Making Groth's zk-SNARK Simulation Extractable in the Random Oracle Model

Sean Bowe
sean@z.cash

Ariel Gabizon
ariel@z.cash

February 15, 2018

1 Introduction

The purpose of this note is to provide a variant of Groth's zk-SNARK [5] that satisfies simulation extractability, which is a strong form of adaptive non-malleability. Let us call such a construction a zk-SE-SNARK for brevity. A straightforward alteration of the construction gives a succinct *Signature of Knowledge* (SoK). Our construction of both primitives uses a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and a proof/signature requires three $\mathbb{G}_1$ elements and two $\mathbb{G}_2$ elements.

Groth and Maller [6] recently gave a construction of zk-SE-SNARKs and SoKs. Their zk-SE-SNARK has the advantage of requiring only 2 $\mathbb{G}_1$ elements and 1 $\mathbb{G}_2$ element as in [5]. Their SoK requires an additonal string to be output. Furthermore, they rely on concrete assumptions holding in the Generic Group Model, together with a collision-resistant hash function only for the SoK; whereas our analysis for both primitives requires the full generic group model as in [5] together with the random oracle model.[1]

On the other hand, our work has the practical advantage of the prover/signer requiring *only one group operation more* than the prover of [5]; whereas [6], as a result of relying on Square Arithmetic Programs [3] rather than Quadratic Arithmetic Programs [4], require twice as much $\mathbb{G}_2$ operations as [5]. As

[1] As discussed with Jens Groth and Mary Maller, it is possible to phrase a concrete assumption holding in the Generic Group Model under which our construction and [5] are secure; however this assumption would be quite strong and have an ad-hoc flavor, and in particular would still be stronger than the assumptions in [6] with one exception: [6] require an assumption following from an "asymmetric" group model where there is no efficient isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$. Our work, as [5], does not require assuming this, and the analysis works in particular when $\mathbb{G}_1 = \mathbb{G}_2$.

$\mathbb{G}_2$ operations are typically much more expensive than $\mathbb{G}_1$ operations, this significantly increases the total running time of the prover [1].

2 Definitions

For a relation R we denote

$$L_{\mathsf{R}} := \{\mathbf{x} | \ \exists \omega \ s.t. \ (\mathbf{x}, \omega) \in \mathsf{R}\} .$$

Random oracles We assume all parties have access to a *random oracle* mapping arbitrary strings to uniform elements of a certain domain $\mathcal{D}$. When discussing NILPs it will be convenient to assume $\mathcal{D} = \mathbb{F}_p^*$ and when discussing SNARKs we'll assume $\mathcal{D} = \mathbb{G}_1^*$. To clarify we refer to parties as $\mathbb{F}_p^*$-oracle machines in the first case, and $\mathbb{G}_1^*$-oracle machines in the second.

For a string s we'll denote by y_{s} the output of the random oracle on s. And we'll denote $Y := \{y_{\mathsf{s}}\}$ the set of all such outputs.

We will at times below discuss circuits/algorithms running in time $\text{poly}(\lambda)$ doing linear operations on the (infinite) vector Y. What we mean by this is that when the party chooses the matrix Π describing the linear operation, he also chooses a $\text{poly}(\lambda)$-length sequence X of strings s all of length $\text{poly}(\lambda)$. And in fact, only applies Π on the vector $(Y_{\mathsf{s}})_{\mathsf{s} \in X}$.

Asymptotics Implicitly, all algorithms/circuits and parameters described below depend on an integer security parameter λ. For example, when we discuss a relation R we mean an infinite sequence of relations indexed by λ. When we refer in Section 2.1 to a prime field $\mathbb{F}_p$, we also refer to an infinite sequence of prime fields indexed by λ.

Notation For a domain $\mathcal{D}$ we denote by $\boldsymbol{(\mathcal{D})}$ the set of vectors over $\mathcal{D}$.

2.1 NILPs

What we define here as a NILP is what [5] in fact calls a split-NILP, with the addition of participants having access to a random oracle over $\mathbb{F}_p^*$.

Definition 2.1 (NILPs with a random oracle)**.** *A Non-Interactive Linear Proof system $\mathcal{N}$ in the Random-Oracle model over prime field $\mathbb{F}_p$ (RO-NILP) for R consists of four (possibly randomized) algorithms $(\mathsf{Gen}, \mathrm{P}, \mathrm{V}, \mathrm{P}^{\mathrm{sim}})$ that are $\mathbb{F}_p^*$-oracle machines running in time $\text{poly}(\lambda)$.*

1. Gen *outputting a trapdoor* τ *and common reference string* $\sigma = (\sigma_1, \sigma_2) \in (\mathbb{F}_p)$.

2. P *that takes as input* σ *and* $(\mathbf{x}, \omega) \in \mathsf{R}$. P *first computes* Π_1, Π_2 *where* Π_i *is a matrix over* $\mathbb{F}_p$ as a function of $\mathbf{x}, \omega$ only. *Then* P *outputs* $\pi = (\pi_1, \pi_2) = (\Pi_1 \cdot (\sigma_1, Y), \Pi_2 \cdot \sigma_2)$. *(See explanation about* Y *in Section 2, "Random Oracles".)*

3. $\mathrm{V}(\sigma, \mathbf{x}, \pi)$ *computes matrices* $T_1, \ldots, T_d$ depending only on $\mathbf{x}$. *It then outputs* acc *iff for each* $i \in [d]$

$$(\sigma_1, Y, \pi_1) \cdot T_i(\sigma_2, \pi_2) = 0.$$

4. $\mathrm{P}^{\mathrm{sim}}$ *taking as input* $\mathbf{x}$, τ *and outputting* π.

We assume for any $\mathbf{x} \in L_{\mathsf{R}}$ *the first coordinate of* $\mathbf{x}$ *is one (to enable* P *to always take affine functions of* σ*).*

We say $\mathscr{N}$ *as above is a* Simulation-Extractable Non-Interactive Linear Proof system in the Random Oracle model *(RO-SE-NILP) over* $\mathbb{F}_p$ *for* R *if*

1. ***Completeness****: for any* $(\mathbf{x}, \omega) \in \mathsf{R}$*, if* $\pi = \mathrm{P}(\mathbf{x}, \omega)$ *then* $\mathrm{V}(\pi) = \mathsf{acc}$ *with probability one.*

2. ***Zero-Knowledge:*** *For any output* (τ, σ) *of* Gen *and* $(\mathbf{x}, \omega) \in \mathsf{R}$*, the distribution* $\mathrm{P}^{\mathrm{sim}}(\tau, \mathbf{x})$ *is identical to* $\mathrm{P}(\sigma, \mathbf{x}, \omega)$.

3. ***Simulation-Extractability:*** *For any efficient* $\mathcal{A}$ *there exists an efficient* χ *such that the following holds: Fix any output* (τ, σ) *of* Gen*. Suppose that* $\mathcal{A}$ *make a non-adaptive sequence of queries* $Q = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$ *to* $\mathrm{P}^{\mathrm{sim}}(\tau, \cdot)$*. that returns answers* $A = \{(\mathbf{x}_1, \pi_1), \ldots, (\mathbf{x}_\ell, \pi_\ell)\}$*. Finally* $\mathcal{A}$ *computes matrices* T_1, T_2 *depending only on* $\mathbf{x}$ *and outputs* $\pi = (\pi_1, \pi_2)$ *with* $\pi_i := (A, \sigma_i) \cdot T_i$*.* χ *given* $\mathbf{x}, T_1, T_2$ *outputs* ω*. The probability that*

   - $\mathcal{A}$ *"wins":* $(\mathbf{x}, \pi) \notin A$ *and also* $\mathrm{V}(\mathbf{x}, \pi) = \mathsf{acc}$*, while*
   - χ *"loses":* $(\mathbf{x}, \omega) \notin \mathsf{R}$

   is $\mathrm{negl}(\lambda)$.

We say a NILP over $\mathbb{F}_p$ has degree d *if* Gen *and* $\mathrm{P}^{\mathrm{sim}}$ *consist of sampling a random vector* $\mathbf{z}$ *over* $\mathbb{F}_p$ *and outputting* $\{P_i(\mathbf{z})\}_{i \in [s]}$ *where* P_i *is a polynomial over* $\mathbb{F}_p$ *of degree at most* d.

The set of polynomial $\{P_i\}_{i \in [s]}$ *possibly depends on* $\mathbf{x}$ *in the case of* $\mathrm{P}^{\mathrm{sim}}$*, but must not depend on the value of* τ *in both cases.*

2.2 Adaptive NILP adversaries

In the regular soundness and simulation extractability definitions of a NILP, the adversary must choose his matrices and the resultant output as a function only of the public input $\mathbf{x}$. We give a definition of a more adaptive adversary that may check if a certain bilinear relation holds amongst the CRS elements, and take into account the result of these checks when constructing his proof. This exactly captures the power of an adversary in the generic group model when we compile the NILP into a SNARK as in [5] using a bilinear group. We will encorporate into our definition the interaction of the adversary with a party like $\mathrm{P}^{\mathrm{sim}}$ and his access to a random oracle with $\mathbb{F}_p^*$ output.

Definition 2.2 (Adaptive bilinear adversary)**.** *An* adaptive bilinear adversary $\mathcal{A}$ over $\mathbb{F}_p$ *is a* $\mathbb{F}_p^*$*-oracle machine operating as follows. It begins with an explicit input* $\mathbf{x}$*, and auxiliary inputs* $\sigma_1, \sigma_2 \in (\mathbb{F}_p)$*. It initializes an empty vector* U *that will hold boolean values. At each step* $\mathcal{A}$ *does the following.*

1. *Depending only on* $\mathbf{x}$ *and the value of the vector* U*, it chooses matrices* Π_1, Π_2, T *over* $\mathbb{F}_p$ *and possibly also a message* m*.*
2. *If* $\mathcal{A}$ *is in interaction with a party* P *it may send* P *the message* m*, and if* P *replies with a vector* v_1 *of* $\mathbb{G}_1$ *elements and* v_2 *of* $\mathbb{G}_2$ *elements,* $\mathcal{A}$ *appends* v_1 *to* σ_1 *and* v_2 *to* σ_2*.*
3. *It then checks if* $(\sigma_1, Y) \cdot (T \cdot \sigma_2) = 0$ *and adds the value* 0 *to* U *if so, and adds the value* 1 *to* U *otherwise.*

After each step $\mathcal{A}$ *decides whether to continue or terminate, in which case it outputs matrices* Π_1, Π_2 *and the values* $(\sigma_1, Y) \cdot \Pi_1, \sigma_2 \cdot \Pi_2$*. All decisions (on whether to terminate and what values to output), depend only on* $\mathbf{x}$ *and the values in* U *at that point.*

We say that an RO-SE-NILP $\mathscr{N}$ is an *Adaptively-Bilinear Simulation-Extractable NILP* (AB-SE-NILP) if the simulation extractability property holds also with respect to adpatively bilinear adversaries making at most $\mathrm{poly}(\lambda)$ steps, adaptively making queries $\{\mathbf{x}_i\}$ to $\mathrm{P}^{\mathrm{sim}}(\tau, \cdot)$.

Theorem 2.3. *If* $\mathscr{N}$ *is a degree* d *RO-SE-NILP over* $\mathbb{F}_p$ *for a relation* R*, where* $d/|\mathbb{F}_p| = \mathrm{negl}(\lambda)$*, then it is also an AB-SE-NILP over* $\mathbb{F}_p$ *for* R*.*

Proof. The proof is based on Lemma 1 and Theorem 2 of [5]. Let $\mathcal{A}$ be an adaptive bilinear adversary. We assume for simplicity $\mathcal{A}$ is deterministic (if there exists a randomized circuit $\mathcal{A}$ breaking simulation extractability there

exists a fixing of its randomness where it breaks simulation extractability). We construct a non-adaptive adversary $\mathcal{A}$' such that for any $\mathbf{x}$, the probability over the randomness of Gen when outputting σ and the randomness of $\mathrm{P}^{\mathrm{sim}}$ in its replies that $\mathcal{A}(\mathbf{x}, \sigma) \neq \mathcal{A}'(\mathbf{x}, \sigma)$ is $\mathrm{negl}(\lambda)$. This means that the extractor χ for $\mathcal{A}$' guaranteed to exist by the properties of an RO-SE-NILP is also good for $\mathcal{A}$. $\mathcal{A}$' works as follows. He begins running $\mathcal{A}$ on $(\sigma, \mathbf{x})$. In the first step he is able to choose the matrices Π_1, Π_2, T and message m to $\mathrm{P}^{\mathrm{sim}}$ (consisting of the desired inputs $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ on which $\mathcal{A}$ wishes to see simulated proofs), as $\mathcal{A}$ would have chosen them, as in the first step these depend only on $\mathbf{x}$ and the empty vector U.

The main question is what to do when $\mathcal{A}$ wishes to do the bilinear check:

$$(\sigma_1, Y) \cdot (T \cdot \sigma_2) \stackrel{?}{=} 0.$$

and insert its result into U.

If we denote by $\mathbf{z}$ the variables used by Gen and $\mathrm{P}^{\mathrm{sim}}$, the main point is that this equation corresponds to an equation of degree at most $2d$ between polynomials in $\mathbf{z}, Y$. It follows from the Schwartz-Zippel Lemma that if the equation is not a polynomial identity, equality will hold with probability at most $2d/|\mathbb{F}_p| = \mathrm{negl}(\lambda)$. Motivated by this, $\mathcal{A}$' returns 0 to $\mathcal{A}$ if the equation is a polynomial identity and 1 otherwise. He proceeds to run $\mathcal{A}$ until the next bilinear check, where again the check will be a polynomial equation (*fully determined by the matrices chosen by $\mathcal{A}$ up to this point*), and he responds to $\mathcal{A}$ using the same strategy.

Finally, $\mathcal{A}$' outputs ($\mathcal{A}$)'s output in the end of this process. The probability that their outputs differ is at most the probability that one of ($\mathcal{A}$')'s responses to the bilinear checks was different from the correct response given the values of σ and $\mathrm{P}^{\mathrm{sim}}$'s replies. This probability is union bounded by $\mathrm{poly}(\lambda) \cdot \mathrm{negl}(\lambda) = \mathrm{negl}(\lambda)$.

□

3 The construction

We describe our NILP construction. It is based on the variant of [5] described in [2] where the CRS is slightly extended (this extension was important to [2] for their multi-party computation protocol, and is not crucial here). We use the same notation regarding QAPs as in [2, 5].

Let $\mathscr{Q} = \left\{ \{u_i\}_{i \in [0..m]}, \{v_i\}_{i \in [0..m]}, \{w_i\}_{i \in [0..m]}, t \right\}$ be a QAP over $\mathbb{F}_p$ of degree n and size m. Let $0 < \ell < m$ be an integer. We define the relation R

to consist of all pairs $(\mathbf{x} = (a_0 = 1, a_1, \ldots, a_\ell), \omega = (a_{\ell+1}, \ldots, a_m))$ such that $(a_0, \ldots, a_m)$ satisfies $\mathscr{Q}$ (see [5] for a definition of QAPs in this notation).

We use the shorthand $\mathsf{ic} := \sum_{i=0}^{\ell} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))$ below. This is the element relating to the primary QAP input $\mathbf{x} = (a_0 = 1, a_1, \ldots, a_\ell)$.

We present a NILP for the relation R.

The idea is to have the prover randomize the δ element of the CRS with a secret scalar, and require her to prove knowledge of this secret scalar. This creates a situation where the adversary $\mathcal{A}$ must use his own different randomization of δ, making it hard for him to use elements from $\mathrm{P}^{\mathrm{sim}}$'s simulated proofs.

Generator Gen**:** Choose uniform elements $\alpha, \beta, \delta, x \in \mathbb{F}_p^*$.

Output:

$$\sigma_1 := \left\{ \alpha, \beta, \delta, \left\{x^i\right\}_{i\in[0..2n-2]}, \left\{\alpha x^i\right\}_{i\in[1..n-1]}, \left\{\beta x^i\right\}_{i\in[1..n-1]}, \right.$$

$$\left. \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}\right\}_{i\in[\ell+1..m]}, \left\{\frac{x^i \cdot t(x)}{\delta}\right\}_{i\in[0..n-2]} \right\}$$

$$\sigma_2 := \left\{\beta, \delta, \left\{x^i\right\}_{i\in[0..n-1]}\right\}$$

Prover P**:** Fix public input $a_1, \ldots, a_\ell$. P, given witness $\omega = (a_{\ell+1}, \ldots, a_m)$ does the following.

1. She chooses a random element $d \in \mathbb{F}_p^*$, and computes $\delta' := d \cdot \delta$.
2. She chooses random $r, s \in \mathbb{F}_p$.
3. She computes
$$A := \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta', B := \beta + \sum_{i=0}^{m} b_i v_i(x) + s\delta'$$
4. She computes
$$C := \frac{\sum_{i=\ell+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + h(x)t(x)}{\delta'} + As + Br - rs\delta'.$$
5. Define $\mathsf{s} := (A, B, C, \delta')$, and $z := y_{\mathsf{s}} \cdot \delta$.
6. She outputs $\pi_1 = (A, C, z), \pi_2 = (B, \delta')$.

Verifier V**:** Given A, B, C, δ', z, check that:

1. $A \cdot B = \alpha \cdot \beta + \mathsf{ic} + C \cdot \delta$.
2. $y_{\mathsf{s}} \cdot \delta' = z \cdot \delta$; for $\mathsf{s} := (A, B, C, \delta')$.

Simulator $\mathrm{P}^{\mathrm{sim}}$**:** Given α, β, δ, x,

1. Choose random $\delta' \in \mathbb{F}_p^*$.
2. Choose random $A, B \in \mathbb{F}_p$ and let
$$C := \frac{A \cdot B - \mathsf{ic} - \alpha\beta}{\delta'}$$
3. Let $\mathsf{s} := (A, B, C, \delta')$, and $z := y_{\mathsf{s}} \cdot (\delta'/\delta)$.
4. Output (A, B, C, δ', z).

4 Security proof

Theorem 4.1. *The above construction is an RO-SE-NILP for* R *over* $\mathbb{F}_p$*.*

An immediate corollary from Theorem 2.3 is

Corollary 4.2. *If we are starting from a QAP of degree* d *over* $\mathbb{F}_p$*, and* $d/|\mathbb{F}_p| = \mathrm{negl}(\lambda)$*; then the above construction is an AB-SE-NILP.*

Proof. (of Theorem 4.1) Completeness and Zero-Knowledge are straightforward and almost identical to [5]. We concentrate on simulation extractability. Suppose $\mathcal{A}$ has made a sequence of queries $\mathbf{x}_1, \ldots, \mathbf{x}_v$ to $\mathrm{P}^{\mathrm{sim}}(\tau, \cdot)$, and received answers $\{\pi_j = (A_j, B_j, C_j, \delta_j, z_j)\}_{j\in[v]}$. Denote $\mathsf{s}_j := (A_j, B_j, C_j, \delta_j)$. Let Q‘ be the union of elements in the CRS (σ_1, σ_2) together with those from the random oracle and $\mathrm{P}^{\mathrm{sim}}$'s replies; so

$$Q' := \left\{ \delta, \left\{x^i\right\}_{i\in[0..2n-2]}, \left\{\alpha x^i, \beta x^i\right\}_{i\in[0..n-1]}, \right.$$
$$\left. \left\{ \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} \right\}_{i\in[\ell+1..m]}, \left\{ \frac{x^i \cdot t(x)}{\delta} \right\}_{i\in[0..n-2]} \right\} \cup$$
$$\cup \left\{Y_{\mathsf{s}}\right\}_{string\ \mathsf{s}} \cup \left\{ A_j, B_j, C_j = \frac{A_j B_j - \alpha\beta - \mathsf{ic}_j}{\delta_j}, \delta_j, \frac{\delta_j Y_{\mathsf{s}_j}}{\delta} \right\}_{j\in[v]}.$$

We emphasize that we think of $x, \alpha, \beta, \delta, \{A_j, B_j, \delta_j\}, \{Y_{\mathsf{s}}\}$ as formal variables in this proof. Thus, the elements of Q', and all other elements discussed here, belong to the ring of Laurent polynomials in these variables, i.e.

$$K := \mathbb{F}_p\left[x, x^{-1}, \alpha, \alpha^{-1}, \beta, \beta^{-1}, \delta, \delta^{-1}, \left\{A_j, B_j, \delta_j, A_j^{-1}, B_j^{-1}, \delta_j^{-1}\right\}, \left\{Y_{\mathsf{s}}, Y_{\mathsf{s}}^{-1}\right\}\right]$$

Motivated by this, when we use the term *monomial* henceforth, we mean a Laurent monomial, i.e. a ratio of two monomials, in the variables $\{x, \alpha, \beta, \delta, \{A_j, B_j, \delta_j\}, \{Y_{\mathsf{s}}\}\}$ with no common factors between numerator and denumerator, e.g. $\frac{\alpha A_3}{\delta_2}$.

Observe that the elements of Q' are linearly independent over $\mathbb{F}_p$, thus any element in $\mathrm{span}(Q')$ has a unique representation as an $\mathbb{F}_p$-linear combination of elements of Q'.

Suppose $\mathcal{A}$ has produced elements $A, B, C, \delta', z \in \mathrm{span}(Q')$ such that

$$A \cdot B \equiv C \cdot \delta' + \mathsf{ic} + \alpha\beta$$

and, for $\mathsf{s} := (A, B, C, \delta')$,

$$\delta' \cdot Y_{\mathsf{s}} \equiv \delta \cdot z.$$

For $M \in Q'$ and $P \in \{A, B, C, \delta', z\}$, we denote by $P(M)$ the coefficient of M when writing P as a linear combination of elements of Q'.

Denote $V := \left\{\{\delta_j, A_j, B_j\}_{j\in[v]}, \{Y_{\mathsf{s}}\}_{\text{string } \mathsf{s}}\right\}$

These are the new variables not appearing in (σ_1, σ_2) - which is exactly the CRS of [2].

We show that A, B, C do not use elements of Q' involving the variables V and thus $\mathcal{A}$ only uses the elements in the CRS of [2] to generate A, B, C. From this point the proofs in [2,5] will imply that a witness ω with $(\mathbf{x}, \omega) \in \mathsf{R}$ can be extracted from A, B, C except with probability $\mathrm{negl}(\lambda)$. We first introduce some terminology.

For a *monic* monomial M, and $P \in K$ we write $M \in P$ to mean M appears with non-zero coefficient when writing P as a linear combination of monic monomials. For $P \in K, P \neq A, B, C, \delta', z$, we denote by $P(M)$ the coefficient of M when writing P as a linear combination of monic monomials.

For two monomials M, M' we use the notation $M \sim M'$ to mean $M = c \cdot M'$ for some $c \in \mathbb{F}_p^*$.

At times below, it will be convenient to work with the following set Q of monic monomials such that $\mathrm{span}(Q') \subseteq \mathrm{span}(Q)$:

$$Q := \{\delta\} \cup \left\{x^i, \alpha x^i, \beta x^i, x^i/\delta, \alpha x^i/\delta, \beta x^i/\delta\right\}_{i\in[0..2n-2]} \cup \{Y_{\mathsf{s}}\}_{string\ \mathsf{s}} \cup$$

$$\left\{A_j, B_j, \frac{A_j B_j}{\delta_j}, \frac{\alpha\beta}{\delta_j}, \left\{x^i/\delta_j, \alpha x^i/\delta_j, \beta x^i/\delta_j\right\}_{i\in[0..2n-2]}, \delta_j, \frac{\delta_j Y_{\mathsf{s}_j}}{\delta}\right\}_{j\in[v]}.$$

The second equation implies

$$z \equiv \frac{\delta'}{\delta} \cdot Y_{\mathsf{s}}.$$

This means that if $M \in z$, we have $M \in Q$ and $M\delta/Y_{\mathsf{s}} \in Q$, because $z, \delta' \in \mathrm{span}(Q)$. Inspection shows the possibilities for such monic M are Y_{s}, and in the case $\mathsf{s} = \mathsf{s}_j$ for some $j \in [v]$, also $\frac{\delta_j Y_{\mathsf{s}_j}}{\delta}$. We wish to rule out the second: Note that for a verifying proof π, the value of z is determined by A, B, C, δ'. Hence if two verifying proofs agree on the first four elements A, B, C, δ', they are identical. Conversely, if $\pi \neq \pi_j, \forall j \in [v]$ we also have $\mathsf{s} \neq \mathsf{s}_j, \forall j \in [v]$.

So we must have $z \sim Y_{\mathsf{s}}$. This implies $\delta' \sim \delta$.

We introduce some more notation before proceeding. Denote $C^* := C \cdot \delta', C_0 := \mathsf{ic} + \alpha\beta$. Thus, we have

$$AB \equiv C^* + C_0$$

Denote

$$Q_0 := \left\{x^i, \alpha x^i, \beta x^i\right\}_{i\in[0..2n-2]} \cup \{\alpha\beta\}$$

Note that $C_0 \in \mathrm{span}(Q_0)$. We often use the argument below that if $M \in AB$ but $M \notin \mathrm{span}(Q_0)$ we must have $M \in C^*$ and therefore $M/\delta' \in C$; and since we showed $\delta' \sim \delta$ this means that $M/\delta \in C$.

We now show that $\alpha \in A, \beta \in B$ or $\alpha \in B, \beta \in A$:

For this we first claim that $\alpha\beta \in AB$: We have $\alpha\beta \in C_0$. It suffices to show $\alpha\beta \notin C \cdot \delta'$. $\alpha\beta \in C \cdot \delta'$ implies $\alpha\beta/\delta' \in C$; and thus $\alpha\beta/\delta \in C$. But $\alpha\beta/\delta \notin Q$.

We can thus assume $\alpha\beta \in AB$.

Looking at Q' we have

$$AB(\alpha\beta) = A(\alpha)B(\beta) + A(\beta)B(\alpha) + A(C_j)B(\delta_j) + A(\delta_j)B(C_j)$$

Assume for contradiction that the first two terms are zero.

Then we have

$$A(C_j)B(\delta_j) + A(\delta_j)B(C_j) \neq 0$$

We look at two cases

1. $A(A_j)B(B_j) + A(B_j)B(A_j) = 0$. In this case we have $C^*(A_jB_j) = A(C_j)B(\delta_j) + A(\delta_j)B(C_j) = AB(\alpha\beta) \neq 0$. which means either $A_jB_j \in C_0$ which is false, or $A_jB_j/\delta \in C$, but $A_jB_j/\delta \notin \mathrm{span}(Q\text{‘})$.

2. We have $A(A_j)B(B_j)+A(B_j)B(A_j) \neq 0$. We claim that we can't have $A(A_j), B(A_j) \neq 0$: $AB(A_j^2) = C^*(A_j^2) = A(A_j)B(A_j)$. But $A_j^2/\delta \notin Q$, so either $A_j \notin A$ or $A_j \notin B$. Now assume $A_j \in A, A_j \notin B$. Look at two cases:

   (a) $\delta_j \in B$: Then $AB(A_j\delta_j) = C^*(A_j\delta_j) = A(A_j)B(\delta_j) \neq 0$. But $A_j\delta_j/\delta \notin Q$.

   (b) $C_j \in B$: Then $AB(A_j^2B_j/\delta_j) = C^*(A_j^2B_j/\delta_j) = A(A_j)B(C_j) \neq 0$. But $A_j^2B_j/(\delta_j\delta) \notin Q$.

   The case $A_j \notin A, B_j \in A$ is refuted similarly.

Assume w.l.g. from now on that $A(\alpha), B(\beta) \neq 0$ (otherwise flip A and B, this doesn't change the verification equation holding).

We show that $\beta \notin A, \alpha \notin B$:

Assume for contradiction $\beta \in A$. We have

$$AB(\beta^2) = A(\beta)B(\beta) \neq 0$$

Since $\beta^2 \notin C_0$, we have $\beta^2/\delta \in C$. But $\beta^2/\delta \notin Q$ which is a contradiction.

An analogous argument shows $\alpha \notin B$.

Now suppose $C_j \in A$. Then,

$$AB(A_jB_j\beta/\delta_j) = A(C_j)B(\beta) \neq 0.$$

Hence $A_jB_j\beta/(\delta_j\delta) \in C$ - a contradiction as this monomial is not in Q. An analogous argument shows $C_j \notin B$.

Suppose $A_j \in A$.Then,

$$AB(A_j\beta) = A(A_j)B(\beta) \neq 0.$$

Then $A_j\beta/\delta \in C$ a contradiction as this monomial is not in Q. Analgous arguments show $B_j \notin A, A_j \notin B, B_j \notin B$.

Suppose $Y_s \in A$ for some string s. Then

$$AB(Y_s\beta) = A(Y_s)B(\beta) \neq 0.$$

Hence $Y_s\beta/\delta \in C$ - a contradiction. An analogous argument shows $Y_{\mathtt{s}} \notin B$.

Suppose $\frac{\delta_j Y_{s_j}}{\delta} \in A$ for some $j \in [v]$. Then

$$AB\left(\frac{\delta_j Y_{s_j}\beta}{\delta}\right) = A\left(\frac{\delta_j Y_{s_j}}{\delta}\right)B(\beta) \neq 0.$$

Hence $\frac{\delta_j Y_{s_j} \beta}{\delta^2} \in C$ - a contradiction. An analogous argument shows $\frac{\delta_j Y_{s_j}}{\delta} \notin B$.

Suppose $\delta_j \in A$ for some $j \in [v]$.

$$AB(\delta_j \beta) = A(\delta_j)B(\beta) \neq 0.$$

Hence $\delta_j \beta / \delta \in C$ - a contradiction. An analogous argument shows $\delta_j \notin B$. We have shown that no monomials involving the variables in V appear in A or B. It is left to show they do not appear in C either.

But if such a monomial M appeared in C, the monomial $M\delta$ (that also involves variables of V) appears in AB which means a monomial involving variables from V appears in A or B - a contradiction.

□

5 NILPs to SNARKs

We proceed to translate our NILP into a SNARK using bilinear groups. The translation is the same as in [5] and straightforward given previous works.

Group generators and generic oracle adversaries Let $F = \{\mathbb{F}_p(\lambda)\}_{\lambda \in \mathbb{N}}$ be a sequence of prime fields. A *group generator* $\mathcal{G}$ for F is an algorithm that given integer parameter λ runs in time poly(λ); and outputs groups $\mathbb{G}_1$, $\mathbb{G}_2$ written additively and $\mathbb{G}_T$ written multiplicatively all of order p, uniformly chosen generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and circuits of size poly(λ) for computing group operations in the three groups and a non-degenerate bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

For $a \in \mathbb{F}_p$, we denote below $[a]_1 := a \cdot g_1, [a]_2 := a \cdot g_2$. Before defining SNARKs we define a generic oracle adversary.

Definition 5.1 (Generic oracle adversary)**.** *A* generic $\mathcal{G}$-oracle adversary $\mathcal{A}$ *is a* $\mathbb{G}_1^*$*-oracle machine operating as follows. It begins with an explicit input* $\mathbf{x}$*, and encoded inputs* $[\sigma_1]_1 \in (\mathbb{G}_1), [\sigma_2]_2 \in (\mathbb{G}_2)$ *where* σ_i *is a vector over* $\mathbb{F}_p$*. It initializes an empty vector* U*. At each step* $\mathcal{A}$ *does the following.*

1. *Depending only on* $\mathbf{x}$ *and the value of the vector* U*, it chooses matrices* Π_1, Π_2, T *over* $\mathbb{F}_p$ *and possibly also a message* m*.*
2. *It computes* $y_1 := \Pi_1 \cdot v_1$ *where* v_1 *is the set of* $\mathbb{G}_1$ *elements it computed so far, and add the elements of* y_1 *to* v_1*. It does the analogous thing in* $\mathbb{G}_2$*.*

3. *If $\mathcal{A}$ is in interaction with a party P it may send P the message m, and if P replies with a set of $\mathbb{G}_1$ and $\mathbb{G}_2$ elements $\mathcal{A}$ may add them to v_1 and v_2.*

4. *It then checks if $v_1 \cdot (T \cdot v_2) = 0$ and adds the value 0 to U if so, and adds the value 1 to U otherwise.*

At a certain point it decides to terminate outputting matrices Π_1, Π_2.

Definition 5.2. *Let $\mathcal{G}$ be a group generator for a prime field $\mathbb{F}_p$, and $\mathsf{R} \subset (\mathbb{F}_p)$ a relation. An RO-SE-SNARK $\mathscr{S}$ (zero-knowledge Simulation Extractable Succint Non-interactive Argument of Knowledge in the Random Oracle model) against $\mathcal{G}$ for R consists of the following four possibly randomized algorithms.*

1. Gen *outputting a trapdoor τ and common reference string σ.*

2. P *that takes as input σ and $(\mathbf{x}, \omega) \in \mathsf{R}$ and outputs π.*

3. V *that takes as input a common reference string σ, an input $\mathbf{x}$, and a proof π, and outputs a value in $\{\mathsf{acc}, \mathsf{rej}\}$.*

4. $\mathrm{P}^{\mathrm{sim}}$ *taking as input $\mathbf{x}$, and trapdoor τ and outputting π. (It will be convenient to think of $\mathrm{P}^{\mathrm{sim}}$ as returning $(\mathbf{x}, \pi)$.)*

All algorithms are $\mathbb{G}_1^*$-oracle machines running in time $\mathrm{poly}(\lambda)$.

The quadruple of algorithms $\mathscr{S} = (\mathsf{Gen}, \mathrm{P}, \mathrm{V}, \mathrm{P}^{\mathrm{sim}})$ is satisfies

1. ***Completeness**: For any common reference string σ output by Gen, and any $(\mathbf{x}, \omega) \in \mathsf{R}$, if $\pi = \mathrm{P}(\sigma, \mathbf{x}, \omega)$ then $\mathrm{V}(\sigma, \mathbf{x}, \pi) = \mathsf{acc}$ with probability one.*

2. ***Perfect Zero-Knowledge:** For any output (τ, σ) of Gen and $(\mathbf{x}, \omega) \in \mathsf{R}$, the distribution of $\mathrm{P}^{\mathrm{sim}}(\tau, \mathbf{x})$ is identical to that of $\mathrm{P}(\sigma, \mathbf{x}, \omega)$.*

3. ***Simulation-Extractability:***

   Fix any output (τ, σ) of Gen. For any $\mathcal{G}$-generic oracle adversary $\mathcal{A}$ making $\mathrm{poly}(\lambda)$ steps, there exists a circuit χ of size $\mathrm{poly}(\lambda)$ such that the following holds: Suppose that $\mathcal{A}$ adaptively makes queries $Q = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$ to $\mathrm{P}^{\mathrm{sim}}(\tau, \cdot)$, that returns answers $A = \{(\mathbf{x}_1, \pi_1), \ldots, (\mathbf{x}_\ell, \pi_\ell)\}$. Finally $\mathcal{A}$ outputs a pair $(\mathbf{x}, \pi)$ and χ seeing the sequences Q, A and $\mathsf{rand}_{\mathcal{A}}$, outputs ω. The probability that

   - *$\mathcal{A}$ "wins": $(\mathbf{x}, \pi) \notin A$ and also $\mathrm{V}(\mathbf{x}, \pi) = \mathsf{acc}$, while*

- χ *"loses":* $(\mathbf{x}, \omega) \notin \mathsf{R}$

is $\text{negl}(\lambda)$.

5.1 SE-NILPs to SE-SNARKs

Given a group generator $\mathcal{G}$ and a NILP $\mathscr{N} = (\mathsf{Gen}, \mathrm{P}, \mathrm{V}, \mathrm{P}^{\mathrm{sim}})$ for the same prime field $\mathbb{F}_p$, we define the SNARK $\mathscr{N}_{\mathcal{G}} = (\mathsf{Gen}_{\mathcal{G}}, \mathrm{P}_{\mathcal{G}}, \mathrm{V}_{\mathcal{G}}, {\mathrm{P}^{\mathrm{sim}}}_{\mathcal{G}})$ as follows.

- $\mathsf{Gen}_{\mathcal{G}}$: Run Gen to obtain output $(\tau, \sigma_1, \sigma_2)$. Output $\tau, \sigma' = (\sigma'_1 = [\sigma_1]_1, \sigma'_2 = [\sigma_2]_2)$.
- $\mathrm{P}_{\mathcal{G}}$: Run the first phase of P on input $(\mathbf{x}, \omega)$ to obtain matrices Π_1, Π_2. Output $\pi_1 = \Pi_1 \cdot (\sigma'_1, Y), \pi_2 = \Pi_2 \cdot \sigma'_2$.
- $\mathrm{V}_{\mathcal{G}}$: Run the first phase of $\mathrm{V}(\sigma, \mathbf{x}, \pi)$ to obtain matrices $T_1, \ldots, T_d$. Output accept iff for each $i \in [d]$

$$(\sigma_1, Y, \pi_1) \cdot T_i(\sigma_2, \pi_2) = 0$$

- ${\mathrm{P}^{\mathrm{sim}}}_{\mathcal{G}}(\tau, \mathbf{x})$: Run $\mathrm{P}^{\mathrm{sim}}(\tau, \mathbf{x})$ to obtain (π_1, π_2). Output $[\pi_1]_1, [\pi_2]_2$.

The following is clear.

Theorem 5.3. *Suppose that $\mathcal{G}$ is a group generator for $\mathbb{F}_p$ and $\mathscr{N}$ is a degree $d = o(2^\lambda)$ AB-SE-NILP for R over $\mathbb{F}_p$. Then $\mathscr{N}_{\mathcal{G}}$ as defined above is a RO-SE-SNARK against $\mathcal{G}$ for R.*

Corollary 5.4. *Let $\mathbb{F}_p$ be a prime finite field. Suppose we are given a QAP relation R of degree d over $\mathbb{F}_p$, and a group generator $\mathcal{G}$ for $\mathbb{F}_p$, such that $d/|\mathbb{F}_p| = \text{negl}(\lambda)$. Then we can construct an RO-SE-SNARK against $\mathcal{G}$ for R.*

Signatures of Knowledge We do not give full details and definitions. Suppose m is the message we wish to sign. Then the construction of section 3 is modified in the descriptions of P,V and $\mathrm{P}^{\mathrm{sim}}$ simply by concatenating m to what is called s there.

Acknowledgements

We think Matthew D. Green for helpful conversations. We thank Jens Groth and Mary Maller for discussions on their construction.

References

[1] `https://github.com/scipr-lab/libsnark/tree/master/libsnark/zk_proof_systems/ppzksnark`.

[2] S. Bowe, A. Gabizon, and I. Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. `https://eprint.iacr.org/2017/1050`.

[3] G. Danezis, C. Fournet, J. Groth, and M. 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, 2014.

[4] R. Gennaro, C. Gentry, B. Parno, and M. 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, 2013.

[5] J. 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, 2016.

[6] J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 581–612, 2017.