Groth16 SNARKs are Randomizable and (Weakly) Simulation Extractable

Markulf Kohlweiss[1,2] and Mikhail Volkhov[1]

[1]The University of Edinburgh and [2]IOHK
mkohlwei@ed.ac.uk mikhail.volkhov@ed.ac.uk

Abstract

Due to its simplicity, succinctness, and performance, Groth16 is currently the most widely deployed succinct (zero-knowledge) argument of knowledge (SNARK) system. Groth16 is known to be perfectly zero-knowledge and knowledge sound in the generic (and algebraic) group model. However, the existing security arguments for Groth16 are silent about the soundness of the proof system in the presence of simulated proofs—a common requirement for both the composable security and game-hopping style security analysis of protocols built using such argument systems. This important gap let to a line of work on simulation-extractable, also called simulation knowledge sound, succinct proof systems. Groth16 itself cannot satisfy the strongest notion of simulation-extractability that implies proof non-malleability—in fact proofs are perfectly randomizable.

Surprisingly, in this short note we show that Groth16 does satisfy a weaker notion of simulation-extractability implying statement non-malleability. This property is often sufficient for typical applications that motivate the use of strong simulation-extractability. Notably, one can obtain UC security using efficient compilers.

1 Introduction

Succinct non-interactive arguments of knowledge (SNARK) have revolutionized the deployment of zero-knowledge proofs, particularly in the blockchain and cryptographic currency space [BCG+14; KMS+16; KKK20; BCG+18]. The ready availability of cryptographic libraries implementing SNARKs has also inspired other applications [NT16; DFKP16]. See also the application chapter of [ZKPce].

Due to its exceptional performance and simplicity the most widely deployed SNARK proof system is perhaps Groth16 [Gro16]. Naturally, this is susceptible to change, especially if its security is undermined by quantum attacks. That withstanding, however, due to its near optimal proof size and verification performance, Groth16 is likely to be a mainstay of cryptographic deployments, maybe comparable to ElGamal encryption [Gam85] and Schnorr [Sch91] or DSA [PUB93] signature schemes. In this short note we identity and close what we believe is a small but important gap in the security analysis of Groth16, namely its malleability and the limits of said malleability.

Arguably, the strongest non-malleability property for SNARK systems is *simulation-extractability* (SE) [Sah99; DSDCO+01], a security notion that extends knowledge-soundness (KS) by allowing the adversary to also access the simulaton oracle. One of the important properties of this notion is that its straight-line, blackbox variant is necessary to achieve Universally Composable (UC [Can01]) security, as shown more generally in [CLOS02; GOS06; Gro06] for non-interactive zero-knowledge (NIZK) proof systems. Moreover, it is also needed in game-hopping style proofs [Sho04] in which one game hop introduces the simulator and a subsequent game hop relies on knowledge soundness [KMS+16; CDD17].

In this work we focus on the weaker notion of simulation-extractability, that allows for the limited malleability of proofs, which we call weak simulation-extractability (weak-SE). Note that the notion of weak simulation extractability from [FKMV12] is unrelated. The "weakening" there refers to a non-blackbox and non-straight line flavor of simulation extraction in the Random Oracle model. Rather, weak-SE and SE of *proof systems* are in analogy to chosen message attack (CMA) and strong CMA unforgeability of *signatures*. Indeed, in weak-SE it is the statement rather than the proof that cannot be mauled, a weak-SE based SNARKY signature scheme for a hard key-pair relation, give rise to CMA, rather than strong CMA secure, signature scheme.

Our contribution. In this short note we show that Groth16 is both weakly-simulation extractable and perfectly and efficiently randomizable. As in Groth16 the randomization of proofs obtained from the simulator are distributed just like freshly generated proofs, this is seemingly the strongest extractability property that one can hope for.

In the algebraic group model, however, we can show something even stronger, namely the extractor can either obtain a witness or point to the unique simulated proof that was randomized to obtain the proof produced by the adversary. Consequently, even if the adversary queries multiple proofs for the same statement, they cannot be combined to form a new proof of the same statement. Therefore, weak-SE Groth16 can be directly lifted to (weak) blackbox SE, which is required by UC, using the technique explained in [Bag19], improving the performance of the resulting SNARK compared to Groth and Maller SNARK used in [Bag19].

Related Work. Simulation-extractability applies both to CRS-based and random-oracle (RO) based NIZKs. NIZKs obtained from Σ-protocols using Fiat-Shamir heuristic in the random oracle (RO) model, are showed to always satisfy simulation-extractability [FKMV12]. In this work we focus on simulation-extractability of CRS-based NIZKs, and on SNARKs in particular.

SE SNARKs have been discovered only recently. Groth and Maller [GM17] presented the first construction in 2017, targeting SAP, together with a lower bound of three group elements for the proof size, and two equations for verification, for all *non-interactive linear proofs* (NILP) based SNARKs, which covers many previously known pairing-based SNARKs, including [Gro16; GM17]. Bowe and Gabizon [BG18] provide a RO-based variant of Groth16 for QAP that is simulation-extractable, and has five group elements and two verification equations. Lipmaa [Lip19] presents a different technique that allows to lift known SNARKs for QAP and the three other arithmetisation techniques from the QAP family (namely, SAP, SSP, and QSP), together with a simpler security proof. Kim, Lee, and Oh [KLO19] present a SNARK for QAP with three elements but just a single verification equation, avoiding the lower bound of Groth and Maller by using a random oracle in addition to a knowledge extraction assumptions and a CRS.

General transformations and UC. A generic transformation that lifts ordinary NIZKs to be simulation extractable has been known since [DSDCO+01] at least. Along this direction, [KZM+15b; KZM+15a] extend, analyse, and optimise this transformation technique, while Atapoor and Baghery [AB19] apply it directly to Groth16 and evaluate the efficiency of the resulting strong SE argument. The transformation from non-blackbox SE to blackbox SE is analysed by Baghery [Bag19], with particular focus on (strong-SE) SNARK by Groth and Maller, although this technique should also work for lifting non-blackbox weak-SE to blackbox weak-SE. Other generic transformations take into account CRS subversion and updatability [ARS20; BS20].

Regarding UC functionalities for NIZKs, it has been shown that a (non-malleable) $\mathcal{F}_{\mathsf{NIZK}}$ functionality can be realised using (strong) blackbox-extractable SE NIZKs [CLOS02; Gro06] assuming static corruption. Kosba et al. [KZM+15b; KZM+15a] suggest their own variant of $\mathcal{F}_{\mathsf{WEAK-NIZK}}$ without proving that a weak-SE NIZK can realise it.

Weaker simulation extraction notions. Although (strong) SE is sometimes a desirable property, weak-SE can be sufficient for UC applications, for instance in Hawk [KMS+16], as argued in [KZM+15b; KZM+15a].

Hawk uses SE NIZKs directly as a raw primitive (without employing a functionality), and it suggests to use non-succinct strong SE NIZK, since no other candidates were known at that time. Kosba et al. [KZM+15b; KZM+15a] point out that weak-SE NIZK can be used instead, without providing a formal proof. Lipmaa's [Lip19] presents an SE notion that is tag-based, although the construction presented prevents standard randomization.

2 Preliminaries

2.1 Notation

PPT stands for (uniform) probabilistic polynomial-time. We denote the security parameter by $\lambda \in \mathbb{N}$. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible, if $f < 1/p(n)$ for all polynomials $p(n)$ and $n \in \mathbb{N}$ big enough, which we denote as $f = \mathrm{negl}(\lambda)$. For a distribution X we denote random sampling by $x \xleftarrow{\$} X$, and when this notation is used with a finite set S, $x \xleftarrow{\$} S$ denotes sampling uniformly from S. We write vectors in bold, and write $\boldsymbol{a} \cdot \boldsymbol{b}$ for the inner product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

When working with polynomials, we generally use upper case letters for indeterminates as X, Y, Δ, X_γ, and lower case for concrete values x, y, δ, γ. We use vector notation to denote a list of formal variables, so for $\boldsymbol{X} = X_1, \ldots, X_n$, we write $P(\boldsymbol{X}) \in \mathbb{F}[X_1 \ldots X_n] = \mathbb{F}[\boldsymbol{X}]$ for a polynomial in these variables, and for a $\boldsymbol{x} \in \mathbb{F}^n$, $P(\boldsymbol{x})$ will denote the polynomial evaluation $P(x_1 \ldots x_n)$. For a polynomial $P(\boldsymbol{X})$ and a monomial $M = X_1^{b_1} X_2^{b_2} \cdots X_n^{b_n}$, $P_{[M]}$ will denote the coefficient of $P(\boldsymbol{X})$ at M, that is $P(\boldsymbol{X}) = \sum_M P_{[M]} M$.

Bilinear groups. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot), p)$ be a Type III bilinear group (asymmetric, with $\mathbb{G}_1 \neq \mathbb{G}_2$ and without any efficiently computable nontrivial homomorphism in either direction between $\mathbb{G}_1$ and $\mathbb{G}_2$, according to the classification of [GPS08]) of prime order p with generators G, H, and $e(G, H)$ respectively. The pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map. We will write $\mathbb{G}_1$ and $\mathbb{G}_2$ additively, but $\mathbb{G}_T$ multiplicatively. It will be

convenient to use square brackets notation to represent group elements by specifying their exponents: $[a]_\iota \triangleq [a]G_\iota$. We will denote the (exponent-level) pairing for the square brackets notation as $[a]_1 \bullet [b]_2 \triangleq e([a]G, [b]H)$, and on that instance $\mathbb{G}_T$ notation is additive: $[a]_1 \bullet [b]_2 + [c]_1 \bullet [d]_2 = [ab + cd]_T$ instead of $e([a]G, [b]H) \cdot e([c]G, [d]H) = e(G, H)^{ab+cd}$.

When $\boldsymbol{a}$ is a vector of values $a_i \in \mathbb{Z}_p$, we will overload the square brackets notation, and denote a vector of $[a_i]_\iota$ by $[\boldsymbol{a}]_\iota$. In the same way we will overload $[\{a, b, c, \ldots\}]_\iota = \{[a]_\iota, [b]_\iota, [c]_\iota, \ldots\}$ for sets. When set or vector A contains elements from several groups, we will denote it by combining all the group indices in the subscript, e.g. $[A]_{1,2,T}$ if A contains elements from all the three groups.

2.2 Randomization and Simulation-Extractability

Weak simulation extractability is an extension of knowledge-soundness, that allows the adversary access to the simulator, and tolerates randomization of proofs. We first recall the definitions of KS and (derivation-private) randomization.

For an algorithm $\mathcal{P}$ we define an *execution transcript* $\mathsf{trans}_\mathcal{P}$ to be a structure containing private coins of $\mathcal{P}$ and a list of $\mathcal{P}$'s inputs and outputs, including its interactions with any oracles that it is provided with. Note that the fact we are using $\mathsf{trans}_\mathcal{P}$ implies non-blackbox access to $\mathcal{P}$.

Definition 2.1 (Knowledge Soundness)**.** *Let* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ *be a NIZK for the relation* R*. We say that* NIZK *is* knowledge sound *if for any PPT adversary* $\mathcal{A}$ *there exists a polynomial time extractor* $\mathcal{X}_\mathcal{A}$ *such that* $\mathsf{Adv}^{\mathsf{weak-SE}}_{\mathcal{A},\mathcal{X}_\mathcal{A}}(\lambda) \triangleq$

$$\Pr\left[\ (\boldsymbol{\sigma}, \tau) \leftarrow \mathsf{Setup}(R, 1^\lambda); (\phi, \pi) \leftarrow \mathcal{A}(\boldsymbol{\sigma}); w \leftarrow \mathcal{X}_\mathcal{A}(\mathsf{trans}_\mathcal{A}); \mathsf{Verify}(\boldsymbol{\sigma}, \phi, \pi) = 1 \wedge (\phi, w) \notin R\ \right] = \mathrm{negl}(\lambda)$$

where $\mathsf{trans}_\mathcal{A}$ *is a transcript of the execution of* $\mathcal{A}$*.*

We will call the proof system for the relation $\mathcal{R}$ *randomizable*, if there exists a (non-trivial) PPT procedure Rand such that $\Pr[\mathsf{Verify}(\mathsf{crs}, \phi, \mathsf{Rand}(\pi))] = 1$ for all honestly generated proofs π for crs and ϕ.

Definition 2.2 (Derivation-Private Randomization)**.** *The proof system is (perfectly)* derivation-private *with respect to the randomization transformation* Rand*, if for all* crs *for* λ *and* R *and* $(x, w) \in R$*:*

$$\{\mathsf{Prove}(\mathsf{crs}, x, w)\}_\lambda = \{\mathsf{Rand}(\mathsf{Prove}(\mathsf{crs}, \mathsf{x}, \mathsf{w}))\}_\lambda$$

where the randomness is over the random variables used in Prove *and* Rand*.*

Note that a proof system can in principal have several randomization procedures, so the notion is defined with respect to a particular one. The Rand for Groth16 will be parametrised with two random variables r_1, r_2, and it will be still valid when setting one of the two to zero (thus, at least three randomization procedures are known), but in this case the transformation will lose derivation-privacy which the original Rand achieves.

Definition 2.3 (Weak Simulation Extractability, [KZM+15b])**.** *Let* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ *be a NIZK for the relation* R*. We say that* NIZK *is* weakly simulation extractabile *if for any PPT adversary* $\mathcal{A}$ *there exists a polynomial time extractor* $\mathcal{X}_\mathcal{A}$ *such that* $\mathsf{Adv}^{\mathsf{weak-SE}}_{\mathcal{A},\mathcal{X}_\mathcal{A}}(\lambda) \triangleq$

$$\Pr\left[\begin{array}{c|l} (\boldsymbol{\sigma}, \tau) \leftarrow \mathsf{Setup}(R, 1^\lambda); (\phi, \pi) \leftarrow \mathcal{A}^{\mathcal{S}_{\boldsymbol{\sigma},\tau}}(\boldsymbol{\sigma}); & \mathsf{Verify}(\boldsymbol{\sigma}, \phi, \pi) = 1 \wedge \\ w \leftarrow \mathcal{X}_\mathcal{A}(\mathsf{trans}_\mathcal{A}) & (\phi, w) \notin R \wedge \phi \notin Q \end{array}\right] = \mathrm{negl}(\lambda)$$

where $\mathcal{S}_{\boldsymbol{\sigma},\tau}(\phi)$ *is a simulator oracle that calls* $\mathsf{Sim}(\boldsymbol{\sigma}, \tau, \phi)$ *internally, and also records* ϕ *into* Q*, and* $\mathsf{trans}_\mathcal{A}$ *is a transcript of the execution of* $\mathcal{A}$*.*

The important distinction between this notion and *strong* simulation-extractability lies in the last condition of the presented security game. In strong SE one would require $(\phi, \pi) \notin Q$, where $\mathcal{S}$ would record pairs of queried instances and simulated proofs. If NIZK is randomizable, $\mathcal{A}$ could just pass re-randomized simulated proof for a false instance and win the strong SE game. This is forbidden, thus the strong-SE scheme must be non-malleable. Honest proofs are also non-randomizable, otherwise zero-knowledge would not hold. Weak-SE relaxes this non-malleability requirement by allowing to produce $\pi' \neq \pi$ for the simulated (and thus also real) proof π.

Lemma 2.1. *The notion of weak-SE is not trivial: there exist weak-SE NIZKs for circuit satisfiability that are not strongly simulation extractable.*

The proof of this statement is presented in the Appendix A and is based on the randomizability of Groth-Sahai proofs [GS08].

2.3 Algebraic Proof Techniques for NILPs based SNARKs

The purpose of this section is to give general lemmas for SNARKs based on non-interactive linear proofs with algebraic assumptions, that will simplify the proofs of this note. We do not define NILPs formally, since our intention is to cover pairing based SNARKs in the CRS model, and although the definition would capture the required class of SNARKs, we formalise the involved properties separately, thus minimising the statements we prove.

Following [FKL18; Lip19], we say that the algorithm $\mathcal{A}$ is *algebraic*, if there is a way to express any element it returns as a linear combination of elements it has seen before with known (extracted) coefficients. Security against algebraic adversaries can be formalised either as a standard model non-blackbox knowledge-extraction assumption [BV98; PV05; Lip19], or by lifting this assumption and defining a separate cryptograpic model as done in the AGM [FKL18], similar to the generic group or RO model. We are following the extraction assumption style from [Lip19], without considering the stronger hashed version that additionally allows $\mathcal{A}$ to sample random elements in $\mathbb{G}$ without knowing their exponents.

Definition 2.4 (Algebraic Algorithm, [Lip19])**.** *A PPT algorithm $\mathcal{A}$ is* algebraic *with respect to a cyclic group $\mathbb{G}_\iota$ of prime order p, if there exists a polynomial time extractor $\mathcal{X}_{\mathcal{A}}^{\mathsf{alg}}$ returning a coefficients matrix K, such that for all m and all efficiently sampleable distributions $\mathcal{D}$ over $(\mathbb{Z}_p^*)^m$, $\mathsf{Adv}^{\mathsf{ak}}_{\mathbb{G},\mathcal{D},\mathcal{A},\mathcal{X}_{\mathcal{A}}^{\mathsf{alg}}}(\lambda) \triangleq$*

$$\Pr\left[\boldsymbol{\sigma} \xleftarrow{\$} \mathcal{D}_\lambda; \boldsymbol{e} \xleftarrow{\$} \mathcal{A}([\boldsymbol{\sigma}]_\iota); K \leftarrow \mathcal{X}_{\mathcal{A}}^{\mathsf{alg}}(\mathsf{trans}_{\mathcal{A}}) : \boldsymbol{e} \neq [K\boldsymbol{\sigma}]_\iota\right] = \mathrm{negl}(\lambda)$$

It is easy to see how this definition extends to the asymmetric bilinear group setup (now $\mathcal{X}_{\mathcal{A}}^{\mathsf{alg}}$ should return K with $m_1 + m_2$ rows, and $(\boldsymbol{e}_1\ \boldsymbol{e}_2)^T = \left[K(\boldsymbol{\sigma}_1\ \boldsymbol{\sigma}_2)^T\right]_{1,2}$), and to the case when $\mathcal{A}$ obtains elements from an oracle ($\mathsf{trans}_{\mathcal{A}}$ captures communication with it). In the soundness and KS games, adversary $\mathcal{A}$ is limited to the CRS elements only, and in the simulation-based setting $\mathcal{A}$ also sees the simulated proof elements.

Definition 2.5 (q-Discrete Logarithm Assumption)**.** *Let $\mathbb{G}$ be a cyclic group with generator G. We say that q-**dlog** holds in $\mathbb{G}$, if for all PPT $\mathcal{A}$, $\mathsf{Adv}^{q-\mathbf{dlog}}_{\mathbb{G},\mathcal{A}} \triangleq$*

$$\Pr\left[x \xleftarrow{\$} \mathbb{Z}_p^*; z \xleftarrow{\$} \mathcal{A}([x]G, [x^2]G, \ldots, [x^q]G) : x = z\right] = \mathrm{negl}(\lambda)$$

Since we will be working with asymmetric bilinear groups, we will say that (m_1, m_2)-**dlog** holds, when m_ι-**dlog** holds in $\mathbb{G}_\iota$ for $\iota \in \{1, 2\}$.

For the following lemma we assume a two-step sampling procedure $S_\lambda = (\mathcal{D}_\lambda, \mathsf{Setup}_\lambda)$, where an effectively sampleable distribution $\mathcal{D}_\lambda$ defines a set of trapdoors $\boldsymbol{\tau} \in (\mathbb{Z}_p^*)^n$, and a polynomial time deterministic procedure $\mathsf{Setup}_\lambda(\tau)$ generates elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ as polynomials of $\boldsymbol{\tau}$. Let $\boldsymbol{T} = T_1, \ldots, T_n$ be a set of formal variables corresponding to the trapdoors. This setup models CRS generation, that is Setup_λ constructs two sets of elements $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$, where every $\sigma_{\iota,i} = P_{\iota,i}(\boldsymbol{\tau})$ for some $\{P_{\iota,i}(\boldsymbol{T})\}_{\iota,i}$.

Lemma 2.2 (Algebraic Verification Satisfiability)**.** *Let $\boldsymbol{E} = (E_{1,1}, \ldots, E_{1,m_1}, E_{2,1}, \ldots, E_{2,m_2})$ be a vector of formal variables in $\mathbb{Z}_p$, where $E_{\iota,i}$ represents an exponent value of some $[E_{\iota,i}]_\iota \in \mathbb{G}_\iota$. Let $V(\boldsymbol{E})$ be a pairing equation, expressed in the $\mathbb{G}_T$ exponent*[1]*.*

For all algebraic PPT $\mathcal{A}$, and all two-step sampling procedures S_λ with trapdoor variables $\boldsymbol{T}$:

$$\Pr\left[\begin{array}{l}\boldsymbol{\tau} \xleftarrow{\$} \mathcal{D}_\lambda; (\boldsymbol{\sigma}_1\ \boldsymbol{\sigma}_2) \leftarrow \mathsf{Setup}_\lambda(\boldsymbol{\tau}); [\boldsymbol{e}]_{1,2} \xleftarrow{\$} \mathcal{A}([\boldsymbol{\sigma}_1]_1, [\boldsymbol{\sigma}_2]_2); \\ \qquad\qquad K \leftarrow \mathcal{X}_{\mathcal{A}}^{\mathsf{alg}}([\boldsymbol{\sigma}_1]_1, [\boldsymbol{\sigma}_2]_2, \mathsf{trans}_{\mathcal{A}})\end{array} : V(\boldsymbol{e}) = 0 \wedge V\big(K(\mathsf{Setup}_\lambda(\boldsymbol{T}))\big) \neq 0\right] = \mathrm{negl}(\lambda)$$

*assuming (d_1, d_2)-**dlog** holds, where $d_\iota = \max_i(\deg P_{\iota,i}(\boldsymbol{T}))$ of Setup_λ, and $V\big(K(\mathsf{Setup}_\lambda(\boldsymbol{T}))\big)$ denotes $V(\boldsymbol{e})$ interpreted as a polynomial over $\boldsymbol{T}$. The probability is quantified over $\mathcal{D}_\lambda$ and the private coins of $\mathcal{A}$.*

In other words, the lemma says that $\mathcal{A}$ has negligible success in constructing $\boldsymbol{e}$ as linear combination of CRS elements such that $V(\boldsymbol{e})$ evaluates to zero, but $V'(\boldsymbol{T}) = V(K \cdot \mathsf{Setup}_\lambda(\boldsymbol{T}))$ is not identically zero as a polynomial in $\boldsymbol{T}$. It is not hard to generalise this statement for an adversary $\mathcal{A}$ that also obtains some group elements through queries to oracles, or for multiple equations that $\mathcal{A}$ aims to satisfy.

The proof of this statement is based on the observation that if $\mathcal{A}$ is "blind" to $\boldsymbol{\sigma}$, then by the Schwartz-Zippel lemma $\mathcal{A}$ cannot guess $\boldsymbol{e}$ such that $V(\boldsymbol{e}) = 0 \wedge V'(\boldsymbol{T}) \neq 0$; and if $\mathcal{A}$ otherwise makes $\boldsymbol{e}$ depend on the *particular* choice of $\boldsymbol{\sigma}$, we can embed (d_1, d_2)-**dlog** into $\boldsymbol{\sigma}$ and solve it. The proof sketch builds on [FKL18] and is deferred to Appendix B.

[1] That is, $V(\boldsymbol{E}) = \sum_i \Gamma_i t_{1,i} t_{2,i}$ for $t_{\iota,i}$ being either some $E_{\iota,i}$ or a constant from $\mathbb{Z}_p^*$, and $\Gamma_i \in \mathbb{Z}_p^*$. This corresponds to the base group elements pairing equation $\prod_i e(z_{1,i}, z_{2,i})^{\Gamma_i} = 1$ with $z_{\iota,i}$ being either variable or constant group elements $[t_{\iota,i}]_\iota$.

Another small remark is that the lemma is defined with respect to positive powers polynomials, while Groth16 CRS is defined for Laurent polynomials. This obstacle is easy to overcome – as shown in [FKL18], it is enough to modify the group generator by raising it to a certain trapdoor power such that all the negative powers cancel out. This does not change the main statement of Lemma 2.2, although it slightly increases the required degree of (d_1, d_2)-**dlog**[2].

3 Weak-SE of Groth16

Groth16 has been shown to be both knowledge sound (KS) and randomizable. The main result of this section is that Groth16 SNARK is additionally weakly simulation extractable (SE). We present both KS and weak-SE proofs – although KS was shown in the original work. Our SE proof builds on KE, and is simpler to understand in conjunction with our KS proof. We first remind the reader how Groth16 is constructed.

Quadratic arithmetic programs (QAP) Recall that a QAP consists of the quotient polynomial $t(x)$ of degree n, and three sets of polynomials $\{u_i(X)\}_{i=0}^m, \{v_i(X)\}_{i=0}^m$ and $\{w_i(X)\}_{i=0}^m$ of degree $n-1$ that define the circuit being computed. A particular wire assignment $\{a_i\}_{i=0}^m$, that we split into l instance wires, and the remaining $m-l$ witness wires, satisfies the QAP if and only if $(\sum_{i=0}^m a_i u_i(X))(\sum_{i=0}^m a_i v_i(X)) - (\sum_{i=0}^m a_i w_i(X)) = h(X)t(X)$ for some $h(X)$ of degree $n-2$. That is, $t(x)$ divides the left hand side of the equation.

Groth16 reference string The CRS consists of the following two sets of elements, in $\mathbb{G}_1$ and $\mathbb{G}_2$ correspondingly:

$$\sigma_1 : \ [\alpha, \beta, \delta]_1, \{[x^i]_1\}_{i=0}^{n-1}, \left\{\left[\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\right]_1\right\}_{i=0}^{l}, \left\{\left[\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}\right]_1\right\}_{i=l+1}^{m}, \left\{\left[\frac{x^i t(x)}{\delta}\right]_1\right\}_{i=0}^{n-2},$$

$$\sigma_2 : \ [\beta, \gamma, \delta]_2, \{[x^i]_2\}_{i=0}^{n-1}$$

with $\alpha, \beta, \gamma, \delta, x$ being trapdoors, uniformly chosen from $\mathbb{Z}_p^*$. Note that the group generators G and H are available, and explicitly included as $[x^0]_\iota$.

Groth16 verification equation Let $\boldsymbol{\phi} = \{a_i\}_{i=0}^l$ and $\boldsymbol{w} = \{a_i\}_{i=l+1}^m$ be the instance and the witness for which we are constructing the proof. The verification equation, parametrised by three proof elements $[A]_1, [B]_2, [C]_1$ is:

$$[A]_1 \bullet [B]_2 = [\alpha]_1 \bullet [\beta]_2 + [\sum_{i=0}^{l} a_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}]_1 \bullet [\gamma]_2 + [C]_1 \bullet [\delta]_2$$

It can be represented in a more convenient way as a polynomial in the $\mathbb{G}_T$ exponent, over $\mathbb{Z}_p[A, B, C]$:

$$V(A, B, C) = AB - \alpha\beta - \varphi(\boldsymbol{\phi})\gamma - C\delta = 0 \tag{1}$$

where $\varphi(\boldsymbol{\phi}) = \sum_{i=0}^l a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\gamma$.

Groth16 proof generation and simulation The proof $([a]_1, [b]_2, [c]_1)$ is constructed in the following way:

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

$$c = \sum_{i=l+1}^{m} \frac{a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\delta} + \frac{h(x)t(x)}{\delta} + as + br - rs\delta$$

where $(r, s) \xleftarrow{\$} \mathbb{Z}_p^2$ is a pair of uniformly selected random values. We note that although it is possible and even more practically feasible for some applications to use a variant of Groth16 with random values set to zero, thus sacrificing zero-knowledge, our primary interest covers the original, randomizable version of Groth16.

Following the distribution defined by the honest proof generation, the simulator also uses two random elements $(\mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^2$ (as opposed to just one for SAP in Groth and Maller SNARK), by setting $a = \mu, b = \nu$, and

$$c = \frac{\mu\nu - \alpha\beta - \sum_{i=0}^{l} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\delta}$$

[2]In case of Groth16, we multiply by $\gamma\delta$, thus $[x^{n-2}t(x)/\delta]_1$ becomes $[\gamma x^{n-2}t(x)]_1$ of degree $2n-1$, hence $d_1 = 2n-1$

Groth16 is randomizable and thus malleable It is known that Groth16 is randomizable, which is a property of the proof verification equations. If $\pi = (a, b, c)$ satisfies 1, then so does $\pi' = \mathsf{Rand}(\pi) = (a', b', c')$ where:

$$a' = \frac{1}{r_1}a \qquad b' = r_1 b + r_1 r_2 \delta \qquad c' = c + r_2 a \tag{2}$$

The correctness of this randomization is trivial to verify:

$$\begin{aligned} a'b' - \alpha\beta - \varphi(\boldsymbol{\phi})\gamma - c'\delta &= a(b + r_2\delta) - \alpha\beta - \varphi(\boldsymbol{\phi})\gamma - (c + r_2 a)\delta \\ &= ab - \alpha\beta - \varphi(\boldsymbol{\phi})\gamma - c\delta \end{aligned}$$

Although the randomization of Groth16 is mentioned in many works that explore simulation extraction of related constructions, it is not known whether proofs can be mauled in any other way. In particular, this can depend on the restrictions placed on the adversary. In this work, we show as a corollary of Theorem 3.2 that for algebraic adversaries the randomization of proofs is the most general form of malleability for Groth16.

Theorem 3.1 ([FKL18])**.** *Groth16 achieves knowledge soundness against algebraic adversaries under the* $(2n-1, n-1)$***-dlog*** *assumption.*

Proof. We start by assuming a certain number of variables to be unknown to $\mathcal{A}$, in this particular case these are just the CRS trapdoors $\boldsymbol{\tau} = (\alpha, \beta, \gamma, \delta, x)$. We rely on Lemma 2.2. When $\mathcal{A}$ presents the proof $\pi = ([a]_1, [b]_2, [c]_1)$ that satisfies the verification equation (1), that is $V(\pi) = 0$, we conclude that $\mathcal{A}$ could not come up with π satisfying V unless for $V' = V(K \cdot \mathsf{Setup}_\lambda(\boldsymbol{T}))$ we have $V'(\boldsymbol{T}) = 0$ as a polynomial. Then we, step by step, analyze the coefficients K of the verification equation, by relying on the property that every monomial coefficient of the equation is zero (because the polynomial is constant zero). This is the most technical part of the proof, and we remind the reader that the other part that provides the reduction to $(2n-1, n-1)$**-dlog** is deferred generically to Lemma 2.2.

The matrix K contains a representation of A, B, and C as linear combination of public CRS elements:

$$\begin{aligned} A = {} & A_1\alpha + A_2\beta + A_3\delta + \sum_{i=0}^{n-1} A_{4,i}x^i + \sum_{i=0}^{l} A_{5,i}\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} + \\ & \sum_{i=l+1}^{m} A_{6,i}\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \sum_{i=0}^{n-2} A_{7,i}\frac{x^i t(x)}{\delta} \\ B = {} & B_1\beta + B_2\gamma + B_3\delta + \sum_{i=0}^{n-1} B_{4,i}x^i \\ C = {} & C_1\alpha + C_2\beta + C_3\delta + \sum_{i=0}^{n-1} C_{4,i}x^i + \sum_{i=0}^{l} C_{5,i}\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} + \\ & \sum_{i=l+1}^{m} C_{6,i}\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \sum_{i=0}^{n-2} C_{7,i}\frac{x^i t(x)}{\delta} \end{aligned}$$

We let $\boldsymbol{C} = (A_1, \ldots, A_{7,n-2}, \ldots, B_{4,n-1}, \ldots, C_{7,n-2})$ denote this set of variables serving as linear combination coefficients. In the following we will write CRS trapdoors as concrete values $(\alpha, \beta, \ldots, x)$, though they can be equally interpreted as formal variables $(X_\alpha, X_\beta, \ldots, X_x)$; we will avoid these former notation for convenience, since the main variables in scope that the system of equation is over are $\{A_i\}, \{B_i\}, \{C_i\}$, and we use trapdoor variables only to show how to form the system. This is, however, an important distinction: When we write $P(\alpha, x) = 0$, we imply $P(X_\alpha, X_x)$ is constant zero, and not just zero at (α, x).

For each monomial M, we write out the corresponding monomial coefficient $V'_{[M]}$ as an equation $V'_{[M]} = 0$, and iteratively simplify the system of equations in $\boldsymbol{C}$. To simplify the proof, the 'monomials' we consider implicitly contain sums of powers of x [3], thus x^i will appear in coefficients. We start with examining the following equations:

$$\begin{aligned} &\alpha\beta \text{ in } AB - \alpha\beta : A_1 B_1 = 1 \implies A_1 \neq 0, B_1 \neq 0 \\ &\qquad \beta^2 \text{ in } AB : A_2 B_1 = 0 \implies A_2 = 0 \end{aligned}$$

[3]For monomial M instead of analysing $V'_{[M]} = 0$ we set $\tilde{V}'_{[M]} = \sum_i V_{[Mx^i]} = 0$. This is still a valid statement, since $V'(\boldsymbol{T}) = 0$ implies $V'_{[Mx^i]} = 0$ for each i, so each sum over x^i for M not containing any powers of x is also zero. It is always possible to split $\tilde{V}'_{[M]}$ further as $(\tilde{V}'_{[M]})_{[x^i]}$, extracting coefficients of x^i from it. We will do so implicitly in the "different spans of x powers" argument in the proof.

$$\alpha\gamma : A_1 B_2 = 0 \Longrightarrow B_2 = 0$$

$$\beta^2/\delta : \Big(\sum_{i=l+1}^{m} A_{6,i}u_i(x)\Big)B_1 = 0 \Longrightarrow \sum_{i=l+1}^{m} A_{6,i}u_i(x) = 0$$

$$\beta\alpha/\delta : \Big(\sum_{i=l+1}^{m} A_{6,i}v_i(x)\Big)B_1 = 0 \Longrightarrow \sum_{i=l+1}^{m} A_{6,i}v_i(x) = 0$$

$$\beta/\delta \text{ in } AB : \Big(\sum_{i=l+1}^{m} A_{6,i}w_i(x) + \sum_{i=0}^{n-2} A_{7,i}x^i t(x)\Big)B_1 + \Big(\sum_{i=l=1}^{m} A_{6,i}u_i(x)\Big)\Big(\sum_{i=0}^{n-1} B_{4,i}x^i\Big) = 0 \wedge$$

$$1/\delta : \Big(\sum_{i=l+1}^{m} A_{6,i}w_i(x) + \sum_{i=0}^{n-2} A_{7,i}x^i t(x)\Big)\Big(\sum_{i=0}^{n-1} B_{4,i}x^i\Big) = 0$$

$$\Longrightarrow \sum_{i=0}^{n-2} A_{7,i}x^i t(x) = 0 \wedge \sum_{i=l+1}^{m} A_{6,i}w_i(x) = 0$$

If $(\sum_{i=0}^{n-1} B_{4,i}x^i) = 0$ then from β/δ we have $\sum_{i=l+1}^{m} A_{6,i}w_i(x) + \sum_{i=0}^{n-2} A_{7,i}x^i t(x) = 0$,

otherwise from $1/\delta$ we have $\sum_{i=l+1}^{m} A_{6,i}w_i(x) + \sum_{i=0}^{n-2} A_{7,i}x^i t(x) = 0$,

since the sums have different spans of x^i powers, $\sum_{i=0}^{n-2} A_{7,i}x^i t(x) = 0$ and $\sum_{i=l+1}^{m} A_{6,i}w_i(x) = 0$.

$$\beta^2/\gamma \text{ in } AB : \Big(\sum_{i=0}^{l} A_{5,i}u_i(x)\Big)B_1 = 0 \Longrightarrow \sum_{i=0}^{l} A_{5,i}u_i(x) = 0$$

$$\beta\alpha/\gamma : \Big(\sum_{i=0}^{l} A_{5,i}v_i(x)\Big)B_1 = 0 \Longrightarrow \sum_{i=0}^{l} A_{5,i}v_i(x) = 0$$

$$\beta/\gamma : \Big(\sum_{i=0}^{l} A_{5,i}w_i(x)\Big)B_1 + \Big(\sum_{i=0}^{l} A_{5,i}u_i(x)\Big)\Big(\sum_{i=0}^{n-1} B_{4,i}x^i\Big) = 0 \wedge$$

$$1/\gamma : \Big(\sum_{i=0}^{l} A_{5,i}w_i(x)\Big)\Big(\sum_{i=0}^{n-1} B_{4,i}x^i\Big) = 0 \Longrightarrow \sum_{i=0}^{l} A_{5,i}w_i(x) = 0 \text{ from } \beta/\gamma \text{ and } 1/\gamma \text{ as with } 1/\delta$$

We now consider the following three monomials (β, α, and 1 that is only x powers) that we will call critical (and, respectively, the related equations too). Critical equations contain parts of the QAP, and we will eventually extract the witness from them. The underlined coefficients are already known to be zero, and thus the related sums are immediately cancelled:

$$\beta \text{ in } AB - \varphi(\boldsymbol{\phi})\gamma - C\delta :$$

$$\Big(\sum_{i=0}^{n-1} A_{4,i}x^i\Big)B_1 + \Big(\sum_{i=0}^{n-1} B_{4,i}x^i\Big)\underline{A_2} + \underline{\Big(\sum_{i=0}^{l} A_{5,i}u_i(x)\Big)\underline{B_2}} + \underline{\Big(\sum_{i=l+1}^{m} A_{6,i}u_i(x)\Big)}B_3 = \sum_{i=0}^{l} a_i u_i(x) + \sum_{i=l+1}^{m} C_{6,i}u_i(x)$$

$$\alpha \text{ in } AB - \varphi(\boldsymbol{\phi})\gamma - C\delta :$$

$$\Big(\sum_{i=0}^{n-1} B_{4,i}x^i\Big)A_1 + \underline{\Big(\sum_{i=0}^{l} A_{5,i}v_i(x)\Big)\underline{B_2}} + \underline{\Big(\sum_{i=l+1}^{m} A_{6,i}v_i(x)\Big)}B_3 = \sum_{i=0}^{l} a_i v_i(x) + \sum_{i=l+1}^{m} C_{6,i}v_i(x)$$

$$1 \text{ (only } x) \text{ in } AB - \varphi(\boldsymbol{\phi})\gamma - C\delta :$$

$$\Big(\sum_{i=0}^{n-1} A_{4,i}x^i\Big)\Big(\sum_{i=0}^{n-1} B_{4,i}x^i\Big) + \underline{\Big(\sum_{i=0}^{l} A_{5,i}w_i(x)\Big)\underline{B_2}} + \underline{\Big(\sum_{i=l+1}^{m} A_{6,i}w_i(x) + \sum_{i=0}^{n-2} A_{7,i}x^i t(x)\Big)}B_3$$

$$= \sum_{i=0}^{l} a_i w_i(x) + \sum_{i=l+1}^{m} C_{6,i}w_i(x) + \sum_{i=0}^{n-2} C_{7,i}x^i t(x)$$

Substituting the first two equations into the left hand side of the third one, using that $A_1B_1 = 1$:

$$\Big(\sum_{i=0}^{l} a_i u_i(x) + \sum_{i=l+1}^{m} C_{6,i} u_i(x)\Big)\Big(\sum_{i=0}^{l} a_i v_i(x) + \sum_{i=l+1}^{m} C_{6,i} v_i(x)\Big) = \sum_{i=0}^{l} a_i w_i(x) + \sum_{i=l+1}^{m} C_{6,i} w_i(x) + \sum_{i=0}^{n-2} C_{7,i} x^i t(x)$$

And what we obtain is exactly a QAP statement with $h(x) = \sum_{i=0}^{n-2} C_{7,i} x^i$, hence $\{C_{6,i}\}_{i=l+1}^{m}$ is the assignment of the witness wires. The extractor can thus simply return these values. □

Theorem 3.2. *Assume that $\{u_i(x)\}_{i=0}^{l}$ are linearly independent and $\mathrm{Span}\{u_i(x)\}_{i=0}^{l} \cap \mathrm{Span}\{u_i(x)\}_{i=l+1}^{m} = \emptyset$. Then Groth16 achieves (weak) simulation-extractability against algebraic adversaries under the $(2n-1, n-1)$-**dlog** assumption.*

Proof. Let q denote the number of simulation queries of $\mathcal{A}$, and $\{a_{i,j}\}_{j=0}^{l}$ denote the instance for the ith query. We now add the three proof elements $[\tilde{a}_i]_1, [\tilde{b}_i]_2, [\tilde{c}_i]_1$ revealed in each simulation to the list of elements that $\mathcal{A}$ can use as an algebraic extraction basis: $\tilde{a}_i = \mu_i, \tilde{b}_i = \nu_i$, and $\tilde{c}_i = (\mu_i\nu_i - \alpha\beta - \sum_{j=0}^{l} a_{i,j}(\beta u_j(x) + \alpha v_j(x) + w_j(x)))/\delta$. We write out the representation of A, B, C from the verification equation as the linear combination of the public CRS and these new simulated proof elements (in boxes):

$$A = A_1\alpha + A_2\beta + A_3\delta + \sum_{i=0}^{n-1} A_{4,i} x^i + \sum_{i=0}^{l} A_{5,i} \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} + \sum_{i=l+1}^{m} A_{6,i} \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}$$
$$+ \sum_{i=0}^{n-2} A_{7,i} \frac{x^i t(x)}{\delta} + \boxed{\sum_{i=1}^{q} A_{8,i}\mu_i + \sum_{i=1}^{q} A_{9,i} \frac{\mu_i\nu_i - \alpha\beta - \sum_{j=0}^{l} a_{i,j}(\beta u_j(x) + \alpha v_j(x) + w_j(x))}{\delta}}$$

$$B = B_1\beta + B_2\gamma + B_3\delta + \sum_{i=0}^{n-1} B_{4,i} x^i + \boxed{\sum_{i=1}^{q} B_{5,i}\nu_i}$$

$$C = C_1\alpha + C_2\beta + C_3\delta + \sum_{i=0}^{n-1} C_{4,i} x^i + \sum_{i=0}^{l} C_{5,i} \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} + \sum_{i=l+1}^{m} C_{6,i} \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}$$
$$+ \sum_{i=0}^{n-2} C_{7,i} \frac{x^i t(x)}{\delta} + \boxed{\sum_{i=1}^{q} C_{8,i}\mu_i + \sum_{i=1}^{q} C_{9,i} \frac{\mu_i\nu_i - \alpha\beta - \sum_{j=0}^{l} a_{i,j}(\beta u_j(x) + \alpha v_j(x) + w_j(x))}{\delta}}$$

Our goal is to reduce the theorem to the knowledge-soundness case by restricting the coefficients related to the new simulated proofs variables, namely $A_{8,i}, A_{9,i}, B_{5,i}, C_{8,i}, C_{9,i}$. We will show that a successful $\mathcal{A}$ must either reuse one of the simulated proofs (potentially randomizing it), or it must not have used any simulation-related variables, thus allowing for the reuse of the extraction argument from knowledge soundness. We start by inspecting the coefficients of the following monomials (affect by simulated proofs):

$$\alpha\beta \text{ in } AB - C\delta : A_1B_1 - \sum_{i=1}^{q} A_{9,i}B_3 + \sum_{i=1}^{q} C_{9,i} = 1$$
$$\mu_i\nu_j (i \neq j) \text{ in } AB : A_{8,i}B_{5,j} = 0$$
$$\mu_i\nu_i \text{ in } AB - C\delta : A_{9,i}B_3 + A_{8,i}B_{5,i} - C_{9,i} = 0$$
$$\mu_i\nu_i\nu_j/\delta \text{ in } AB : A_{9,i}B_{5,j} = 0$$
$$\mu_i\nu_i\beta/\delta \text{ in } AB : A_{9,i}B_1 = 0$$
$$\mu_i\beta \text{ in } AB : A_{8,i}B_1 = 0$$
$$\mu_i\gamma \text{ in } AB : A_{8,i}B_2 = 0$$
$$\mu_i\delta \text{ in } AB - C\delta : A_{8,i}B_3 - C_{8,i} = 0$$
$$\nu_i\alpha \text{ in } AB : B_{5,i}A_1 = 0$$
$$\nu_i\beta \text{ in } AB : B_{5,i}A_2 = 0$$
$$\nu_i\delta \text{ in } AB : B_{5,i}A_3 = 0$$

First, we show that all $A_{9,i} = 0$. Assume the contrary: $A_{9,k} \neq 0$ for some k. Then from Equation $(\mu_k\nu_k\nu_j/\delta)$ for all j: $B_{5,j} = 0$. From Equation $(\mu_i\nu_i)$ for all i we have that $C_{9,i} = A_{9,i}B_3$, which, substituted into Equation $(\alpha\beta)$ give us $A_1B_1 = 1$. Hence $B_1 \neq 0$, but from Equation $(\mu_k\nu_k\beta/\delta)$ we see that $A_{9,k}B_1 = 0$, but neither $A_{9,k}$ nor B_1 is zero, a contradiction. Thus, all $A_{9,i} = 0$, and furthermore Equation $(\alpha\beta)$ simplifies to $A_1B_1 + \sum_{i=1}^{q} C_{9,i} = 1$ and Equation $(\mu_i\nu_i)$ simplifies to $A_{8,i}B_{5,i} = C_{9,i}$.

We now show, that if at least one $A_{8,k} \neq 0$, then $\mathcal{A}$ reuses the kth simulatated proof, and otherwise if all $A_{8,i} = 0$ it does not use any simulation-related elements.

- Assume, first, that all $A_{8,i} = 0$: From Equation $(\mu_i\nu_i)$ all $C_{9,i} = 0$. Then, $A_1B_1 = 1$ by Equation $(\alpha\beta)$, so from Equation $(\nu_i\alpha)$ all $B_{5,i} = 0$ (since $A_1 \neq 0$), and from Equation $(\mu_i\delta)$ all $C_{8,i} = 0$ because all $A_{8,i} = 0$.

We now have cancelled all the simulation-related variables, and thus $\mathcal{A}$ does not use simulation queries when constructing its proof, and we can reduce the proof to the knowledge soundness case.

- Assume, otherwise, that at least one $A_{8,k} \neq 0$: Then $B_1 = B_2 = 0$ from Equation $(\mu_k\beta)$ and Equation $(\mu_k\gamma)$. For all $j \neq k$ from Equation $(\mu_k\nu_j)$ we have $B_{5,j} = 0$, and since $C_{9,j} = B_{5,j}A_{8,j}$, all $C_{9,j} = 0$ for $j \neq k$ too. From Equation $(\alpha\beta)$ we obtain $C_{9,k} = 1$, thus $B_{5,k} = 1/A_{8,k}$ by Equation $(\mu_i\nu_i)$. Since now $B_{5,k} \neq 0$, from the Equations $(\nu_k\alpha)$, $(\nu_k\beta)$, $(\nu_k\delta)$ we have $A_1 = A_2 = A_3 = 0$. Thus, we are only left with exactly one nonzero triple $(A_{8,k}, B_{5,k}, C_{9,k})$, which means $\mathcal{A}$ has used at most one simulated proof number k, not being able to combine several simulated proofs into one.

  We next look at additional coefficients related to monomials that include ν_k and ν_k. From Equations $(\nu_i\beta/\delta)$, $(\nu_i\alpha/\delta)$, (ν_i/δ) we have $\sum_{i=l+1}^{m} A_{6,i}(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta + \sum_{i=0}^{n-2} A_{7,i}x^i t(x)/\delta = 0$ (related terms of A are the only terms matching this ν_i in B):

$$\nu_k\beta/\delta \text{ in } AB : \Big(\sum_{j=l+1}^{m} A_{6,j}u_j(x) - \sum_{i=1}^{q} \underline{A_{9,i}} \sum_{j=0}^{l} u_j(x)\Big)B_{5,k} = 0 \implies \sum_{j=l+1}^{m} A_{6,j}u_j(x) = 0$$

$$\nu_k\alpha/\delta \text{ in } AB : \Big(\sum_{j=l+1}^{m} A_{6,j}v_j(x) - \sum_{i=1}^{q} \underline{A_{9,i}} \sum_{j=0}^{l} v_j(x)\Big)B_{5,k} = 0 \implies \sum_{j=l+1}^{m} A_{6,j}v_j(x) = 0$$

$$\nu_k/\delta \text{ in } AB : \Big(\sum_{j=l+1}^{m} A_{6,j}w_j(x) + \sum_{i=0}^{n-2} A_{7,i}x^i t(x) - \sum_{i=1}^{q} \underline{A_{9,i}} \sum_{j=0}^{l} w_j(x)\Big)B_{5,k} = 0$$

$$\implies \sum_{j=l+1}^{m} A_{6,j}w_j(x) = 0 \wedge \sum_{i=0}^{n-2} A_{7,i}x^i t(x) = 0 \text{ (different powers of } x)$$

  Similarly, from Equations $(\nu_i\beta/\gamma)$, $(\nu_i\alpha/\gamma)$, (ν_i/γ) we have $\sum_{i=0}^{l} A_{5,i}(\beta u_i(x)/\gamma) = \sum_{i=0}^{l} A_{5,i}(\alpha v_i(x)/\gamma) = \sum_{i=0}^{l} A_{5,i}(w_i(x)/\gamma) = 0$ (the coefficients are also extracted from AB).

$$\nu_k\beta/\gamma \text{ in } AB : \Big(\sum_{j=0}^{l} A_{5,j}u_j(x)\Big)B_{5,k} = 0 \implies \sum_{j=0}^{l} A_{5,j}u_j(x) = 0$$

$$\nu_k\alpha/\gamma \text{ in } AB : \Big(\sum_{j=0}^{l} A_{5,j}v_j(x)\Big)B_{5,k} = 0 \implies \sum_{j=0}^{l} A_{5,j}v_j(x) = 0$$

$$\nu_k/\gamma \text{ in } AB : \Big(\sum_{j=0}^{m} A_{5,j}w_j(x)\Big)B_{5,k} = 0 \implies \sum_{j=l+1}^{m} A_{5,j}w_j(x) = 0$$

  Because of Equation (ν_k) and Equation (μ_k) we have $\sum_{i=0}^{n-1} A_{4,i}x^i = 0$ and $\sum_{i=0}^{n-1} B_{4,i}x^i = 0$ related terms cancelled as well:

$$\nu_k \text{ in } AB : \Big(\sum_{i=0}^{n-1} A_{4,i}x^i\Big)B_{5,k} = 0 \implies \sum_{i=0}^{n-1} A_{4,i}x^i = 0$$

$$\mu_k \text{ in } AB : \Big(\sum_{i=0}^{n-1} B_{4,i}x^i\Big)A_{8,k} = 0 \implies \sum_{i=0}^{n-1} B_{4,i}x^i = 0$$

  Which also implies $A_{4,i} = B_{4,i} = 0$ for all i. We now focus on the first critical equation, β, which has the same elements as in the KS case, except for the additional $C_{9,i}$. Its left side vanishes completely, and on the right we have exactly one additional simulated instance wires set corresponding to $C_{9,k} = 1$:

$$0 = \sum_{i=0}^{l} a_i u_i(x) + \sum_{i=l+1}^{m} C_{6,i}u_i(x) - \sum_{i=0}^{l} a_{k,i}u_i(x)$$

  Because of disjointness[4] between $u_i(x)$ for witness and instance sets of indices we have both $\sum_{i=0}^{l}(a_i - a_{k,i})u_i(x) = 0$ and $\sum_{i=l+1}^{m} C_{6,i}u_i(x) = 0$, thus also $a_i = a_{k,i}$ because of the linear independence of the first set. Then $\mathcal{A}$ has reused the simulated instance $\boldsymbol{\phi}_k = \{a_{k,i}\}_{i=0}^{l}$, which concludes the proof.

□

$$a = A_1\alpha + A_3\delta + A_1\sum_{i=0}^{m} a_i u_i(x) \qquad b = \frac{1}{A_1}\beta + B_3\delta + \frac{1}{A_1}\sum_{i=0}^{m} a_i v_i(x)$$

$$c = B_3 A + A_3 B - A_3 B_3 \delta + \sum_{i=l+1}^{m} a_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \sum_{i=0}^{n-2} h_i \frac{x^i t(x)}{\delta}$$

Figure 1: The kernel of Groth16 verification equation (a subspace of $Z_p^{9+5n+2m}$) structured as a proof generation routine (the most general one). Note the additional random value A_1, that is not used in the original honest proof generation, but is affected by randomization.

In fact, what Theorem 3.2 proves is stronger than standard weak-SE. Not only can we decide whether the proof π' provided by algebraic $\mathcal{A}$ was a modification of the simulated proof π queried before (or otherwise, if it's not, extract from it), but we can pinpoint which exact simulated proof it was derived from, and what (randomization) transformation T gets π to π'. Even if $\mathcal{A}$ obtains simulated π_1 and π_2 for $\phi \in \mathcal{L}$, and returns π' equal to the randomization of either of the two, we will be able to decide which π_i was used. This does not contradict derivation-privacy of Groth16, that we also prove here, since the derivaction-privacy adversary does not get to see the AGM coefficients. The following corollaries investigate this stronger property, and the randomizability of Groth16 in general. The proofs of these statements are provided in Appendix C.

Corollary 3.2.1. *Let $V(\boldsymbol{C}) = 0$ with $\boldsymbol{C} = (A_1, \ldots, A_{7,n-2}, B_1, \ldots, B_{4,n-1}, C_1, \ldots, C_{7,n-2})$ be the verification equation of Groth16 (Equation (1)) expressed in terms of exponent of $\mathbb{G}_T$ with the $9 + 5n + 2m$ variables serving as linear coefficients that construct the proof from CRS elements, then the kernel[5] of $V(\boldsymbol{C})$ is as presented in Figure 1.*

Corollary 3.2.2. *The only transformation on Groth16 proofs that an algebraic adversary $\mathcal{A}$ can perform is the randomization procedure described in the Equation (2).*

Corollary 3.2.3. *Groth16 NIZK is derivation-private[6] with respect to the randomization transformation presented in Equation (2).*

Acknowledgements

We thank Mary Maller for encouraging discussions and technical feedback on the final version of this note. This work has been supported in part by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 780477 (project PRIViLEDGE).

References

[AB19] Shahla Atapoor and Karim Baghery. "Simulation Extractability in Groth's zk-SNARK". In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2019, pp. 336–354.

[ARS20] Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. "Lift-and-Shift: Obtaining Simulation Extractable Subversion and Updatable SNARKs Generically". In: *ACM SIGSAC Conference on Computer and Communications Security, CCS*. 2020.

[Bag19] Karim Baghery. "On the efficiency of privacy-preserving smart contract systems". In: *International Conference on Cryptology in Africa*. Springer. 2019, pp. 118–136.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. "Zerocash: Decentralized Anonymous Payments from Bitcoin". In: *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, 2014, pp. 459–474. DOI: 10.1109/SP.2014.36. URL: https://doi.org/10.1109/SP.2014.36.

[4] This technique was applied in a similar manner for strong SE in [GM17]

[5] That is, $X \subset \mathbb{Z}_p^{9+5n+2m}$ such that $\forall \boldsymbol{c} \in X. V(\boldsymbol{c}) = 0$

[6] This property has been observed before, for example in [LCKO19] in a similar context.

[BCG+18] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. "Zexe: Enabling Decentralized Private Computation". In: *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 962. URL: https://eprint.iacr.org/2018/962.

[BG18] Sean Bowe and Ariel Gabizon. "Making Groth's zk-SNARK Simulation Extractable in the Random Oracle Model." In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 187.

[BS20] Karim Baghery and Mahdi Sedaghat. "Tiramisu: Black-Box Simulation Extractable NIZKs in the Updatable CRS Model". In: (2020).

[BV98] Dan Boneh and Ramarathnam Venkatesan. "Breaking RSA May Not Be Equivalent to Factoring". In: *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Springer, 1998, pp. 59–71. DOI: 10.1007/BFb0054117. URL: https://doi.org/10.1007/BFb0054117.

[Can01] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. IEEE Computer Society, 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888. URL: https://doi.org/10.1109/SFCS.2001.959888.

[CDD17] Jan Camenisch, Manu Drijvers, and Maria Dubovitskaya. "Practical UC-Secure Delegatable Credentials with Attributes and Their Application to Blockchain". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM, 2017, pp. 683–699. DOI: 10.1145/3133956.3134025. URL: https://doi.org/10.1145/3133956.3134025.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. "Malleable proof systems and applications". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2012, pp. 281–300.

[CKLM13] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. "Succinct Malleable NIZKs and an Application to Compact Shuffles". In: *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*. Ed. by Amit Sahai. Vol. 7785. Lecture Notes in Computer Science. Springer, 2013, pp. 100–119. DOI: 10.1007/978-3-642-36594-2_6. URL: https://doi.org/10.1007/978-3-642-36594-2_6.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. "Universally composable two-party and multi-party secure computation". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 2002, pp. 494–503.

[DFKP16] Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. "Cinderella: Turning Shabby X.509 Certificates into Elegant Anonymous Credentials with the Magic of Verifiable Computation". In: *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 2016, pp. 235–254. DOI: 10.1109/SP.2016.22. URL: https://doi.org/10.1109/SP.2016.22.

[DSDCO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. "Robust non-interactive zero knowledge". In: *Annual International Cryptology Conference*. Springer. 2001, pp. 566–598.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. "The algebraic group model and its applications". In: *Annual International Cryptology Conference*. Springer. 2018, pp. 33–62.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. "On the Non-malleability of the Fiat-Shamir Transform". In: *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*. Ed. by Steven D. Galbraith and Mridul Nandi. Vol. 7668. Lecture Notes in Computer Science. Springer, 2012, pp. 60–79. DOI: 10.1007/978-3-642-34931-7_5. URL: https://doi.org/10.1007/978-3-642-34931-7_5.

[Gam85] Taher El Gamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE Trans. Inf. Theory* 31.4 (1985), pp. 469–472. DOI: 10.1109/TIT.1985.1057074. URL: https://doi.org/10.1109/TIT.1985.1057074.

[GM17] Jens Groth and Mary Maller. "Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs". In: *Annual International Cryptology Conference*. Springer. 2017, pp. 581–612.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Perfect non-interactive zero knowledge for NP". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2006, pp. 339–358.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. "Pairings for cryptographers". In: *Discret. Appl. Math.* 156.16 (2008), pp. 3113–3121. DOI: 10.1016/j.dam.2007.12.010. URL: https://doi.org/10.1016/j.dam.2007.12.010.

[Gro06] Jens Groth. "Simulation-sound NIZK proofs for a practical language and constant size group signatures". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2006, pp. 444–459.

[Gro16] Jens Groth. "On the size of pairing-based non-interactive arguments". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2016, pp. 305–326.

[GS08] Jens Groth and Amit Sahai. "Efficient non-interactive proof systems for bilinear groups". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2008, pp. 415–432.

[KKK20] Thomas Kerber, Aggelos Kiayias, and Markulf Kohlweiss. "Kachina - Foundations of Private Smart Contracts". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 543. URL: https://eprint.iacr.org/2020/543.

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

[KMS+16] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts". In: *2016 IEEE symposium on security and privacy (SP)*. IEEE. 2016, pp. 839–858.

[KZM+15a] Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, Shelat Abhi, and Elaine Shi. "C0C0: a framework for building composable zero-knowledge proofs". In: *Cryptology ePrint Archive, Report 2015/1093* (2015).

[KZM+15b] Ahmed E Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, T-H Hubert Chan, Charalampos Papamanthou, Rafael Pass, Abhi Shelat, and Elaine Shi. "How to Use SNARKs in Universally Composable Protocols." In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 1093.

[LCKO19] Jiwon Lee, Jaekyoung Choi, Jihye Kim, and Hyunok Oh. "SAVER: Snark-friendly, Additively-homomorphic, and Verifiable Encryption and decryption with Rerandomization". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 1270. URL: https://eprint.iacr.org/2019/1270.

[Lip19] Helger Lipmaa. *Simulation-extractable SNARKs revisited*. Tech. rep. Cryptology ePrint Archive, Report 2019/612, 2019. http://eprint. iacr. org . . ., 2019.

[NT16] Assa Naveh and Eran Tromer. "PhotoProof: Cryptographic Image Authentication for Any Set of Permissible Transformations". In: *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 2016, pp. 255–271. DOI: 10.1109/SP.2016.23. URL: https://doi.org/10.1109/SP.2016.23.

[PUB93] NIST FIPS PUB. *Digital signature standard*. 1993.

[PV05] Pascal Paillier and Damien Vergnaud. "Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log". In: *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*. Ed. by Bimal K. Roy. Vol. 3788. Lecture Notes in Computer Science. Springer, 2005, pp. 1–20. DOI: 10.1007/11593447_1. URL: https://doi.org/10.1007/11593447_1.

[Sah99] Amit Sahai. "Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security". In: *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*. IEEE. 1999, pp. 543–553.

[Sch91] Claus-Peter Schnorr. "Efficient Signature Generation by Smart Cards". In: *J. Cryptology* 4.3 (1991), pp. 161–174. DOI: 10.1007/BF00196725. URL: https://doi.org/10.1007/BF00196725.

[Sho04] Victor Shoup. "Sequences of games: a tool for taming complexity in security proofs". In: *IACR Cryptol. ePrint Arch.* 2004 (2004), p. 332. URL: http://eprint.iacr.org/2004/332.

[ZKPce] ZKProof. *ZKProof community reference, version 0.2*. https://docs.zkproof.org/pages/reference/reference.pdf/. [Online; accessed 26/06/2020; Updated versions at https://zkproof.org]. December 2019.

A Weak-SE is Achievable Generically

Proof of Lemma 2.1 *(Sketch).* Groth-Sahai scheme allows to prove statements about satisfiability of (a set of) pairing equations. In this example considering a single equation will be sufficient. We will show that we can embed SNARK verification equation into a GS proof, and SE-transform it, so that the resulting argument is still succinct, randomizable, and derivation-private.

A pairing equation has a form $\prod_i e(x_i, (\Gamma \boldsymbol{y})_i) = 1 \wedge \boldsymbol{x} \sim \boldsymbol{a} \wedge \boldsymbol{y} \sim \boldsymbol{b}$, where $\boldsymbol{x} = \{x_i\}_{i=1}^n, \boldsymbol{y} = \{y_i\}_{i=1}^n$, $x_i \in \mathbb{G}_1, y_i \in \mathbb{G}_2$, $\boldsymbol{a} \in \{\bot, \mathbb{G}_1\}^n, \boldsymbol{b} \in \{\bot, \mathbb{G}_2\}^n$, $\Gamma \in {\mathbb{G}_2}^{n \times n}$ and $\sim$ indicates partial matching (equality on non-$\bot$ elements only). Vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ form the witness, and $(\Gamma, \boldsymbol{a}, \boldsymbol{b})$ represent the instance. This allows us to express verification equation of Groth16 (although any other pairing-based KS SNARK can be used instead). Groth16 proof contains three elements $\pi = (\pi_A, \pi_B, \pi_C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$, and the verification equation is: $e(\pi_A, \pi_B) = e([\alpha]G, [\beta]H)e(f(\phi), [\delta]H)e(C, [\delta]H)$ where $f(\phi) = \sum_{i=0}^{l}[a_i]([(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta]G)$. To embed π into GS we set $\boldsymbol{x} = (\pi_A, [\alpha]G, f(\phi), -\pi_C)$, $\boldsymbol{y} = (\pi_B, [\beta]H, [\gamma]H, [\delta]H)$, $\Gamma = \mathrm{Id}$, $\boldsymbol{a} = (\bot, [\alpha]G, f(\phi), \bot)$, and $\boldsymbol{b} = (\bot, [\beta]H, [\gamma]H, [\delta]H)$. Now the SNARK instance is contained inside the GS instance, and all the three SNARK proof elements are hidden as part of the GS witness.

In order to create a GS proof for a SNARK proof, we need to commit to the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ first. GS commitments are randomizable and derivation-private, meaning that randomized commitments for the particular instance are indistinguishable from fresh commitments to any other corresponding witness. The GS proof itself enjoys the same randomization properties.

GS proof system can be converted to a simulation-extractable one, this conversion uses EUF-CMA signatures in the simulator, and is described in [CKLM12], Theorem 3.3. What this theorem gives is more than we require, since it achieves *controlled-malleable* SE, which is a strict generalisation of SE. This definition is strong in the sense that it is not possible to randomize a proof avoiding transformations altogether, although applying an identity transformation has the very same randomization effect, from the definition perspective. Hence we only include the identity transformation T_{id} into the transformation set $\mathcal{T}$, effectively reducing the language $\mathcal{L}'$ of the transformed NIZK to "$(x, w) \in \mathcal{L} \vee \mathsf{Verify}_{\mathsf{sig}}(\mathsf{vk}, \sigma, x) = 1$", where vk is a verification key of the signature scheme sig, embedded into the CRS. This also collapses cm-SE to weak-SE, while setting $\mathcal{T} = \emptyset$ collapses cm-SE to strong-SE. The idea of the transformation is simple — honest proofs still need to follow the first clause of the disjunction, but the simulator now having access to the trapdoor signing key sk corresponding to vk, can prove simulated statements for the latter clause, producing the valid signatures. The extractor is just a KS extractor: by KS we know that either $(x, w) \in \mathcal{L}$ or the signature in the witness is correct. For honestly generated proofs we reduce to KS, and otherwise for simulated proofs by unforgeability of sig we conclude that $\mathcal{A}$ could not avoid querying the simulator.

After the transformation, the GS extractor can be combined with the KS extractor from the original SNARK to obtain a base NIZK witness for every verifying GS proof, even in the presense of (GS) simulator. At the same time the proof is now randomizable in the derivation-private sense, which is a GS proofs property that the transformation does not affect. Before the transformation the proof size is 8 elements, 4 in each group (which is constant), and commitments are also constant size of 4 group elements each. Modifying the language by adding a structure-preserving signature into the set of pairing equations of incurs a constant size overhead for a chosen security parameter, so the resulting proof, together with the commitments, is still succinct.

We note that this section only motivates the seach for weak-SE SNARKs, and thus is not trying to achieve any optimality. We conjecture that it is possible to obtain slightly simpler generic transformation using Groth16 randomization directly, without embedding it into GS proofs, using recursive controlled malleability techniques from [CKLM13].

□

B Algebraic Verification Satisfiability

This section elaborates on the lemmas of Section 2.3.

Lemma B.1 (Schwartz-Zippel)**.** *Let $f \in \mathbb{F}[X_1, \ldots, X_n]$ be a non-zero polynomial of degree $d \geq 0$ over a field $\mathbb{F}$. Let $S \subset \mathbb{F}$ finite, and let $\boldsymbol{x} = (x_1, \ldots, x_n)$. Then $\Pr_{\boldsymbol{x} \leftarrow S^n}[f(\boldsymbol{x}) = 0] \leq d/|S|$.*

Proof. By recursion on number of variables. □

Proof of Lemma 2.2 *(Sketch).* The intuition for the lemma is that since CRS trapdoors are chosen uniformly, and are "hidden" in the group exponents (hence the discrete log assumption), $\mathcal{A}$ combines $\boldsymbol{e}$ as if it has no knowledge of the internal structure of the CRS, and thus this is equivalent to choosing the V', and then evaluating it on random $\boldsymbol{T}$ (reversed order), which is negligible by S-Z. For the detailed proof of a similar statement tailored specifically for Groth16 in AGM, see [FKL18]. Here we present a sketch of the proof that is slightly more general, and can also be applied to other NILP based SNARKs, e.g. to Groth and Maller SNARK.

The original generic algebraic verification game has the step $[\boldsymbol{e}]_{1,2} \xleftarrow{\$} \mathcal{A}(\boldsymbol{\sigma}); K \leftarrow \mathcal{X}_{\mathcal{A}}^{\mathsf{alg}}(\mathsf{trans}_{\mathcal{A}})$, where K is a matrix of algebraic coefficients. We modify the game, launching $\mathcal{A}$ also on another independently generated CRS and $\boldsymbol{\xi}$ – we can do that since we know K, essentially "how $\boldsymbol{e}$ was constructed from $\boldsymbol{\tau}$", so we just replace the trapdoors and emulate the execution of $\mathcal{A}$. If verification passes on both CRSs, it means that $\mathcal{A}$ constructed its proof $\pi = [\boldsymbol{e}]_{1,2}$ indepedently of the concrete CRS structure, and otherwise he has used it in proof construction.

We split the game in two scenarios according to the result of this test: either (i) $\mathcal{A}$ does not use the *concrete* CRS and returns coefficients blindly (then we arrive at the main positive lemma statement), or (ii) it uses the CRS, thus we break the (d_1, d_2)-**dlog** assumption.

The first option is that $\mathcal{A}$ succeeded without using the concrete CRS $\boldsymbol{\sigma}$ – meaning that it guessed $\boldsymbol{c}_{\iota,i}$ as if it only knew the structure of the CRS (Setup_λ and all $P_{\iota,i}$, but not the concrete $\boldsymbol{\sigma}_i$ themselves). Then the probability for $\mathcal{A}$ to win is low and bounded by S-Z lemma, since the unknown $\boldsymbol{\tau}$ for $\mathcal{A}$ is equivalent to the randomly chosen one – we can generate the concrete CRS after the call to $\mathcal{A}$. By S-Z we know that $\Pr_{\boldsymbol{e}\leftarrow\mathcal{A}(\ldots)}[V(\boldsymbol{e}) = 0 \mid V'(\boldsymbol{T}) \neq 0] < \mathsf{negl}(\lambda)$ where $V'(\boldsymbol{T}) = V\big(K(\mathsf{Setup}_\lambda(\boldsymbol{T}))\big)$, and we also assume that $\Pr[V(\boldsymbol{e}) = 0] = p(\lambda)$ is non-negligible, which means that V can be satisfied by an prover. Then:

$$\Pr[V'(\boldsymbol{T}) \neq 0 \mid V(\boldsymbol{e}) = 0] = \frac{\Pr[V(\boldsymbol{e}) = 0 \mid V'(\boldsymbol{T}) \neq 0]\Pr[V'(\boldsymbol{T}) \neq 0]}{\Pr[V(\boldsymbol{e}) = 0]} = \frac{\mathsf{negl}(\lambda) \cdot \Pr[V'(\boldsymbol{T}) \neq 0]}{p(\lambda)} = \mathsf{negl}(\lambda)$$

So in the end we arrive at the conclusion that $V'(\boldsymbol{T}) = 0$ in case $V(\boldsymbol{e}) = 0$ with high probability.

The other option is that $\mathcal{A}$ has used the CRS nontrivally, possibly extracting knowledge about the trapdoor, which allowed it to satilfy the verification equation. Formally, $\mathcal{A}$ constructed $\boldsymbol{e}$ such that $V'(\boldsymbol{T}) \neq 0$, but $V'(\boldsymbol{\tau}) = V(\boldsymbol{e}) = 0$ for $\boldsymbol{\tau}$ being a concrete trapdoor. Then we can embed (d_1, d_2)-**dlog** instance $([z]_\iota, [z^2]_\iota, \ldots, [z^{d_\iota}]_\iota)$ into the CRS before generation (by using the challenge to generate trapdoors) and solve it. We embed by transforming the challenge into CRS trapdoors $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^n$ in the following way: $[\tau_i]_\iota = [\alpha_i z + \beta_i]_\iota$ for random (α_i, β_i), and then $[\tau_i^j]_\iota = [(\alpha_i z + \beta_i)^j]_\iota$ is a polynomial in z will all known coefficients, so it can be constructed from the q-**dlog** challenge higher powers. Then, after $\mathcal{A}$ returns $\boldsymbol{e}$ that depends on this particular CRS σ with z embedded inside, and satisfies $V(\boldsymbol{e}) = 0$, we factor $V'(\boldsymbol{T})$, reconstructed using K, and reinterpreted as a single variable polynomial over z (since in fact it is parametrised only by one unknown z, and we know all of the other coefficient of this equation except for z), and then one of the roots of this $V'(z)$ will be a solution to the discrete log challenge. □

C Randomizability of Groth16

Proof of Corollary 3.2.1. We start by taking the KS version of the proof elements parametrisation (A, B, C expressed as a linear combination of CRS elements with coefficients containing A_i, B_i and C_i), and applying the constraints we obtained in the KS proof. The malleability constraints we will show are the same for both simulated and real proofs because of indistinguishability of simulated proofs. We apply the reductions from the KS proof, and immediately cancel A_2, B_2, $A_{6,i}$ and $A_{5,i}$ related sums, and the sum with $A_{7,i}$. We also substitute a_i instead of $C_{6,i}$ and $h(x)$ instead of $C_{7,i}$. Since $A_1 B_1 = 1$, we set $B_1 = 1/A_1$.

$$A = A_1\alpha + A_3\delta + \sum_{i=0}^{n-1} A_{4,i}x^i \qquad B = \frac{1}{A_1}\beta + B_3\delta + \sum_{i=0}^{n-1} B_{4,i}x^i$$

$$C = C_1\alpha + C_2\beta + C_3\delta + \sum_{i=0}^{n-1} C_{4,i}x^i + \sum_{i=0}^{l} C_{5,i}\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} +$$
$$+ \sum_{i=l+1}^{m} a_i\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \sum_{i=0}^{n-2} h_i\frac{x^i t(x)}{\delta}$$

In order to restrain $C_{5,i}$ we need to investigate another set of coefficients:

$$\beta\delta/\gamma : \Big(\sum_{i=0}^{l} A_{5,i}u_i(x)\Big)B_3 + \sum_{i=0}^{l} C_{5,i}u_i(x) = 0 \qquad \alpha\delta/\gamma : \Big(\sum_{i=0}^{l} A_{5,i}v_i(x)\Big)B_3 + \sum_{i=0}^{l} C_{5,i}v_i(x) = 0$$
$$\delta/\gamma : \Big(\sum_{i=0}^{l} A_{5,i}w_i(x)\Big)B_3 + \sum_{i=0}^{l} C_{5,i}w_i(x) = 0$$

And as sums with $A_{5,i}$ are zero, we conclude that the relevant sums with $C_{5,i}$ are also zero, so we can exclude them from C. We once again investigate critical equations' coefficients:

$$\beta : \Big(\sum_{i=0}^{n-1} A_{4,i}x^i\Big) = A_1\Big(\sum_{i=0}^{l} a_i u_i(x) + \sum_{i=l+1}^{m} C_{6,i}u_i(x)\Big)$$

$$\alpha : \Big(\sum_{i=0}^{n-1} B_{4,i}x^i\Big) = \frac{1}{A_1}(\sum_{i=0}^{l} a_i v_i(x) + \sum_{i=l+1}^{m} C_{6,i}v_i(x)\Big)$$

We substitute $A_{4,i}$ and $B_{4,i}$ sums into the equation, given that $C_{6,i} = a_i$. What we get is:

$$A = A_1\alpha + A_3\delta + A_1\sum_{i=0}^{m} a_i u_i(x) \qquad B = \frac{1}{A_1}\beta + B_3\delta + \frac{1}{A_1}\sum_{i=0}^{m} a_i v_i(x)$$
$$C = C_1\alpha + C_2\beta + C_3\delta + \sum_{i=0}^{n-1} C_{4,i}x^i + \sum_{i=l+1}^{m} a_i\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \sum_{i=0}^{n-2} h_i\frac{x^i t(x)}{\delta}$$

We now restrain A_3, B_3 ($A_2 = 0$):

$$\delta^2 : A_3B_3 = C_3$$
$$\beta\delta : A_3B_1 + \underline{A_2}B_3 = C_2$$
$$\alpha\delta : A_1B_3 = C_1$$

And express $C_{4,i}$ related sum using $A_{4,i}$ and $B_{4,i}$:

$$\delta : \Big(\sum_{i=0}^{n-1} B_{4,i}x^i\Big)A_3 + \Big(\sum_{i=0}^{n-1} A_{4,i}x^i\Big)B_3 = \sum_{i=0}^{n-1} C_{4,i}x^i$$

The fully reduced system that we obtain now has three free variables (A_1, A_3, B_3), and has the following form:

$$A = A_1\alpha + A_3\delta + A_1\sum_{i=0}^{m} a_i u_i(x) \qquad B = \frac{1}{A_1}\beta + B_3\delta + \frac{1}{A_1}\sum_{i=0}^{m} a_i v_i(x)$$
$$C = A_1B_3\alpha + \frac{A_3}{A_1}\beta + A_3B_3\delta + B_3A_1\sum_{i=0}^{m} a_i u_i(x) + \frac{A_3}{A_1}\sum_{i=0}^{m} a_i v_i(x) + \sum_{i=l+1}^{m} a_i\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \sum_{i=0}^{n-2} h_i\frac{x^i t(x)}{\delta}$$
$$= B_3A + A_3B - A_3B_3\delta + \sum_{i=l+1}^{m} a_i\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \sum_{i=0}^{n-2} h_i\frac{x^i t(x)}{\delta}$$

Since this general form of proof generation satisfies the verification equation (this is easy to verify), no further reductions are possible. Indeed, two out of three free variables are used in the honest generation procedure, and the third one is modified in the randomization transformation. □

Proof of Corollary 3.2.2. Now, in order to obtain the explicit form randomization transformation, we would need to trasform each proof element so that they still fit the bounds we have just presented. Although, this is easier to show if we repeat the process over again, but with the weak-SE proof, now assuming that $\mathcal{A}$ uses one simulated query (weak-SE has shown that no combination of two proofs can be a valid proof). This makes things simpler, because simulated variables μ_i and ν_i stand exactly for already-composed proof elements a and b.

Assume that $A_{8,k} \neq 0$. In the SE proof we already show almost all the coefficient reductions (all A_i except for $A_{8,k}$, all B_i except for B_3 and $B_{5,k}$, $C_{7,i}$, $C_{9,i}$ for $i \neq k$, and $C_{9,k} = 1$). This gives us the following set of equations:

$$A = A_{8,k}\mu_k \qquad B = B_3\delta + B_{5,k}\nu_k$$
$$C = C_1\alpha + C_2\beta + C_3\delta + \sum_{i=0}^{n-1} C_{4,i}x^i + \sum_{i=0}^{l} C_{5,i}\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}$$
$$+ \sum_{i=l+1}^{m} C_{6,i}\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \sum_{i=1}^{q} C_{8,i}\mu_i + \frac{\mu_k\nu_k - \alpha\beta - \sum_{j=0}^{l} a_{k,j}(\beta u_j(x) + \alpha v_j(x) + w_j(x))}{\delta}$$

Further reductions are also easy to discover. From Equation ($\mu_i\delta$), $B_3 = C_{8,k}/A_{8,k}$, and all other $C_{8,i} = 0$. From Equation (δ^2), $C_3 = A_3B_3 = 0$. From Equation ($\alpha\delta$), $C_1 = A_1B_3 = 0$. From Equation ($\beta\delta$), $C_2 = A_3B_1 + A_2B_3 = 0$. We also substitute already obtained $B_{5,k} = 1/A_{8,k}$ from the SE proof:

$$A = A_{8,k}\mu_k$$
$$B = \frac{1}{A_{8,k}}\nu_k + \frac{C_{8,k}}{A_{8,k}}\delta$$

$$C = \sum_{i=0}^{n-1} C_{4,i}x^i + \sum_{i=0}^{l} C_{5,i}\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} + \sum_{i=l+1}^{m} C_{6,i}\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} +$$
$$+ C_{8,k}\mu_k + \frac{\mu_k\nu_k - \alpha\beta - \sum_{j=0}^{l} a_{k,j}(\beta u_j(x) + \alpha v_j(x) + w_j(x))}{\delta}$$

We now need to remove the $C_{4,i}, C_{5,i}, C_{6,i}$ related sums. Nothing can compensate $\sum_{i=0}^{n-1} C_{4,i}x^i$ if we take a look at δ, so it cancels out. Same for $C_{5,i}$ related sum, and monomials $\beta\delta/\gamma$, $\alpha\beta/\gamma$, δ/γ. $C_{6,i}$ also can not be compensated, because of span disjointness of QAP sets for instance and witness, and since verification equation only includes the instance-related sum (the end of the SE proof explains the linear independence technique). What we left with is precisely the well-known randomization transformation, shown in Equation (2), where $r_1 = 1/A_{8,k}$, and $r_2 = C_{8,k}$:

$$A = A_{8,k}\mu_k$$
$$B = \frac{1}{A_{8,k}}\nu_k + \frac{C_{8,k}}{A_{8,k}}\delta$$
$$C = C_{8,k}\mu_k + \frac{\mu_k\nu_k - \alpha\beta - \sum_{j=0}^{l} a_{k,j}(\beta u_j(x) + \alpha v_j(x) + w_j(x))}{\delta}$$

□

Proof of Corollary 3.2.3. In order to prove the statement, we need to show that the distribution of honestly generated proofs $\{\pi\}_\lambda = \{(A, B, C)\}_\lambda$ is the same as the distribution of re-randomized proofs $\{\mathsf{Rand}(\pi)\}_\lambda = \{(A', B', C')\}_\lambda$. In honestly generated proofs, first two values a, b are independently uniform, and the third element of the tuple is defined from them. Indeed, c does not have any other free variables, according to the result of Corollary 3.2.1 (not only the honest generation procedure, but in principle for any algebraic $\mathcal{A}$, which is a property of the verification equation), so fixing a and b fixes c as well.

By examining the randomization equation Equation (2), where we denote randomization values as r_1, r_2, we immediately see that r_1 makes $a' = ar_1$ uniform, and that the same is true for $b' = r_1 b + r_1 r_2 \delta$, since $r_1 r_2 \delta$ is uniform. Thus in both distributions the first two tuple elements are uniform, and the third depends on them in the same way. □

D Monomial Extraction with SageMath

One of the hard parts of the KS and SE proofs is extracting monomial coefficients from the verification equation. Since we are parametrising proof elements of the equation with linear combination of CRS elements (and simulation variables), which consists of many sets of elements known to $\mathcal{A}$, doing extraction manually is quite time-consuming and error-prone. However, the procedure can be partially simplified, or at least verified, by using SageMath. We present a short and simple snippet that defines Groth16 CRS elements and allows then to extract monomial coefficients for any verification equation.

We note that the script fixes QAP sizes and the number of simulation queries, since modelling sums with variable number of elements, to our best knowledge, is much more complicated. We also note that solving KS or SE symbolically using SageMath is a tempting target, but, even though possible for fixed parameters, we found it hard to achieve reasonable performance, thus leaving the idea as an interesting future work direction.

```
# Helper functions

def defvars(label, n):
    return list(var(label + '_%d' % i) for i in range(n))

def defpoly_from_basis(label, basis):
    coeffs = defvars(label,len(basis))
    poly = sum(c*x for c,x in zip(coeffs,basis))
    return (coeffs,poly)

def defpoly(label, d):
    return defpoly_from_basis(label, list(x**i for i in range(d+1)))

# The concrete QAP parameters. q denotes the number of simulated queries.

n = 4
m = 3
l = 1
q = 2
```

```
# Defining Gro16 CRS and the verification equation

var('a_r, b_r, c_r')
trapdoors = var('alpha,beta,gamma,delta,x')
honest_rand = var('r,s')
as_phi = list(var('a_%d' % i) for i in range(l+1))
as_wit = list(var('a_%d' % i) for i in range(l+1, m+1))
as_all = as_phi + as_wit
mu = list(var('mu_%d' % i) for i in range(q))
nu = list(var('nu_%d' % i) for i in range(q))
simphi = list(list(var('sa_%d_%d' % (i,j)) for j in range(l+1)) for i in range(q))
flatsimphi = [item for sublist in simphi for item in sublist]

ringvars = [alpha,beta,gamma,delta,r,s] + mu + nu + [a_r, b_r, c_r] + flatsimphi
R = LaurentPolynomialRing(SR, ringvars)
R.inject_variables()
mu = list(R('mu_%d' % i) for i in range(q))
nu = list(R('nu_%d' % i) for i in range(q))

t_coeffs,t = defpoly('t',n)
h_coeffs,h = defpoly('h',n-2)

us = []; uscoeffs = []
vs = []; vscoeffs = []
ws = []; wscoeffs = []
for i in range(m+1):
    u_coeffs,u = defpoly('u_%d'%i,n-1)
    us.append(u); uscoeffs.append(u_coeffs)
    v_coeffs,v = defpoly('v_%d'%i,n-1)
    vs.append(v); vscoeffs.append(v_coeffs)
    w_coeffs,w = defpoly('w_%d'%i,n-1)
    ws.append(w); wscoeffs.append(w_coeffs)

sigma_1 = [alpha,beta,delta] + \
          list(x**i for i in range(n)) + \
          list((beta*us[i] + alpha*vs[i] + ws[i])/gamma for i in range(l+1)) + \
          list((beta*us[i] + alpha*vs[i] + ws[i])/delta for i in range(l+1, m+1)) + \
          list(x**i * t / delta for i in range(n-1)) + \
          list(mu[i] for i in range(q)) + \
          list((mu[i] * nu[i] - alpha * beta - sum(simphi[i][j] * (beta * us[j] + alpha * vs[j] + ws[j]) for j
               in range(l+1)))/delta for i in range(q))
sigma_2 = [beta,gamma,delta] + list(x**i for i in range(n)) + \
          list(nu[i] for i in range(q))

var('A_1,A_2,A_3')
A4_vars = defvars('A_4', n)
A5_vars = defvars('A_5', l+1)
A6_vars = defvars('A_6', m-l)
A7_vars = defvars('A_7', n-1)
A8_vars = defvars('A_8', q)
A9_vars = defvars('A_9', q)

var('B_1,B_2,B_3')
B4_vars = defvars('B_4', n)
B5_vars = defvars('B_5', q)

var('C_1,C_2,C_3')
C4_vars = defvars('C_4', n)
C5_vars = defvars('C_5', l+1)
C6_vars = defvars('C_6', m-l)
C7_vars = defvars('C_7', n-1)
C8_vars = defvars('C_8', q)
C9_vars = defvars('C_9', q)

A_vars = [A_1,A_2,A_3] + A4_vars + A5_vars + A6_vars + A7_vars + A8_vars + A9_vars
B_vars = [B_1,B_2,B_3] + B4_vars + B5_vars
C_vars = [C_1,C_2,C_3] + C4_vars + C5_vars + C6_vars + C7_vars + C8_vars + C9_vars
ABC_vars = A_vars + B_vars + C_vars

A = sum(c*x for c,x in zip(A_vars,sigma_1))
B = sum(c*x for c,x in zip(B_vars,sigma_2))
C = sum(c*x for c,x in zip(C_vars,sigma_1))

f = sum(as_all[i] * (beta * us[i] + alpha * vs[i] + ws[i]) for i in range(0,l+1))
```

```
V1 = A * B - alpha*beta - f - C * delta

# Printing coefficients

def print_coeff(coeff):
    show(coeff)
    if coeff == 1:
        print(V1.constant_coefficient())
    else:
        print(V1.monomial_coefficient(coeff))
    print("--------------------------")

ks_coeffs = [ alpha * beta, beta ** 2, alpha * gamma, \
            beta * beta / delta, beta * alpha / delta, beta / delta, 1 / delta, \
            beta * beta / gamma, beta * alpha / gamma, beta / gamma, 1 / gamma, \
            beta, alpha, 1]

for coeff in ks_coeffs:
    print_coeff(coeff)

se_coeffs = [ alpha * beta, mu[0] * nu[1], mu[0] * nu[0], \
            mu[0] * nu[0] * nu[1] / delta, mu[0] * nu[0] * beta / delta, \
            mu[0] * beta, mu[0] * gamma, mu[0] * delta, \
            nu[0] * alpha, nu[0] * beta, nu[0] * delta ]

for coeff in se_coeffs:
    print_coeff(coeff)
```