmetric bilinear groups for their higher efficiency than symmetric bilinear groups. This means that there is more to the story than the number of field elements the prover sends in the LIP and the choice of how aggressive a compilation we use. When working with asymmetric bilinear groups, a field element can appear as an exponent in the first source group, the second source group, or both. Our LIP is carefully designed such that each field element gets compiled into a single source group element in order to minimize the proof size to 3 group elements in total.

Lower bounds. Working towards ever more efficient non-interactive arguments, it is natural to ask what the minimal proof size is. We will show that pairing-based SNARGs with a single group element proof cannot exist. This result relates to an open question raised by Bitansky et al. [BCI+13], whether there are LIPs with a linear decision procedure for the verifier. Such a linear decision procedure would be quite useful; it could for instance enable the construction of SNARGs based on ElGamal encryption.

We answer this open problem negatively by proving that LIPs with a linear decision procedure do not exist. A consequence of this is that any pairing-based SNARG must pair group elements from the proof together to make the decision procedure quadratic instead of linear. Working over asymmetric bilinear groups we must therefore have

elements in both source groups in order to do such a pairing. This rules out the existence of 1 group element SNARGs, regardless of whether it is zero-knowledge or not, and shows our NIZK argument has close to optimal proof size. It remains an intriguing open problem to completely close the gap by either constructing a SNARG with exactly one element from each source group $\mathbb{G}_1$ and $\mathbb{G}_2$, or alternatively rule out the existence of such a SNARG.

2 Preliminaries

Given two functions $f, g : \mathbb{N} \to [0, 1]$ we write $f(\lambda) \approx g(\lambda)$ when $|f(\lambda) - g(\lambda)| = \lambda^{-\omega(1)}$. We say that f is *negligible* when $f(\lambda) \approx 0$ and that f is *overwhelming* when $f(\lambda) \approx 1$. We will use λ to denote a security parameter, with the intuition that as λ grows we would like to have stronger security.

We write $y = A(x; r)$ when algorithm A on input x and randomness r, outputs y. We write $y \leftarrow A(x)$ for the process of picking randomness r at random and setting $y = A(x; r)$. We also write $y \leftarrow S$ for sampling y uniformly at random from the set S. We will assume it is possible to sample uniformly at random from sets such as $\mathbb{Z}_p$.

Following Abe and Fehr [AF07] we write $(y; z) \leftarrow (\mathcal{A} \parallel \mathcal{X}_{\mathcal{A}})(x)$ when $\mathcal{A}$ on input x outputs y and $\mathcal{X}_{\mathcal{A}}$ on the same input (including random coins) outputs z.

2.1 Bilinear groups

We work with bilinear groups $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ with the following properties:

- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order p
- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map, i.e., $e(U^a, V^b) = e(U, V)^{ab}$
- If G is a generator for $\mathbb{G}_1$ and H is a generator for $\mathbb{G}_2$ then $e(G, H)$ is a generator for $\mathbb{G}_T$
- There are efficient algorithms for computing group operations, evaluating the bilinear map, deciding membership of the groups, deciding equality of group elements and sampling generators of the groups. We refer to these as the generic bilinear group operations.

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 $\mathbb{G}_1 \neq \mathbb{G}_2$. Galbraith, Paterson and Smart [GPS08] classify bilinear groups as Type I where $\mathbb{G}_1 = \mathbb{G}_2$, Type II where there is an efficiently computable non-trivial homomorphism $\Psi : \mathbb{G}_2 \to \mathbb{G}_1$, and Type III where no such efficiently computable homomorphism exists 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. We give the lower bound for Type III bilinear groups and but our construction works without change for all 3 types of bilinear groups.

2.2 Non-interactive zero-knowledge arguments of knowledge

Let $\mathcal{R}$ be a relation generator that given a security parameter λ in unary returns a polynomial time decidable binary relation R. For pairs $(\phi, w) \in R$ we call ϕ the statement and w the witness. We define $\mathcal{R}_\lambda$ to be the set of possible relation $\mathcal{R}$ may output

given 1^λ. The relation generator may also output some side information, an auxiliary input z, which will be given to the adversary. An efficient prover publicly verifiable non-interactive argument for $\mathcal{R}$ is a quadruple of probabilistic polynomial algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vfy}, \mathsf{Sim})$ such that

$(\sigma, \tau) \leftarrow \mathsf{Setup}(R)$**:** The setup takes as input a security parameter λ and a relation $R \in \mathcal{R}_\lambda$ and returns a common reference string σ and a simulation trapdoor τ for the relation R.

$\pi \leftarrow \mathsf{Prove}(R, \sigma, \phi, w)$**:** The prover algorithm takes as input a common reference string σ and $(\phi, w) \in R$ and returns an argument π.

$0/1 \leftarrow \mathsf{Vfy}(R, \sigma, \phi, \pi)$**:** The verification algorithm takes as input a common reference string σ, a statement ϕ and an argument π and returns 0 (reject) or 1 (accept).

$\pi \leftarrow \mathsf{Sim}(R, \tau, \phi)$**:** The simulator takes as input a simulation trapdoor and statement ϕ and returns an argument π.

Definition 1. *We say* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vfy})$ *is a non-interactive argument for* $\mathcal{R}$ *if it has perfect completeness and computational soundness as defined below.*

Definition 2. *We say* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vfy}, \mathsf{Sim})$ *is a perfect non-interactive zero-knowledge argument of knowledge for* $\mathcal{R}$ *if it has perfect completeness, perfect zero-knowledge and computational knowledge soundness as defined below.*

Perfect completeness. Completeness says that, given any true statement, an honest prover should be able to convince an honest verifier. For all $\lambda \in \mathbb{N}$, $R \in \mathcal{R}_\lambda$, $(\phi, w) \in R$

$$\Pr\left[(\sigma, \tau) \leftarrow \mathsf{Setup}(R); \pi \leftarrow \mathsf{Prove}(R, \sigma, \phi, w) : \mathsf{Vfy}(R, \sigma, \phi, \pi) = 1\right] = 1.$$

Perfect zero-knowledge. An argument is zero-knowledge if it does not leak any information besides the truth of the statement. We say $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vfy}, \mathsf{Sim})$ is perfect zero-knowledge if for all $\lambda \in \mathbb{N}$, $(R, z) \leftarrow \mathcal{R}(1^\lambda)$, $(\phi, w) \in R$ and all adversaries $\mathcal{A}$

$$\begin{aligned} &\Pr\left[(\sigma, \tau) \leftarrow \mathsf{Setup}(R); \pi \leftarrow \mathsf{Prove}(R, \sigma, \phi, w) : \mathcal{A}(R, z, \sigma, \tau, \pi) = 1\right] \\ =\ &\Pr\left[(\sigma, \tau) \leftarrow \mathsf{Setup}(R); \pi \leftarrow \mathsf{Sim}(R, \tau, \phi) : \mathcal{A}(R, z, \sigma, \tau, \pi) = 1\right]. \end{aligned}$$

Computational soundness. We say $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vfy}, \mathsf{Sim})$ is sound if it is not possible to prove a false statement, i.e., convince the verifier if no witness exists. Let L_R be the language consisting of statements for which there exist matching witnesses in R. Formally, we require that for all non-uniform polynomial time adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{c}(R, z) \leftarrow \mathcal{R}(1^\lambda); (\sigma, \tau) \leftarrow \mathsf{Setup}(R); (\phi, \pi) \leftarrow \mathcal{A}(R, z, \sigma) : \\ \phi \notin L_R \text{ and } \mathsf{Vfy}(R, \sigma, \phi, \pi) = 1\end{array}\right] \approx 0.$$

Computational knowledge soundness. Strengthening the notion of soundness, we call $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vfy}, \mathsf{Sim})$ an argument of knowledge if there is an extractor that can compute a witness whenever the adversary produces a valid argument. The extractor

gets full access to the adversary's state, including any random coins. Formally, we require that for all non-uniform polynomial time adversaries $\mathcal{A}$ there exists a non-uniform polynomial time extractor $\mathcal{X}_{\mathcal{A}}$ such that

$$\Pr\left[\begin{array}{c}(R,z) \leftarrow \mathcal{R}(1^\lambda); (\sigma,\tau) \leftarrow \mathsf{Setup}(R); ((\phi,\pi);w) \leftarrow (\mathcal{A} \parallel \mathcal{X}_{\mathcal{A}})(R,z,\sigma) : \\ (\phi,w) \notin R \text{ and } \mathsf{Vfy}(R,\sigma,\phi,\pi) = 1\end{array}\right] \approx 0.$$

Public verifiability and designated verifier proofs. We can naturally generalize the definition of a non-interactive argument by splitting σ into two parts σ_P and σ_V used by the prover and verifier respectively. We say the non-interactive argument is publicly verifiable when σ_V can be deduced from σ_P. Otherwise we refer to it as a designated verifier argument. For designated verifier arguments it is possible to relax soundness and knowledge soundness such that the adversary only sees σ_P but not σ_V.

SNARGs and SNARKs. A non-interactive argument where the verifier runs in polynomial time in $\lambda + |\phi|$ and the proof size is polynomial in λ is called a preprocessing succinct non-interactive argument (SNARG) if it sound, and a preprocessing succinct argument of knowledge (SNARK) if it is knowledge sound. If we also restrict the common reference string to be polynomial in λ we say the non-interactive argument is a fully succinct SNARG or SNARK. Bitansky et al. [BCCT13] show that preprocessing SNARKs can be composed to yield fully succinct SNARKs. The focus of this paper is on preprocessing SNARKs.

Benign relation generators. Bitansky et al. [BCPR14] show that indistinguishability obfuscation implies that for every candidate SNARK there are auxiliary output distributions that enable the adversary to create a valid proof without it being possible to extract the witness. Assuming also public coin differing input obfuscation and other cryptographic assumptions, Boyle and Pass [BP15] strengthen this impossibility to show that there is an auxiliary output distribution that defeats witness extraction for all candidate SNARKs. These counter examples, however, rely on specific auxiliary input distributions. We will therefore in the following assume the relationship generator is *benign* in the sense that the relation and the auxiliary input are distributed in such a way that SNARKs can exist.

2.3 Quadratic arithmetic programs

Consider an arithmetic circuit consisting of addition and multiplication gates over a finite field $\mathbb{F}$. We may designate some of the input/output wires as specifying a statement and use the rest of the wires in the circuit to define a witness. This gives us a binary relation R consisting of statement wires and witness wires that satisfy the arithmetic circuit, i.e., make it consistent with the designated input/output wires.

Generalizing arithmetic circuits, we may be interested in relations described by equations over a set of variables. Some of the variables correspond to the statement; the remaining variables correspond to the witness. The relation consists of statements and witnesses that satisfy all the equations. The equations will be over $a_0 = 1$ and variables $a_1, \ldots, a_m \in \mathbb{F}$ and be of the form

$$\sum a_i u_{i,q} \cdot \sum a_i v_{i,q} = \sum a_i w_{i,q},$$

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

We observe that addition and multiplication gates are special cases of such equations so such systems of arithmetic constraints do indeed generalize arithmetic circuits. A multiplication gate can for instance be described as $a_i \cdot a_j = a_k$ (using $u_i = 1, v_j = 1$ and $w_k = 1$ and setting the remaining constants for this gate to 0). Addition gates are handled for free in the sums defining the equations, i.e., if $a_i + a_j = a_k$ and a_k is multiplied by a_ℓ, we may simply write $(a_i + a_j) \cdot a_\ell$ and skip the calculation of a_k.

Following Gennaro, Gentry, Parno and Raykova [GGPR13] we can reformulate the set of arithmetic constraints as a quadratic arithmetic program assuming $\mathbb{F}$ is large enough. Given n equations we pick arbitrary distinct $r_1, \ldots, r_n \in \mathbb{F}$ and define $t(x) = \prod_{q=1}^{n}(x - r_q)$. Furthermore, let $u_i(x), v_i(x), w_i(x)$ be degree $n-1$ polynomials such that

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

We now have that $a_0 = 1$ and the variables $a_1, \ldots, a_m \in \mathbb{F}$ satisfy the n equations if and only if in each point $r_1, \ldots, r_q$

$$\sum_{i=0}^{m} a_i u_i(r_q) \cdot \sum_{i=0}^{m} a_i v_i(r_q) = \sum_{i=0}^{m} a_i w_i(r_q).$$

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

$$\sum_{i=0}^{m} a_i u_i(X) \cdot \sum_{i=0}^{m} a_i v_i(X) \equiv \sum_{i=0}^{m} a_i w_i(X) \mod t(X).$$

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

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

where $\mathbb{F}$ describes a finite field, aux is some auxiliary information, $1 \leq \ell \leq m$, $u_i(X), v_i(X), w_i(X), t(X) \in \mathbb{F}[X]$ and $u_i(X), v_i(X), w_i(X)$ have strictly lower degree than n, the degree of $t(X)$. A quadratic arithmetic program with such a description defines the following binary relation, where we define $a_0 = 1$,

$$R = \left\{ (\phi, w) \middle| \begin{array}{l} \phi = (a_1, \ldots, a_\ell) \in \mathbb{F}^\ell \\ w = (a_{\ell+1}, \ldots, a_m) \in \mathbb{F}^{m-\ell} \\ \\ \sum_{i=0}^{m} a_i u_i(X) \cdot \sum_{i=0}^{m} a_i v_i(X) \equiv \sum_{i=0}^{m} a_i w_i(X) \mod t(X) \end{array} \right\}.$$

We say $\mathcal{R}$ is a quadratic arithmetic program generator if it generates relations of the form given above with fields of size larger than $2^{\lambda-1}$.

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 field $\mathbb{F}$ is generated and then the rest of the relation is built on top of the field. Or it may be that the polynomials are specified first and then a random field 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.

Looking ahead, we will in our pairing-based NIZK arguments let the auxiliary information aux specify a bilinear group. 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.

2.4 Linear interactive proofs

Bitansky et al. [BCI$^+$13] give a useful characterization of the information theoretic underpinning of recent SNARK constructions. A two-move algebraic linear interactive proof (LIP) of degree (d_Q, d_D) for a relation generator $\mathcal{R}$, where we assume the relations specify a finite field $\mathbb{F}$, is a non-interactive argument system where the algorithms work as follows:

$(\boldsymbol{\sigma}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R)$**:** The setup first runs a deterministic polynomial time algorithm to create an arithmetic circuit that computes polynomials of total degree bounded by d_Q. The circuit then takes as input randomness $\boldsymbol{r} \in \mathbb{F}^\mu$ and returns vectors $\boldsymbol{\sigma} \in \mathbb{F}^m$ and $\boldsymbol{\tau} \in \mathbb{F}^n$. We will for notational simplicity assume that $\boldsymbol{\sigma}$ always contains 1 as an entry such that there is no distinction between affine and linear functions of $\boldsymbol{\sigma}$.

$\boldsymbol{\pi} \leftarrow \mathsf{Prove}(R, \boldsymbol{\sigma}, \phi, w)$**:** The prover operates in two stages:

- First it runs $\Pi \leftarrow \mathsf{ProofMatrix}(R, \phi, w)$, where $\mathsf{ProofMatrix}$ is a probabilistic polynomial time algorithm that generates a matrix $\Pi \in \mathbb{F}^{k \times m}$.
- Then it computes the proof as $\boldsymbol{\pi} = \Pi\boldsymbol{\sigma}$.

$0/1 \leftarrow \mathsf{Vfy}(R, \boldsymbol{\sigma}, \phi, \boldsymbol{\pi})$**:** The verifier runs in two stages:

- First it runs a deterministic polynomial time algorithm $\boldsymbol{t} \leftarrow \mathsf{Test}(R, \phi)$ to get an arithmetic circuit $\boldsymbol{t} : \mathbb{F}^{m+k} \rightarrow \mathbb{F}^\eta$ corresponding to the evaluation of polynomials of total degree d_D.
- It then accepts the proof if and only if $\boldsymbol{t}(\boldsymbol{\sigma}, \boldsymbol{\pi}) = \mathbf{0}$.

The degrees and dimensions $d_Q, d_D, \mu, m, n, k, \eta$ may be constants or polynomials in the security parameter λ.

Definition 3 (Linear interactive proof). *The tuple* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vfy})$ *is a linear interactive proof for* $\mathcal{R}$ *if it has perfect completeness and statistical knowledge soundness against affine prover strategies as defined below.*

Statistical knowledge soundness against affine prover strategies. An LIP has knowledge soundness against affine prover strategies if a witness can be extracted from a successful proof matrix Π. More precisely, there is a polynomial time extractor $\mathcal{X}$ such that for all adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{c}(R, z) \leftarrow \mathcal{R}(1^\lambda); (\boldsymbol{\sigma}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R); (\phi, \Pi) \leftarrow \mathcal{A}(R, z); w \leftarrow \mathcal{X}(R, \phi, \Pi) : \\ \Pi \in \mathbb{F}^{m \times k} \ \wedge \ \mathsf{Vfy}(R, \boldsymbol{\sigma}, \phi, \Pi\boldsymbol{\sigma}) = \mathbf{0} \ \wedge \ (\phi, w) \notin R\end{array}\right] \approx 0.$$

2.5 Non-interactive arguments from linear interactive proofs.

LIPs are useful concepts because they can be compiled into publicly verifiable non-interactive arguments using pairings and designated verifier non-interactive arguments using Paillier encryption [BCI+13]. If we work in the pairing setting, the intuition is that an algebraic LIP of degree $(\text{poly}(\lambda), 2)$ can be executed "in the exponents": The common reference string contains exponentiations of the field elements in $\boldsymbol{\sigma}$. The prover computes the proof as multi-exponentiations of group elements, corresponding to linear operations on the field elements in $\boldsymbol{\sigma}$. The verifier checks the argument by verifying a number of pairing product equations (equations formed by multiplying together the results of pairings), which corresponds to checking quadratic equations in the exponents. We will now formalize this methodology.

When working with Type III pairings, executing the LIP in the exponents requires that we specify for each element in which group the exponentiation should take place. We will therefore for the sake of Type III pairings define a special case of LIPs, which we will call a split LIP. A split LIP is a 2-message algebraic LIP, where the verifier's first message can be split into two parts $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ and the prover's proof can be split into two parts $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$. Each part of the proof is computed from the corresponding part of the verifier's message. Finally, when verifying the proof, we want the verifier's test to be a quadratic equation where each variable has degree 1. Writing it out, a split LIP is a 2-message algebraic LIP of degree $(d_Q, 2)$, of the following form:

$(\boldsymbol{\sigma}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R)$**:** It creates an arithmetic circuit of multiplicative depth d_Q that takes as input randomness $\boldsymbol{r} \in \mathbb{F}^{\mu}$ and returns vectors $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2) \in \mathbb{F}^{m_1} \times \mathbb{F}^{m_2}$ and $\boldsymbol{\tau} \in \mathbb{F}^{n}$. We will for notational simplicity assume that $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ both contain 1 as an entry such that there is no distinction between affine and linear functions of $\boldsymbol{\sigma}$.

$\boldsymbol{\pi} \leftarrow \mathsf{Prove}(R, \boldsymbol{\sigma}, \phi, w)$**:** The prover operates in two stages:

- First it runs $\Pi \leftarrow \mathsf{ProofMatrix}(R, \phi, w)$, where $\mathsf{ProofMatrix}$ is a probabilistic polynomial time algorithm that generates a matrix $\Pi = \begin{pmatrix} \Pi_1 & 0 \\ 0 & \Pi_2 \end{pmatrix}$, where $\Pi_1 \in \mathbb{F}^{k_1 \times m_1}$ and $\Pi_2 \in \mathbb{F}^{k_2 \times m_2}$.
- Then it computes $\boldsymbol{\pi}_1 = \Pi_1 \boldsymbol{\sigma}_1$ and $\boldsymbol{\pi}_2 = \Pi_2 \boldsymbol{\sigma}_2$ and returns $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$.

$0/1 \leftarrow \mathsf{Vfy}(R, \boldsymbol{\sigma}, \phi, \boldsymbol{\pi})$**:** The verifier runs in two stages:

- First it runs a deterministic polynomial time algorithm $\boldsymbol{t} \leftarrow \mathsf{Test}(R, \phi)$ to get an arithmetic circuit $\boldsymbol{t} : \mathbb{F}^{m_1+k_1+m_2+k_2} \to \mathbb{F}^{\eta}$ corresponding to matrices $T_1, \ldots, T_\eta \in \mathbb{F}^{(m_1+k_1)\times(m_2+k_2)}$.
- It then accepts the proof if and only if for all matrices $T_1, \ldots, T_\eta$

$$(\boldsymbol{\sigma}_1^\top, \boldsymbol{\pi}_2^\top) T_i \begin{pmatrix} \boldsymbol{\sigma}_2 \\ \boldsymbol{\pi}_2 \end{pmatrix} = 0.$$

Intuitively, we want to argue soundness by saying a prover that uses generic group operations cannot deviate from the LIP. However, when the prover sees a real common reference string, she may learn useful information from it and choose her matrix Π in a way that depends on it. To counter this type of prover, we will define a *disclosure-free* common reference string as one where the prover does not gain useful information that can help her choose a special matrix Π.

Definition 4 (Disclosure-free LIP). *We say a LIP is disclosure-free if for all adversaries $\mathcal{A}$*

$$\Pr\left[(R,z) \leftarrow \mathcal{R}(1^\lambda); t \leftarrow \mathcal{A}(R,z); (\boldsymbol{\sigma},\boldsymbol{\tau}), (\boldsymbol{\sigma}',\boldsymbol{\tau}') \leftarrow \mathsf{Setup}(R) : t(\boldsymbol{\sigma}) = 0 \text{ and } t(\boldsymbol{\sigma}') \neq 0\right] \approx 0,$$

where $t : \mathbb{F}^m \to \mathbb{F}$ is a degree d_t arithmetic circuit.

The way to interpret the definition is that the outcome of any degree d_t test the prover may run on $\boldsymbol{\sigma}$ can be predicted by running it on an independently generated $\boldsymbol{\sigma}'$. This means that degree d_t tests do not disclose any non-trivial information to the prover about the verifier's first message. In the case of using executing an LIP in the exponent of bilinear groups, we will need disclosure-freeness against tests t of degree $d_t \leq 2$.

We are now ready to define a compiler of a disclosure-free 2-message algebraic LIP of degree $(d_Q, 2)$ for a relation R and returns a ...???

$(\sigma,\tau) \leftarrow \mathsf{Setup}(R)$**:** Run $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R)$. Compute

$$\boldsymbol{\Sigma}_1 = G^{\boldsymbol{\sigma}_1} \qquad\qquad \boldsymbol{\Sigma}_2 = H^{\boldsymbol{\sigma}_2}$$

and return common reference string $\sigma = (\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ and simulation trapdoor $\tau = \boldsymbol{\tau}$.

$\pi \leftarrow \mathsf{Prove}(R,\sigma,\phi,w)$**:** Generate $(\Pi_1, \Pi_2) \leftarrow \mathsf{ProofMatrix}(R,x,w)$ and return $\pi = (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$ computed as

$$\boldsymbol{\pi}_1 = \Sigma_1^{\Pi_1} \qquad\qquad \boldsymbol{\pi}_2 = \Sigma_2^{\Pi_2}.$$

$0/1 \leftarrow \mathsf{Vfy}(R,\sigma,\phi,\pi)$**:** Generate $(T_1,\ldots,T_\eta) \leftarrow \mathsf{Test}(R,\phi)$. Parse $\pi = (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) \in \mathbb{G}_1^{k_1} \times \mathbb{G}_2^{k_2}$. Accept the proof if and only if for all $T_1,\ldots,T_\eta$

$$e((\Sigma_1^\top)^{T_i}, \Sigma_2) = 1.$$

$\pi \leftarrow \mathsf{Sim}(R,\tau,\phi)$**:** Simulate $(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) \leftarrow \mathsf{Sim}(R,\boldsymbol{\tau},\phi)$ and return $\pi = (G^{\boldsymbol{\pi}_1}, H^{\boldsymbol{\pi}_2})$.

Theorem 1. *The protocol given above is a non-interactive zero-knowledge argument with perfect completeness and perfect zero-knowledge. It has statistical knowledge soundness against adversaries that only use a polynomial number of generic bilinear group operations.*

Proof. Perfect completeness follows from the perfect completeness of the LIP and the fact it is a split LIP, which allows the adversary to compute the two parts of the proof $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2$ using only generic group operations in the relevant groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

Perfect zero-knowledge follows from the perfect zero-knowledge property of the LIP.

It remains to argue statistical soundness against and generic adversaries. A generic adversary can use the generic group operations to multiply elements in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, test membership of the groups, and evaluate the pairing.

Given the common reference string $\sigma = (\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ the adversary may try to learn something about the common reference string by multiplying together group elements in $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, respectively, and then compute products of pairings of these elements. This corresponds to conducting tests of the form $e(\boldsymbol{\Sigma}_1^T, \boldsymbol{\Sigma}_2) = 1$ for matrices $T \in \mathbb{Z}_p^{m_1 \times m_2}$. Taking discrete logarithms, we see this corresponds to tests of the form $\boldsymbol{\sigma}_1^\top T \boldsymbol{\sigma}_2 = 0$. By the disclosure-freeness, the adversary could conduct these tests on an independently

chosen common reference string $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2) \leftarrow \mathsf{Setup}(R)$ and with overwhelming probability get the same answers. An adversary that only uses a polynomial number of generic bilinear group operations therefore has negligible chance of learning any useful information from the common reference string, and we can without loss of generality assume the adversary does not test the common reference string.

The adversary that creates the proof using generic group operations, corresponds to an adversary that picks matrices Π_1, Π_2 and computes the proof as $\pi = (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$ with $\boldsymbol{\pi}_1 = \boldsymbol{\Sigma}_1^{\Pi_1}, \boldsymbol{\pi}_2 = \boldsymbol{\sigma}_2^{\Pi_2}$. When the adversary does not test the common reference string, this adversary must compute Π_1 and Π_2 independently of $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$, so they can only depend on R and ϕ.

Taking discrete logarithms of the verification equation, we see that this corresponds to finding ϕ and Π_1, Π_2 such that for the test matrices $T_1, \ldots, T_\eta \leftarrow \mathsf{Test}(R, \phi)$

$$(\boldsymbol{\sigma}_1^\top, (\Pi_1 \boldsymbol{\sigma}_1)^\top) T_i \begin{pmatrix} \boldsymbol{\sigma}_2 \\ \Pi_2 \boldsymbol{\sigma}_2 \end{pmatrix} = 0.$$

By the statistical soundness of the split LIP this has negligible probability of happening. □

3 Constructions of non-interactive arguments

We will construct a pairing-based NIZK argument for quadratic arithmetic programs where proofs consist of only 3 group elements. We give the construction in two steps, first we construct a LIP, and then we convert the LIP into a pairing-based NIZK argument.

3.1 Linear interactive proofs for quadratic arithmetic programs

We will now construct a LIP for quadratic arithmetic program generators that outputs relations of the form

$$R = (\mathbb{F}, \mathrm{aux}, \ell, \{u_i(X), v_i(X), w_i(X)\}_{i=0}^m, t(X)) .$$

The relation defines a language of statements $(a_1, \ldots, a_\ell) \in \mathbb{F}^\ell$ and witnesses $(a_{\ell+1}, \ldots, a_m) \in \mathbb{F}^{m-\ell}$ such that with $a_0 = 1$

$$\sum_{i=0}^m a_i u_i(X) \cdot \sum_{i=0}^m a_i v_i(X) = \sum_{i=0}^m a_i w_i(X) + h(X)t(X),$$

for some degree $n-2$ quotient polynomial $h(X)$, where n is the degree of $t(X)$.

$(\boldsymbol{\sigma}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R)$**:** Pick $\alpha, \beta, \gamma, \delta, x \leftarrow \mathbb{F}^*$. Set $\boldsymbol{\tau} = (\alpha, \beta, \gamma, \delta, x)$ and

$$\boldsymbol{\sigma} = \left(\alpha, \beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}, \left\{ \tfrac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} \right\}_{i=0}^{\ell}, \left\{ \tfrac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} \right\}_{i=\ell+1}^{m}, \left\{ \tfrac{x^i t(x)}{\delta} \right\}_{i=0}^{n-2} \right).$$

$\boldsymbol{\pi} \leftarrow \mathsf{Prove}(R, \boldsymbol{\sigma}, a_1, \ldots, a_m)$**:** Pick $r, s \leftarrow \mathbb{F}$ and compute a $3 \times (m + 2n + 4)$ matrix Π such that $\boldsymbol{\pi} = \Pi \boldsymbol{\sigma} = (A, B, C)$ where

$$A = \alpha + \sum_{i=0}^m a_i u_i(x) + r\delta \qquad B = \beta + \sum_{i=0}^m a_i v_i(x) + s\delta$$

$$C = \frac{\sum_{i=\ell+1}^m a_i \left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right) + h(x)t(x)}{\delta} + As + rB - rs\delta.$$

$0/1 \leftarrow \mathsf{Vfy}(R, \boldsymbol{\sigma}, a_1, \ldots, a_\ell)$**:** Compute a quadratic multi-variate polynomial $\boldsymbol{t}$ such that $\boldsymbol{t}(\boldsymbol{\sigma}, \boldsymbol{\pi}) = 0$ corresponds to the test

$$A \cdot B = \alpha \cdot \beta + \frac{\sum_{i=0}^{\ell} a_i \left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right)}{\gamma} \cdot \gamma + C \cdot \delta.$$

Accept the proof if the test passes.

$\boldsymbol{\pi} \leftarrow \mathsf{Sim}(R, \boldsymbol{\tau}, a_1, \ldots, a_\ell)$**:** Pick $A, B \leftarrow \mathbb{F}$ and compute $C = \frac{AB - \alpha\beta - \sum_{i=0}^{\ell} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\delta}$. Return $\boldsymbol{\pi} = (A, B, C)$.

Before formally proving this is a LIP, let us give a little intuition behind the different components. The role of α and β is to ensure A, B and C are consistent with each other in the choice of $a_0, \ldots, a_m$. The product $\alpha \cdot \beta$ in the verification equation guarantees that A and B involve non-trivial α and β components. This means the product $A \cdot B$ involves a linear dependence on α and β, and we will later prove that this linear dependence can only be balanced out by C with a consistent choice of $a_0, \ldots, a_m$ in all three of A, B and C. The role of γ and δ is to make the two latter products of the verification equation independent from the first product, by dividing the left factors with γ and δ respectively. This prevents mixing and matching of elements intended for different products in the verification equation. Finally, we use r and s to randomize the proof to get zero-knowledge.

Theorem 2. *The construction above yields a LIP with perfect completeness, perfect zero-knowledge and statistical knowledge soundness against affine prover strategies.*

Proof. Perfect completeness is straightforward to verify. Perfect zero-knowledge follows from both real proofs and simulated proofs having uniformly random field elements A, B. These elements uniquely determine C through the verification equation, so real proofs and simulated proofs have identical probability distributions.

What remains is to demonstrate that for any affine prover strategy with non-negligible success probability we can extract a witness. When using an affine prover strategy we have

$$\begin{aligned} A = A_\alpha\alpha + A_\beta\beta + A_\gamma\gamma + A_\delta\delta + A(x) + \sum_{i=0}^{\ell} A_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} \\ + \sum_{i=\ell+1}^{m} A_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + A_h(x)\frac{t(x)}{\delta}, \end{aligned}$$

for known field elements $A_\alpha, A_\beta, A_\gamma, A_\delta, A_i$ and polynomials $A(x), A_h(x)$ of degrees $n-1$ and $n-2$, respectively that correspond to the first row of the matrix Π. We can write out B and C in a similar fashion from the second and third rows of Π.

We now view the verification equation as an equality of multi-variate Laurent polynomials. By the Schwartz-Zippel lemma the prover has negligible success probability unless the verification equation holds when viewing A, B and C as formal polynomials in indeterminates $\alpha, \beta, \gamma, \delta, x$.

The terms with indeterminate α^2 are $A_\alpha B_\alpha \alpha^2 = 0$, which means $A_\alpha = 0$ or $B_\alpha = 0$. Since $AB = BA$ we can without loss of generality assume $B_\alpha = 0$. The terms with

indeterminate $\alpha\beta$ give us $A_\alpha B_\beta + A_\beta B_\alpha = A_\alpha B_\beta = 1$. This means $AB = (AB_\beta)(A_\alpha B)$ so we can without loss of generality after rescaling assume $A_\alpha = B_\beta = 1$. The terms with indeterminate β^2 now give us $A_\beta B_\beta = A_\beta = 0$. We have now simplified A and B constructed by the adversary to be of the form

$$A = \alpha + A_\gamma\gamma + A_\delta\delta + A(x) + \cdots \qquad B = \beta + B_\gamma\gamma + B_\delta\delta + B(x) + \cdots .$$

Next, let us consider the terms involving $\frac{1}{\delta^2}$. We have

$$\left(\sum_{i=\ell+1}^{m} A_i\left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right) + A_h(x)t(x)\right) \cdot \left(\sum_{i=\ell+1}^{m} B_i\left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right) + B_h(x)t(x)\right) = 0,$$

showing either the left factor is 0 or the right factor is 0. By symmetry, let us without loss of generality assume $\sum_{i=\ell+1}^{m} A_i\left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right) + t(x)A_t(x) = 0$. The terms in $\alpha\frac{\sum_{i=\ell+1}^{m} B_i(\beta u_i(x)+\alpha v_i(x)+w_i(x))+B_h(x)t(x)}{\delta} = 0$ now show us that also $\sum_{i=\ell+1}^{m} B_i\left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right) + B_h(x)t(x) = 0$.

The terms involving $\frac{1}{\gamma^2}$ give us

$$\sum_{i=0}^{\ell} A_i\left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right) \cdot \sum_{i=0}^{\ell} B_i\left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right) = 0,$$

showing either the left factor is 0 or the right factor is 0. By symmetry, let us without loss of generality assume $\sum_{i=0}^{\ell} A_i\left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right) = 0$. The terms in $\alpha\frac{\sum_{i=0}^{m} B_i(\beta u_i(x)+\alpha v_i(x)+w_i(x))}{\gamma} = 0$ now show us $\sum_{i=0}^{\ell} B_i\left(\beta u_i(x) + \alpha v_i(x) + w_i(x)\right) = 0$ as well.

The terms $A_\gamma\beta\gamma = 0$ and $B_\gamma\alpha\gamma = 0$ show us that $A_\gamma = 0$ and $B_\gamma = 0$. We now have

$$A = \alpha + A(x) + A_\delta\delta \qquad B = \beta + B(x) + B_\delta\delta.$$

The remaining terms in the verification equation that involve α give us $\alpha B(x) = \sum_{i=0}^{\ell} a_i\alpha v_i(x) + \sum_{i=\ell+1}^{m} C_i\alpha v_i(x)$. The terms involving β give us $\beta A(x) = \sum_{i=0}^{\ell} a_i\beta u_i(x) + \sum_{i=\ell+1}^{m} C_i\beta u_i(x)$. Defining $a_i = C_i$ for $i = \ell+1, \ldots, m$ we now have

$$A(x) = \sum_{i=0}^{m} a_i u_i(x) \qquad B(x) = \sum_{i=0}^{m} a_i v_i(x).$$

Finally, we look at the terms involving powers of x to get

$$\sum_{i=0}^{m} a_i u_i(x) \cdot \sum_{i=0}^{m} a_i v_i(x) = \sum_{i=0}^{m} a_i w_i(x) + C_h(x)t(x).$$

This shows that $(a_{\ell+1}, \ldots, a_m) = (C_{\ell+1}, \ldots, C_m)$ is a witness for the statement $(a_1, \ldots, a_\ell)$. □

2 FIELD ELEMENT LIPs. It is natural to ask whether the number of field elements the prover sends in the LIP can be reduced further. The square span programs of Danezis et al. [DFGK14] give rise to 2 field element LIPs for boolean circuit satisfiability. It is also possible to get a 2-element LIP for arithmetic circuit satisfiability by rewriting the circuit into one that only uses squaring gates, each multiplication gate $a \cdot b = c$ can be rewritten as a $(a+b)^2 - (a-b)^2 = 4c$. When an arithmetic circuit only has squaring gates we get $u_i(x) = v_i(x)$ for all i. By choosing $r = s$ in the LIP, we now have that $B = A + \beta - \alpha$, so the prover only needs to send two elements A and C to make a convincing proof. Rewriting the arithmetic circuit to only use squaring gates may double the number of gates and also requires some additional wires for the subtraction of the squares, so the reduction of the size of the LIP comes at a significant computational cost though.

3.2 NIZK arguments for quadratic arithmetic programs

We will now give a pairing-based NIZK argument for quadratic arithmetic programs. We consider relation generators $\mathcal{R}$ that return relations of the form

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

with $|p| = \lambda$. The relation defines a field $\mathbb{Z}_p$ and a language of statements $(a_1, \ldots, a_\ell) \in \mathbb{Z}_p^\ell$ and witnesses $(a_{\ell+1}, \ldots, a_m) \in \mathbb{Z}_p^{m-\ell}$ such that with $a_0 = 1$

$$\sum_{i=0}^m a_i u_i(X) \cdot \sum_{i=0}^m a_i v_i(X) = \sum_{i=0}^m a_i w_i(X) + h(X)t(X),$$

for some degree $n-2$ quotient polynomial $h(X)$.

We will construct the pairing-based argument by using the LIP from the previous section "in the exponents". An important design feature of the LIP is that the elements A, B and C are only used once in the verification equation and therefore it is easy to assign them to different source groups such that the verification equation can be carried out using a pairing product equation. Since pairing-friendly elliptic curves can be constructed such that the group element representations are smaller in $\mathbb{G}_1$ than in $\mathbb{G}_2$ [GPS08] we choose to assign A and C to the first source group and B to the second source group for maximal efficiency. This gives us the following NIZK argument.

$(\sigma, \tau) \leftarrow \mathsf{Setup}(R)$**:** Pick arbitrary generators G and H for $\mathbb{G}_1$ and $\mathbb{G}_2$. Pick $\alpha, \beta, \gamma, \delta, x \leftarrow \mathbb{Z}_p^*$. Define $\tau = (\alpha, \beta, \gamma, \delta, x)$ and compute

$$\sigma = \left(\begin{array}{c} G^\alpha, G^\beta, H^\beta, H^\gamma, G^\delta, H^\delta, \left\{G^{x^i}\right\}_{i=0}^{n-1}, \left\{H^{x^i}\right\}_{i=0}^{n-1} \\ \left\{G^{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}}\right\}_{i=0}^{\ell}, \left\{G^{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}}\right\}_{i=\ell+1}^{m}, \left\{G^{\frac{x^i t(x)}{\delta}}\right\}_{i=0}^{n-2} \end{array} \right).$$

$\pi \leftarrow \mathsf{Prove}(R, \sigma, a_1, \ldots, a_m)$**:** Pick $r, s \leftarrow \mathbb{Z}_p$ and compute $\pi = (A, B, C)$, where

$$A = G^{\alpha + \sum_{i=0}^m a_i u_i(x) + r\delta} \qquad B = H^{\beta + \sum_{i=0}^m a_i v_i(x) + s\delta}$$

$$C = G^{\frac{\sum_{i=\ell+1}^m a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + h(x)t(x)}{\delta} + s\left(\alpha + \sum_{i=0}^m a_i u_i(x)\right) + r\left(\beta + \sum_{i=0}^m a_i v_i(x)\right) + rs\delta}.$$

$0/1 \leftarrow \mathsf{Vfy}(R, \sigma, a_1, \ldots, a_\ell, \pi)$**:** Parse $\pi = (A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$. Accept the proof if and only if

$$e(A, B) = e(G^\alpha, H^\beta) e(G^{\frac{\sum_{i=0}^{\ell} a_i (\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\gamma}}, H^\gamma) e(C, H^\delta).$$

$\pi \leftarrow \mathsf{Sim}(R, \tau, a_1, \ldots, a_\ell)$**:** Pick $r, s \leftarrow \mathbb{Z}_p$ and compute a simulated proof $\pi = (A, B, C)$ as

$$A = G^r \qquad B = H^s \qquad C = G^{\frac{rs - \alpha\beta - \sum_{i=0}^{\ell} a_i (\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\delta}}.$$

Theorem 3. *The protocol given above is a non-interactive zero-knowledge argument with perfect completeness and perfect zero-knowledge. It has statistical knowledge soundness against adversaries that only use a polynomial number of generic bilinear group operations.*

Proof. Perfect completeness follows by direct verification. Perfect zero-knowledge follows from the fact that both in real proofs and simulated proofs A, B are uniformly random group elements and through the verification equation uniquely determine C.

To see that we have statistical knowledge soundness against generic adversaries [Sho97,BBS04] first note that any test the adversary can do on the common reference string corresponds to an equality test of Laurent polynomials. Either the polynomials match formally, or by the Schwartz-Zippel lemma there is negligible probability of them matching up over the random choices of $\alpha, \beta, \gamma, \delta, x$. The adversary therefore has negligible probability of learning anything it did not already know about the common reference string using only generic group operations. What remains is the possibility that the adversary computes A, B and C as exponentiations of group elements to known field elements. This corresponds exactly to an affine prover strategy on the LIP "in the exponents" and by the knowledge soundness of the LIP we can extract a witness from these known field elements. □

Efficiency. The proof size is 2 elements in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$. The common reference string contains a description of the relation R, n elements in $\mathbb{Z}_p$, $m + 2n + 3$ elements in $\mathbb{G}_1$, and $n + 3$ elements in $\mathbb{G}_2$.

The verifier does not need to know the entire common reference string, it suffices to know

$$\sigma_V = \left(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, H^\gamma, H^\delta, \left\{ G^{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}} \right\}_{i=0}^{\ell}, e(G^\alpha, H^\beta) \right).$$

The verifier's reference string only contains a description of the bilinear group, $\ell + 1$ elements in $\mathbb{G}_1$, 2 elements in $\mathbb{G}_2$, and 1 element in $\mathbb{G}_T$.

The verification consists of checking that the proof consists of three appropriate group elements and checking a single pairing product equation. The verifier computes ℓ exponentiations in $\mathbb{G}_1$, a small number of group multiplications, and 3 pairings (assuming $e(G^\alpha, H^\beta)$ is precomputed in the verifier's reference string).

The prover has to compute the polynomial $h(X)$. The prover can compute the polynomial evaluations

$$\sum_{i=0}^{m} a_i u_i(r_q) = \sum_{i=0}^{m} a_i u_{i,q} \qquad \sum_{i=0}^{m} a_i v_i(r_q) = \sum_{i=0}^{m} a_i v_{i,q} \qquad \sum_{i=0}^{m} a_i w_i(r_q) = \sum_{i=0}^{m} a_i w_{i,q}$$

for $q = 1, \ldots, n$. It depends on the relation how long time this computation takes; if it arises from an arithmetic circuit where each multiplication gate connects to a constant number of wires, the relation will be sparse and the computation will be linear in n. Since the polynomials have degree $n-1$ they are completely determined by these evaluation points. If $r_1, \ldots, r_n$ are roots of unity for a suitable prime p she can compute $h(X)$ using standard Fast Fourier Transform techniques in $O(n \log n)$ operations in $\mathbb{Z}_p$. The prover can also compute the coefficients of $\sum_{i=0}^{m} a_i u_i(X)$ and $\sum_{i=0}^{m} a_i v_i(X)$ using FFT techniques. Having all the coefficients, the prover does $m + 3n - \ell + 3$ exponentiations in $\mathbb{G}_1$ and $n+1$ exponentiations in $\mathbb{G}_2$.

Asymptotically the exponentiations are the dominant cost as the security parameter grows. However, in practice the multiplications that go into the FFT computations may be more costly for moderate security parameters and large statements. In that case, it may be worth to use a larger common reference string that contains precomputed $G^{u_i(x)}, G^{v_i(x)}, H^{v_i(x)}$ elements for $i = 0, \ldots, m$ such that A and B can be constructed directly instead of the prover having to compute the coefficients of $\sum_{i=0}^{m} a_i u_i(X)$ and $\sum_{i=0}^{m} a_i v_i(X)$ and then do the exponentiations. In the case of boolean circuits we have $a_i \in \{0, 1\}$ and the prover can with such precomputed elements just do m group multiplications for each when computing A and B. We have for this reason let the CRS be longer in Table 1 to get a low computational cost for the prover.

4 Lower bounds for non-interactive arguments

It is an intriguing question how efficient non-interactive arguments can be. We will now give a lower bound showing that pairing-based non-interactive arguments must have proofs with at least 2 group elements if one-way functions exist. More precisely, we look at pairing-based arguments where the common reference string contains a description of a bilinear group and a number of group elements, the proof consists of a number of group elements computed by the prover using generic group operations, and the verifier checks the proof using generic bilinear group operations. We will show that for such pairing-based argument systems, the proof needs to have elements from both $\mathbb{G}_1$ and $\mathbb{G}_2$ if the language includes hard decisional problems as defined below.

Let us consider sampleable decisional problems for a relation R, where there are two sampling algorithms Yes and No. Yes samples statements and witnesses in the relation. No samples statements outside the language L_R defined by the relation. We are interested in relations where it is hard to tell whether a statement ϕ has been sampled by Yes or No.

Definition 5. *We say the relation generator $\mathcal{R}$ has hard decisional problems if there are two efficient algorithms Yes and No such that for $(R, z) \leftarrow \mathcal{R}(1^\lambda)$ we have $\mathsf{Yes}(R) \to (\phi, w) \in R$ and $\mathsf{No}(R) \to \phi \notin L_R$ with overwhelming probability, and for all non-uniform*

polynomial time distinguishers $\mathcal{A}$

$$\Pr\left[(R,z) \leftarrow \mathcal{R}(1^\lambda); \phi_0 \leftarrow \mathsf{No}(R); (\phi_1, w_1) \leftarrow \mathsf{Yes}(R); b \leftarrow \{0,1\} : \mathcal{A}(R,z,\phi_b) = b\right] \approx \frac{1}{2}.$$

If one-way functions exist, we can construct pseudorandom generators. A pseudorandom generator can be used to generate a pseudorandom string, a Yes-instance, with the seed being the witness. To get a No-instance we sample a uniform random string, which with overwhelming probability is not pseudorandom. If the relation R is NP-complete, or just expressive enough to capture pseudorandom generators, then it has a hard decisional problem.

4.1 Linear interactive proofs cannot have linear decision procedures

We will now prove that LIPs cannot have a linear decision procedure. This answers an open question raised by Bitansky et al. [BCI+13]. The result holds even if we consider designated verifier LIPs and instead of knowledge soundness only consider the weaker notion of soundness that we now define.

Definition 6 (Statistical soundness against affine prover strategies). *We say a LIP is (adaptively) sound against affine prover strategies if for all adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{c}(R,z) \leftarrow \mathcal{R}(1^\lambda); (\boldsymbol{\sigma}_P, \boldsymbol{\sigma}_V, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R); (\phi, \Pi) \leftarrow \mathcal{A}(R,z) \\ \boldsymbol{\pi} = \Pi \boldsymbol{\sigma}_P; \boldsymbol{t} \leftarrow \mathsf{Test}(R,\phi) : \phi \notin L_R \ \wedge \ \boldsymbol{t}(\boldsymbol{\sigma}_V, \boldsymbol{\pi}) = \mathbf{0}\end{array}\right] \approx 0.$$

Theorem 4. *There are no 2-move algebraic linear interactive proofs with a linear decision procedure for relation generators with hard decisional problems.*

Proof. When the decision procedure is linear, the test $\boldsymbol{t}(\boldsymbol{\sigma}_V, \boldsymbol{\pi}) = \mathbf{0}$ can be rewritten as $T\Pi\boldsymbol{\sigma}_P = T'\boldsymbol{\sigma}_V$, where the matrices $T \in \mathbb{F}^{\eta \times k}$ and $T' \in \mathbb{F}^{\eta \times m_V}$ can be efficiently computed from $\boldsymbol{t}$.

Let us now construct an adversary $\mathcal{A}$ that given R and ϕ has a good chance of determining whether ϕ is sampled as a Yes-instance or a No-instance. First, $\mathcal{A}$ repeatedly runs $(\phi_i, w_i) \leftarrow \mathsf{Yes}(R)$ and computes the matching proof and test matrices Π_i and (T_i, T_i'). Let V be the vector space generated by the tuples $(T_i\Pi_i, T_i')$. The adversary keeps sampling tuples until it hits a sequence of λ tuples in a row that already belong to V. With overwhelming probability the vector space V is such that there more than 50% chance for a new Yes-instance yielding a tuple already in V. The adversary runs in polynomial time since any new linearly independent tuple increases the dimension of V and the maximal possible dimension is $\eta(m_P + m_V)$, in which case V would include all tuples.

Now the adversary looks at the statement ϕ that it is trying to classify as a Yes-instance or a No-instance. It computes the test matrices T and T' for ϕ and then tries to solve $(T\Pi, T') = \sum_i r_i(T_i\Pi_i, T_i')$ for $\Pi \in \mathbb{F}^{k \times m_P}$ and $r_i \in \mathbb{F}$. This is a system of linear equations and can therefore be solved efficiently. If a solution is found it guesses $\phi \in L_R$ and if no solution is found it guesses $\phi \notin L_R$.

Let us first analyze the case where $\phi \in L_R$. Since this is a Yes-instance there is more than 50% chance that there is a solution Π such that $(T\Pi, T')$ belongs to the vector space V, so the adversary has at least 50% chance of guessing $\phi \in L_R$.

Next, let us analyze the case where $\phi \notin L_R$. If we run the setup algorithm $(\boldsymbol{\sigma}_P, \boldsymbol{\sigma}_V, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R)$ and $\phi \notin L_R$ we have negligible probability for $T\Pi\boldsymbol{\sigma}_P = T'\boldsymbol{\sigma}_V$. However, by completeness we have for all tuples in V that $T_i\Pi_i\boldsymbol{\sigma}_P = T'_i\boldsymbol{\sigma}_V$. If there were a matrix Π such that $(T\Pi, T') = \sum_i r_i(T_i\Pi, T'_i)$ we would have $T\Pi\boldsymbol{\sigma}_P = \sum_i r_i T_i\Pi_i\boldsymbol{\sigma}_P = \sum_i r_i T'_i\boldsymbol{\sigma}_V = T'\boldsymbol{\sigma}_V$, so soundness implies this probability is negligible. The adversary guesses $\phi \notin L_R$ with overwhelming probability. □

4.2 Lower bound for the size of generic pairing-based non-interactive arguments

We will now show that a generic pairing-based non-interactive argument over Type III groups must have elements in both $\mathbb{G}_1$ and $\mathbb{G}_2$. The intuition behind this argument is that if we have a unilateral argument with only elements in $\mathbb{G}_1$ or only elements in $\mathbb{G}_2$, then the verification equations become linear and the impossibility result for LIPs apply.

Before we get started with the proof, let us define some useful notation. Define for a vector $\boldsymbol{v} = (v_1, \ldots, v_n)$ that $G^{\boldsymbol{v}} = (G^{v_1}, \ldots, G^{v_n})$. Define for a vector of group elements $G^{\boldsymbol{v}}$ and a matrix A that $(G^{\boldsymbol{v}})^A = G^{\boldsymbol{v}A}$. Also, define for two vectors of group elements $e(G^{\boldsymbol{v}}, H^{\boldsymbol{w}}) = \prod_{i=1}^n e(G^{v_i}, H^{w_i})$.

We will consider pairing-based argument systems $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vfy})$ where the proofs consist of group elements and where the algorithms only use generic group operations. Let us be explicit about how such a system operates and the consequences of using generic group operations.

$(\sigma, \tau) \leftarrow \mathsf{Setup}(R)$**:** The relation contains a description of a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ and the common reference string contains group elements in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. Let us fix generators G and H for $\mathbb{G}_1$ and $\mathbb{G}_2$ and write the vectors of group elements in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ as $\boldsymbol{\Sigma}_1 = G^{\boldsymbol{\sigma}_1}$, $\boldsymbol{\Sigma}_2 = H^{\boldsymbol{\sigma}_2}$ and $\boldsymbol{\Sigma}_T = e(G, H)^{\boldsymbol{\sigma}_T}$.
We want to avoid that the prover can learn non-trivial information about the discrete logarithms $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \boldsymbol{\sigma}_T$ 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. We say the common reference string is *disclosure-free* if for any pairing product equation on the group elements in $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ and $\boldsymbol{\Sigma}_T$ it is possible with overwhelming probability to predict whether the equation holds or not, when we know the distribution of the common reference string but where we do not know the actual group elements.

$\pi \leftarrow \mathsf{Prove}(R, \sigma, \phi, w)$**:** A prover using generic group operations and working on a disclosure-free common reference string has negligible chance of learning any non-trivial information about the common reference string group elements. This means her only viable mode of operation is to pick matrices Π_1, Π_2 and Π_T and compute the proof by setting $\pi = (\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \boldsymbol{\psi}_T)$, where

$$\boldsymbol{\psi}_1 = \boldsymbol{\Sigma}_1^{\Pi_1} \qquad \boldsymbol{\psi}_2 = \boldsymbol{\Sigma}_2^{\Pi_2} \qquad \boldsymbol{\psi}_T = \boldsymbol{\Sigma}_T^{\Pi_T}.$$

$0/1 \leftarrow \mathsf{Vfy}(R, \sigma, \phi, \pi)$**:** A verifier using generic group operations can only verify a proof by mapping ϕ to matrices and vectors $\{A_q, B_q, C_q, D_q, \boldsymbol{e}_q, \boldsymbol{f}_q\}_{q=1}^Q$ of elements in $\mathbb{Z}_p$

and checking pairing product equations of the form

$$e(\boldsymbol{\Sigma}_1^{A_q}, \boldsymbol{\Sigma}_2)e(\boldsymbol{\psi}_1^{B_q}, \boldsymbol{\Sigma}_2)e(\boldsymbol{\Sigma}_1^{C_q}, \boldsymbol{\psi}_2)e(\boldsymbol{\psi}_1^{D_q}, \boldsymbol{\psi}_2) = \boldsymbol{\Sigma}_T^{\boldsymbol{e}_q} \cdot \boldsymbol{\psi_T}^{\boldsymbol{f}_q}.$$

We note that there is no loss of generality in excluding multi-exponentiation equations in $\mathbb{G}_1$ or $\mathbb{G}_2$; such equations can be translated to pairing product equations by pairing them with G or H.

We now get the following corollary to Theorem 4.

Corollary 1. *A pairing-based non-interactive argument with a disclosure-free common reference string and algorithms using generic group operations cannot exist for relation generators with hard decisional problems unless the proofs have elements both in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*.*

Proof. When the common reference string is disclosure free and the algorithms use generic operations they must work as outlined above. Taking discrete logarithms we get verification equations of the form

$$\boldsymbol{\sigma}_1 A_q \boldsymbol{\sigma}_2 + \boldsymbol{\pi}_1 B_q \boldsymbol{\sigma}_2 + \boldsymbol{\sigma}_1 C_q \boldsymbol{\pi}_2 + \boldsymbol{\pi}_1 D_q \boldsymbol{\pi}_2 = \boldsymbol{\sigma}_T \boldsymbol{e}_q + \boldsymbol{\pi_T} \boldsymbol{f}_q,$$

where $\boldsymbol{\psi}_1 = G^{\boldsymbol{\pi}_1}$ and $\boldsymbol{\psi}_2 = H^{\boldsymbol{\pi_2}}$ and $\boldsymbol{\psi}_T = e(G, H)^{\boldsymbol{\pi}_T}$. If either $\boldsymbol{\pi}_1$ or $\boldsymbol{\pi}_2$ are empty, there are no $\boldsymbol{\pi}_1 D_q \boldsymbol{\pi}_2$ parts in the verification equations. Observe also that without loss of generality we can assume all the entries in the outer product of $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ are given in $\boldsymbol{\sigma}_T$ (this does not affect disclosure-freeness) so we can set $A_q = 0$ in every equation. This means all the verification equations are linear. Since the verification equations correspond to verifying a LIP "in the exponents" it follows from the impossibility of having LIPs with a linear decision procedure that the proof must have that both $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ are non-trivial and therefore that the proof has elements both in $\mathbb{G}_1$ and $\mathbb{G}_2$. □

Acknowledgments

We thank Eran Tromer for interesting discussions about the performance of SNARK implementations and the anonymous reviewers for their helpful reviews.

References

[AF07] Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 118–136, 2007.

[AGOT14] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Unified, minimal and selectively randomizable structure-preserving signatures. In *TCC*, volume 8349 of *Lecture Notes in Computer Science*, pages 688–712, 2014.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, 2004.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Innovations in Theoretical Computer Science*, pages 326–349, 2012.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *STOC*, pages 111–120, 2013.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for C: verifying program executions succinctly and in zero knowledge. In *CRYPTO*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108, 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 *IEEE Symposium on Security and Privacy*, pages 459–474, 2014.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, volume 7785 of *Lecture Notes in Computer Science*, pages 315–333, 2013.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In *STOC*, pages 505–514, 2014.

[BCTV14a] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In *CRYPTO*, volume 8617 of *Lecture Notes in Computer Science*, pages 276–294, 2014.

[BCTV14b] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX*, pages 781–796, 2014.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *STOC*, pages 103–112, 1988.

[BP15] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. In *ASIACRYPT*, volume 9453 of *Lecture Notes in Computer Science*, pages 236–261, 2015.

[BSCG+14] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Shaul Kfir, Eran Tromer, and Madars Virza. libsnark, 2014. Available at https://github.com/scipr-lab/libsnark.

[CFH+15] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 253–270, 2015.

[CTV15] Alessandro Chiesa, Eran Tromer, and Madars Virza. Cluster computing in zero knowledge. In *EUROCRYPT*, volume 9057 of *Lecture Notes in Computer Science*, pages 371–403, 2015.

[CV16] Alessandro Chiesa and Madars Virza. Personal communication, 2016.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In *ASIACRYPT*, volume 8873 of *Lecture Notes in Computer Science*, pages 532–550, 2014.

[DFKP13] George Danezis, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio coin: building zerocoin from a succinct pairing-based proof system. In *PETShopCCS*, 2013.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645, 2013.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proofs. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111, 2006.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *Journal of the ACM*, 59(3):11:1–11:35, 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 *ASIACRYPT*, volume 4248 of *Lecture Notes in Computer Science*, pages 444–459, 2006.

[Gro09] Jens Groth. Linear algebra with sub-linear zero-knowledge arguments. In *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 192–208, 2009.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340, 2010.

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

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *STOC*, pages 723–732, 1992.

[Kil95] Joe Kilian. Improved efficient arguments (preliminary version). In *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 311–324, 1995.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 169–189, 2012.

[Lip13] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In *ASIACRYPT*, volume 8269 of *Lecture Notes in Computer Science*, pages 41–60, 2013.

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

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

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, 1997.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18, 2008.

[Wal15] Michael Walfish. A wishlist for verifiable computation: An applied cs perspective, 2015.