Another Look at Extraction and Randomization of Groth's zk-SNARK

Karim Baghery[1], Markulf Kohlweiss[2,3], Janno Siim[4], and Mikhail Volkhov[3]

[1] imec-COSIC, KU Leuven, Belgium
karim.baghery@kuleuven.be
[2] IOHK
[3] The University of Edinburgh, UK
{mkohlwei, mikhail.volkhov}@ed.ac.uk
[4] University of Tartu, Estonia
janno.siim@ut.ee

Abstract. Due to the simplicity and performance of zk-SNARKs they are widely used in real-world cryptographic protocols, including blockchain and smart contract systems. Simulation Extractability (SE) is a necessary security property for a NIZK argument to achieve Universal Composability (UC), a common requirement for such protocols. Most of the works that investigate SE focus on its strong variant which implies proof non-malleability. In this work we investigate a relaxed weaker notion, that *allows proof randomization*, while guaranteeing statement non-malleability, which we argue to be a more natural security property. First, we show that it is already achievable by Groth16, arguably the most efficient and widely deployed SNARK nowadays. Second, we show that because of this, Groth16 can be *efficiently* transformed into a black-box weakly SE NIZK, which is sufficient for UC protocols.
To support the second claim, we present and compare two practical constructions, both of which strike different performance tradeoffs:

– Int-Groth16 makes use of a known transformation that encrypts the witness inside the SNARK circuit. We instantiate this transformation with an efficient SNARK-friendly encryption scheme.
– Ext-Groth16 is based on the SAVER encryption scheme (Lee et al.) that plugs the encrypted witness directly into the verification equation, externally to the circuit. We prove that Ext-Groth16 is black-box weakly SE and, contrary to Int-Groth16, that its proofs are fully randomizable.

Keywords: zk-SNARKs, Simulation Extractability, UC Security

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+20,SBG+19]. The ready availability of cryptographic libraries implementing SNARKs has also inspired numerous other applications [NT16,DFKP16][1].

Due to its exceptional performance and simplicity, currently the most widely deployed SNARK is Groth16 [Gro16]. In this work, we consider an important perspective on security analysis of Groth16, namely the limits of its malleability (and non-malleability). The lack of study in this direction is surprising considering the importance of non-malleability in distributed settings such as blockchain and the popularity of Groth16 for practical applications.

Arguably, the strongest extraction and non-malleability property for SNARK systems is *simulation-extractability* (SE) [Sah99,DDO$^+$01], a security notion that extends knowledge-soundness (KS) by giving the adversary access to the simulation oracle. One of the important properties of this notion is that its straight-line extractable, black-box variant is necessary to achieve universally composable (UC) security [Can01] for non-interactive zero-knowledge (NIZK) proof systems, as shown by [CLOS02,GOS06,Gro06]. This is an important practical concern since applications employing SNARKs often use the UC framework due to its flexibility and expressive power [KMS$^+$16,KKKZ19,KKK20]. Moreover, SE is needed in game-hopping style proofs [Sho04] in which one game hop introduces the simulator and a subsequent game hop relies on extraction [KMS$^+$16,CDD17].

Simulation-extractability comes in two flavors: the adversary against the stronger flavor is required to produce a proof that differs from any simulated proof that the adversary obtained from the simulator. In this work, we focus on the weaker flavor [KZM$^+$15], that allows for a limited malleability of proofs but requires the adversary to produce a proof for a statement that differs from any of the statements queried from the simulator. Weak SE and strong SE of *proof systems* are in analogy to chosen message attack (CMA) and strong CMA unforgeability of *signatures*.

Another important parameter of a SE notion is whether it supports white-box (WB) or black-box (BB) extraction. A well-known impossibility result [GW11] states that SNARKs cannot be proven secure under falsifiable assumptions. In practice, the non-falsifiability of the assumptions used for SNARKs comes from their white-box nature; that is, they imply some knowledge of the adversary's internals. This prevents proving black-box extraction (and black-box SE), which requires extracting from the adversary only using its "input/output" interface. Since precisely this notion is required for UC security, in practice compilers lifting zk-SNARKs to black-box SE are used [KZM$^+$15,AB19,Bag19], and, crucially, their efficiency can benefit from a stronger (white-box) property of the input SNARK as we show in this work.

Although black-box strong SE is sometimes a desirable property, (black-box) weak SE is sufficient for many UC applications, for instance in Hawk [KMS$^+$16], as argued in [KZM$^+$15]. Hawk uses SE NIZKs directly as a raw primitive (without employing a functionality), and it suggests to use a non-succinct strong SE NIZK, since no other candidates were known at that time. Kosba et al. [KZM$^+$15] point out that a weak SE NIZK can be used instead. We also note that weak SE is

[1] See also the application chapter of [ZKP19].

sufficient for the SNARKs to signatures of knowledge (SoK) compiler of [GM17] that embeds a hash of the message into the statement proven. Thus applications employing SoK, such as [BMRS20], can also benefit from our work. Note that in weak SE it is the statement rather than the proof that cannot be mauled. The resulting SoK satisfies CMA unforgeability.

Our contributions. Our results are twofold. First, we show that Groth16, as described in the literature and deployed in practical applications, is already white-box weak SE.

Surprisingly, this was not known before. Proof malleability was noted by [GM17] as an obstacle for proving the strong SE property for Groth16, which resulted in them constructing a new non-malleable SNARK. Allowing proof randomization in the definition resolves the issue differently by proving a security property for the original system that lies in strength between knowledge soundness and strong SE. Additionally, we show that only a specific type of proof malleability is possible and that rerandomized proofs have the same distribution as fresh proofs of the same statement. We show in the algebraic group model (that we state as an assumption) that the extractor can either obtain the witness or point to the unique simulated proof that was randomized to obtain the proof produced by the adversary. Thus, even if the adversary queries multiple proofs for the same statement, it cannot combine them into a new proof of the same statement, which is the main technical challenge in proving white-box weak SE.

As our second contribution, we give two optimized constructions for black-box weak SE: Int-Groth16 and Ext-Groth16. Int-Groth16 is based on the (strong) WB-to-BB SE compiler of [Bag19]. It adds a public key of a cryptosystem to the CRS and a ciphertext containing encryption of the witness to the proof. It then employs a SNARK to prove an extended statement to ensure that the witness is correctly encrypted. We show that this compiler can be used for *weak* WB-to-BB conversion, and therefore instantiated with the more efficient Groth16.[2] We optimize the encryption scheme and employ a SNARK-friendly variant of ElGamal with randomness reuse [Kur02]. A noteworthy technical detail is that the witness needs to be mapped to SNARK-friendly elliptic curve points. The downside of this construction is that even state-of-the-art SNARK-friendly public-key operations incur a substantial overhead in the circuit size.

Ext-Groth16 uses a verifiable encryption technique of Lee et al. [LCKO19] to overcome this limitation. We again encrypt the witness, but with a different encryption scheme in which resulting ciphertexts enter Groth16 verification equation directly and thus have almost no effect on the circuit structure. To show Ext-Groth16 secure, we need to directly prove black-box weak simulation-extractability, which we do by a reduction to white-box weak SE of Groth16. The main technical challenge is, again, to show which transformations exactly are available to the adversary. Additionally, we prove that the zero-knowledge

[2] In fact, even weak simulation soundness without extractability is sufficient for the compiler.

property of Ext-Groth16 can rely on the standard Decisional Diffie-Hellman assumption rather than the novel assumption stated in [LCKO19].

To compare the efficiency of these two constructions, we estimate CRS and proof size, prover time, and verifier time as a function of the encrypted witness size. Our results show that both constructions have low overhead compared to the commonly used generic transformations. In particular, Ext-Groth16 leads to almost no increase in CRS size and prover time, while resulting in slightly bigger proofs and verification time.

Related Work. Simulation-extractability is relevant for both CRS-based and Random-Oracle (RO) based NIZKs. Faust et al. [FKMV12] show that NIZKs obtained from Σ-protocols using the Fiat-Shamir heuristic satisfy simulation-extractability in the RO model. In this work we focus on simulation-extractability of CRS-based NIZKs, and on the Groth16 SNARK in particular.

White-box constructions. White-box SE SNARKs have been discovered only recently. Groth and Maller [GM17] presented the first construction in 2017, targeting the language of Square Arithmetic Programs (SAPs). They also proved a lower bound of three group elements for the proof size and two verification equations for all *non-interactive linear proof* (NILP) based SNARKs, which covers many previously known pairing-based SNARKs. Weak SE allows us to go below this bound with a single verification equation.

Bowe and Gabizon [BG18] give a RO-based variant of Groth16 for Quadratic Arithmetic Programs (QAPs) that is simulation-extractable, and has five group elements and two verification equations. Lipmaa [Lip19] presents a different technique that allows to construct SE SNARKs for QAP and the three other arithmetization techniques from the QAP family (namely, SAP, SSP, and QSP). Kim, Lee, and Oh [KLO19] present a SE SNARK for QAP with three elements but just a single verification equation, avoiding the lower bound of Groth and Maller by using a RO in addition to a knowledge extraction assumptions and a CRS. Recently, Baghery, Pindado, and Ràfols [BPR20] revised Bowe and Gabizon's construction [BG18] and presented a new variation which saves 1 paring in the verification, and gets rid of the RO at the cost of a collision-resistant hash function.

Black-box transformations. A generic transformation that makes ordinary NIZKs black-box SE has been known at least since [DDO$^+$01]. Along this direction, Kosba et al. [KZM$^+$15] extend, analyse, and optimize this transformation technique — they present three transformations; two of which build weak SE NIZKs, while the third builds a strong SE NIZKs. Atapoor and Baghery [AB19] adapt Kosba et al.'s work directly to Groth16 and evaluate the efficiency of the resulting strong SE argument. Baghery [Bag19] analyses a transformation from white-box SE to black-box SE, and instantiates it with the strong SE SNARK by Groth and Maller. We show that this technique also works for lifting white-box *weak* SE to black-box *weak* SE. Other generic transformations take into account CRS subversion and updatability [ARS20,BS20].

2 Preliminaries

Notation. We denote the security parameter by $\lambda \in \mathbb{N}$. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible, if for a big enough λ, $f < 1/p(\lambda)$ for all polynomials $p(\lambda)$. We write $g(\lambda) = \mathrm{negl}(\lambda)$ to mean that g is some negligible function. For a distribution X we denote random sampling by $x \overset{\$}{\leftarrow} X$, and when this notation is used with a finite set S, $x \overset{\$}{\leftarrow} S$ denotes uniform sampling 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 $X, Y, \varDelta, X_\gamma$, and lower case for concrete values x, y, δ, γ. We use vector notation to denote a list of formal variables, so for $\boldsymbol{X} = X_1, \dots, X_n$, we write $P(\boldsymbol{X}) \in \mathbb{F}[X_1 \dots X_n] = \mathbb{F}[\boldsymbol{X}]$ for a polynomial in these variables, and for a $\boldsymbol{x} \in \mathbb{F}^n$, $P(\boldsymbol{x})$ will denote the polynomial evaluation $P(x_1 \dots x_n)$.

PPT stands for (uniform) probabilistic polynomial-time. An *execution transcript* $\mathsf{trans}_{\mathcal{P}}$ of an algorithm $\mathcal{P}$ contains $\mathcal{P}$'s private coins, inputs and outputs, including interactions with any oracles that it is provided with. Having access to $\mathsf{trans}_{\mathcal{P}}$ implies white-box access to $\mathcal{P}$.

Bilinear groups. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot), p)$ be a Type III[3] bilinear group of prime order p with generators G, H, and $e(G, H)$ for the three groups respectively. The pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map. We will write $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ additively. 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 $[a]_1 \bullet [b]_2 \triangleq e([a]G, [b]H)$. 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 $[\{a, b, c, \dots\}]_\iota = \{[a]_\iota, [b]_\iota, [c]_\iota, \dots\}$ 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.

Circuit Form and Quadratic Arithmetic Programs (QAP). Let $\mathcal{R}$ be a relation for an NP language $\mathcal{L}$, such that $(\phi, w) \in \mathcal{R} \Leftrightarrow \phi \in \mathcal{L}$. When $\mathcal{R}$ is implemented as an arithmetic circuit $\mathcal{C}$, we assume it to be of the following form. The input wires are split into: l public input wires corresponding to $\phi_1, \dots, \phi_l$, and l_w private input wires, corresponding to $w_1, \dots, w_{l_w}$. We denote the total number of wires by m, and thus the remaining $m - l - l_w$ wires are called intermediate — they can be computed from ϕ and w.

A quadratic arithmetic program (QAP, [GGPR13]) for the circuit $\mathcal{C}$ consists of the quotient polynomial $t(x)$ of degree n, and three sets of polynomials

[3] 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 [GPS06].

$\{u_i(X)\}_{i=0}^m, \{v_i(X)\}_{i=0}^m$ and $\{w_i(X)\}_{i=0}^m$ of degree $n-1$. A particular QAP assignment $\{a_i\}_{i=0}^m$ contains assignments to the circuit wires, and $a_0 = 1$ is a fixed parameter. We will refer to the sets $\{\phi_i\} \cup \{w_i\}$ and $\{a_i\}$ interchangeably when there is no risk of confusion, with ϕ_0 corresponding to a_0. The assignment $\{a_i\}$ 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.

As QAP relations are defined over a finite field that determines suitable bilinear groups, they need to be compatible with the desired security level λ. Our asymptotic security notions are all quantified over λ-compatible relations $\mathcal{R}_\lambda$. In practice SNARK systems use very specific pre-defined groups for a fixed security level. For these reasons we elide most of these details in our formal modelling and typically write $\mathcal{R}$ instead of $\mathcal{R}_\lambda$.

Algebraic Modelling and Assumptions. Following [FKL18,Lip19], we say that the algorithm $\mathcal{A}$ is *algebraic*, if there is a way to represent any group element it returns using elements it has seen before, specifically as a linear combination of these elements with known (extracted) coefficients. Security against algebraic adversaries can be formalized either as a standard model white-box knowledge-extraction assumption [BV98,PV05,Lip19], or by defining a separate cryptograpic model as done in the algebraic group model (AGM) [FKL18]. 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 1 (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 $(\mathbb{Z}_p^*)^m$,*

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

It is easy to see how this definition extends to the asymmetric bilinear groups ($\mathcal{X}_{\mathcal{A}}^{\mathsf{alg}}$ 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). That means that in the soundness and knowledge soundness games, an algebraic adversary $\mathcal{A}$ gets only CRS elements as an input, and in the simulation-based definitions $\mathcal{A}$ additionally sees the simulated proof elements.

In proofs with algebraic adversaries, we use the following variant of the discrete logarithm assumption [FKL18].

Definition 2 ((q_1, q_2)-Discrete Logarithm Assumption). *Let $(\mathbb{G}_1, \mathbb{G}_2, \cdot, \cdot, p)$ be a Type III bilinear group. We say that (q_1, q_2)-**dlog** holds if for all PPT $\mathcal{A}$,*

$$\Pr\left[x \xleftarrow{\$} \mathbb{Z}_p^*; z \xleftarrow{\$} \mathcal{A}([x, \dots, x^{q_1}]_1, [x, \dots, x^{q_2}]_2) : x = z\right] = \mathrm{negl}(\lambda).$$

In Appendix B, we prove a lemma which intuitively shows that in a typical SNARK soundness proofs against algebraic adversaries, one can view the verification equation as a polynomial equality test where trapdoors are substituted by indeterminates. We believe that this lemma might be of independent interest.

Non-interactive Zero-knowledge Arguments. We introduce security notions for non-interactive zero-knowledge (NIZK) arguments that we use throughout the paper. In particular, we define proof rerandomization and different flavors of simulation-extractability. In the following, NIZK denotes a tuple of efficient algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ unless specified otherwise.

Weak simulation extractability (SE) is an extension of knowledge soundness where adversary can query simulated proofs (even for false statements) and finally has to come up with a statement and a proof for which an extractor cannot recover a witness. Moreover, the statement cannot be any of the statements queried from the oracle. First, we give a definition for white-box version which allows there to be a different extractor for each adversary.

Definition 3 (White-box Weak Simulation-Extractability, [KZM+15]). *We say that* NIZK *is* white-box weak SE *if for any PPT adversary $\mathcal{A}$ there exists a polynomial time extractor $\mathcal{X}_{\mathcal{A}}$ such that for $\mathcal{R}_\lambda$,*

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

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.*

The important distinction between this notion and *strong* SE lies in the last condition in the security game. Strong SE requires $(\phi, \pi) \notin Q$, where $\mathcal{S}$ records pairs of queried instances and simulated proofs. If NIZK is randomizable, $\mathcal{A}$ can just pass re-randomized simulated proof for an instance it does not know a witness of 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.

The black-box variant of weak SE specifies the existence of a single extractor that works for all adversaries.

Definition 4 (Black-box Weak Simulation-Extractability, [KZM+15]). *We say that* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim}, \mathsf{Ext})$ *is* black-box weak SE *if for any PPT adversary $\mathcal{A}$ and $\mathcal{R}_\lambda$,*

$$\Pr\left[\begin{matrix}(\boldsymbol{\sigma}, \tau, \tau_{ext}) \leftarrow \mathsf{Setup}(\mathcal{R}_\lambda); \\ (\phi, \pi) \leftarrow \mathcal{A}^{\mathcal{S}_{\boldsymbol{\sigma},\tau}}(\boldsymbol{\sigma}); w \leftarrow \mathsf{Ext}(\boldsymbol{\sigma}, \tau_{ext}, \phi, \pi)\end{matrix} : \begin{matrix}\mathsf{Verify}(\boldsymbol{\sigma}, \phi, \pi) = 1 \wedge \\ (\phi, w) \notin \mathcal{R}_\lambda \wedge \phi \notin Q\end{matrix}\right] = \mathrm{negl}(\lambda),$$

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.*

Proof malleability can also be a beneficial security property. We call the proof system for the relation $\mathcal{R}$ *randomizable* or *proof malleable*, if there exists a (non-trivial) PPT procedure Rand such that $\Pr[\mathsf{Verify}(\boldsymbol{\sigma}, \phi, \mathsf{Rand}(\pi))] = 1$ for all honestly generated proofs π for $\boldsymbol{\sigma}$ and ϕ. The notion of proof rerandomization we use is similar to [BCC+09] and the ciphertext rerandomization in [LCKO19]:

Definition 5 (Proof Rerandomization). *A proof system is* rerandomizable *with respect to relation $\mathcal{R}_\lambda$ and randomization transformation* Rand*, if for all $(\phi, w) \in \mathcal{R}_\lambda$, all $\boldsymbol{\sigma}$ output by $\mathsf{Setup}(\mathcal{R}_\lambda)$ and all π such that $\mathsf{Verify}(\boldsymbol{\sigma}, \phi, \pi) = 1$: $\{\mathsf{Prove}(\boldsymbol{\sigma}, \phi, w)\}_\lambda = \{\mathsf{Rand}(\boldsymbol{\sigma}, \phi, \pi)\}_\lambda$, where the randomness is over the random variables used in* Prove *and* Rand.

The standard definitions of knowledge soundness (KS), zero-knowledge, and a weaker simulation-soundness notion that is only used by our compiler in Section 4.1 (it is obtained by removing the extractor from SE, similarly to the distinction between KS and soundness) are deferred to Appendix A.

3 White-box Weak SE and Randomizability of Groth16

In this section, we show that Groth16 is white-box weakly simulation extractable, which to our knowledge is the first SNARK construction that is proved to (only) achieve this notion. Additionally, we provide some facts about randomization of Groth16. We start by recalling Groth16 in Fig. 1.

White-box weak SE. Our proof is in the AGM and relies on the same hardness assumptions ((q_1, q_2)-discrete logarithm) as Groth16 knowledge soundness. Additionally we require a form of linear independence from QAP polynomials — a similar requirement was used for square arithmetic programs in [GM17].

Theorem 1. *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 white-box SE against algebraic adversaries under the $(2n-1, n-1)$-**dlog** assumption.*

Proof (Sketch). The proof splits in two branches — we show that either $\mathcal{A}$ uses simulated elements, and in this case it can only use them for a single simulation query k, or it does not use them at all. In particular, this implies that $\mathcal{A}$ cannot combine several elements from different queries algebraically for the π it submits. We then argue that the non-simulation case reduces to knowledge soundness, and in the simulation case we show that $\mathcal{A}$ supplies ϕ that is equal to one of the simulated instances, which proves that $\mathcal{A}$ reuses a simulated proof, potentially randomized. An interesting detail not captured in the weak SE definition is that not only can we decide whether the proof π' provided by algebraic $\mathcal{A}$ is a modification of the simulated proof π queried before in the simulation case, but we can pinpoint which exact simulated proof it was derived from. For the full proof, see Appendix C □

To simplify notation we denote $q_i(\alpha,\beta,x) = \beta u_i(x) + \alpha v_i(x) + w_i(x)$ and $y_i(\alpha,\beta,\gamma,x) = q_i(\alpha,\beta,x)/\gamma$, and use it as $q_i(x)$ and $y_i(x)$ omitting other variables when it is clear from the context. As explained before, $\boldsymbol{a} = \boldsymbol{\phi} \parallel \boldsymbol{w}$.

```
Setup(R):
    $\boldsymbol{\tau} = x,\alpha,\beta,\gamma,\delta \xleftarrow{\$} \mathbb{Z}_p^*$
    $\boldsymbol{\sigma}_1 \leftarrow \left[\alpha,\beta,\delta,\{x^i\}_{i=0}^{n-1}, \left\{\frac{x^i t(x)}{\delta}\right\}_{i=0}^{n-2}, \{y_i(x)\}_{i=0}^{l}, \left\{\frac{q_i(x)}{\delta}\right\}_{i=l+1}^{m}\right]_1$
    $\boldsymbol{\sigma}_2 \leftarrow \left[\beta,\gamma,\delta,\{x^i\}_{i=0}^{n-1}\right]_2$
    return $(\boldsymbol{\sigma} = \boldsymbol{\sigma}_1 \cup \boldsymbol{\sigma}_2, \boldsymbol{\tau})$
Prove($\boldsymbol{\sigma}, \boldsymbol{\phi} = \phi_1 \dots \phi_l, \boldsymbol{w} = w_1 \dots w_{m-l}$):
    $r_a, r_b \xleftarrow{\$} \mathbb{Z}_p^*$
    $[a]_1 \leftarrow \left[\alpha + \sum_{i=0}^m a_i u_i(x) + r_a\delta\right]_1$;  $[b]_2 \leftarrow \left[\beta + \sum_{i=0}^m a_i v_i(x) + r_b\delta\right]_2$
    $[c]_1 \leftarrow \left[\sum_{i=l+1}^m a_i \frac{q_i(x)}{\delta} + \frac{h(x)t(x)}{\delta} + a r_b + b r_a - r_a r_b \delta\right]_1$
    return $([a]_1, [b]_2, [c]_1)$
Verify($\boldsymbol{\sigma}, \boldsymbol{\phi} = \phi_1 \dots \phi_l, \pi = (a,b,c)$):
    assert $e(a,b) = e([\alpha]_1,[\beta]_2) + e(\sum_{i=0}^l \phi_i [y_i(x)]_1, [\gamma]_2) + e(c,[\delta]_2)$
Sim($\boldsymbol{\tau}, \boldsymbol{\phi} = \phi_1 \dots \phi_l$):
    $\mu,\nu \xleftarrow{\$} \mathbb{Z}_p^*$; return $\left([\mu]_1, [\nu]_2, \left[\frac{\mu\nu - \alpha\beta - \sum_{i=0}^l \phi_i q_i(x)}{\delta}\right]_1\right)$
Rand($\boldsymbol{\sigma}, \pi = (a,b,c)$):
    $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$;  $a \mapsto (1/r_1)a$;  $b \mapsto r_1 b + r_1 r_2 [\delta]_2$;  $c \mapsto c + r_2 a$
    return $(a,b,c)$
```

Fig. 1. Groth16 zk-SNARK with simulation and randomization procedures.

Transforming the Proof. It is known that Groth16 has malleable proofs. It is not hard to extend this statement to show that Groth16 is rerandomizable, that is its output of Rand is indistinguishable from honest proofs, even if Rand is applied to maliciously generated (but verifiable) proofs.

Theorem 2. *Groth16 zk-SNARK is rerandomizable*[4] *with respect to the randomization transformation* Rand *presented in Fig. 1.*

Proof. In a nutshell, the proof elements a and b output by Rand are random and independent of each other; and the verification equation fixes a unique c based on $a, b, \boldsymbol{\sigma}, \boldsymbol{\phi}$. A more detailed proof is provided in Appendix D. □

Together with white-box weak SE forbidding instance malleability, and perfect ZK, Theorem 2 implies that randomization is equivalent to any other way to transform the honest (or simulated) proofs. But this does not give an explicit algebraic characterization of the transformation — that is, we do not know if

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

there is any other way to create an honest proof, or any other way to rerandomize it (that would produce the same distribution). One of the interesting properties of the proof of Theorem 1 is that it can be extended to show that Rand is the only algebraic transformation possible, which we present as an independent result. We also show that the most-general algebraic form of the honest generation procedure has at most three random "axes", any two of which are required for perfect zero-knowledge.

Observation 1. The only form of algebraic transformation on $\mathsf{Groth16}$ proofs that is possible without violating its verification equation is the randomization procedure $\mathsf{Rand}(\boldsymbol{\sigma}, \pi = (a, b, c); r_1, r_2)$, where r_1, r_2 are chosen by the adversary.

4 Black-box Weak SE

We study two approaches to achieve black-box weak SE by encrypting the witness. The first construction $\mathsf{Int\text{-}Groth16}$ integrates ciphertexts directly to the relation, and the second construction $\mathsf{Ext\text{-}Groth16}$ proves the correctness of ciphertexts with external techniques.

4.1 Black-box Weak SE with Internal Encryption

First, we describe a generic transformation for achieving black-box weak SE. We let the prover encrypt the witness w with a IND-CPA secure cryptosystem and then use a weak simulation sound NIZK (e.g., $\mathsf{Groth16}$) to prove the relation

$$\mathcal{R}' \triangleq \{((\phi, \mathsf{pk}, \boldsymbol{c}), (w, r)) : (\phi, w) \in \mathcal{R} \land \boldsymbol{c} = \mathsf{Enc}(\mathsf{pk}, w; r)\},$$

where ϕ is the statement the prover wants to prove and $\mathcal{R}$ is the corresponding relation. Since we make the public key pk a part of the reference string, it will be possible to black-box extract the witness from the ciphertext. Full details of the construction can be seen in Fig. 2.

This transformation was first analyzed in [Bag19], where it was shown to lift a white-box *strong* SE NIZK to a black-box *strong* SE. Below we sketch a proof that it also lifts a weak simulation sound NIZK to a black-box weak SE NIZK.

Theorem 3. *Let* $\mathsf{NIZK}' = (\mathsf{Setup}', \mathsf{Prove}', \mathsf{Verify}', \mathsf{Sim}')$ *be a complete, weak simulation sound, and computational zero-knowledge non-interactive proof system and* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *an IND-CPA secure cryptosystem. Then the* NIZK *construction in Fig. 2 is complete, black-box weak SE, and computational zero-knowledge.*

Proof (sketch). Completeness of NIZK follows from the completeness of NIZK' and correctness of the cryptosystem. Computational zero-knowledge holds since $\mathsf{Enc}(\mathsf{pk}, 0)$ is computationally indistinguishable from $\mathsf{Enc}(\mathsf{pk}, w)$ and since NIZK' already has computational zero-knowledge. Finally, suppose that there exists a

```
$\underline{\mathsf{Setup}(\mathcal{R}_\lambda)}$: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$; $(\boldsymbol{\sigma}', \boldsymbol{\tau}') \leftarrow \mathsf{Setup}'(\mathcal{R}')$;
    $\mathbf{return}\ (\boldsymbol{\sigma} = \boldsymbol{\sigma}' \cup \mathsf{pk}, \boldsymbol{\tau} = \boldsymbol{\tau}', \boldsymbol{\tau}_{ext} = \mathsf{sk})$;
$\underline{\mathsf{Prove}(\boldsymbol{\sigma} = \boldsymbol{\sigma}' \cup \mathsf{pk}, \phi, w)}$:
    $r \xleftarrow{\$} \mathbb{Z}_p^*$, $\boldsymbol{c} \leftarrow \mathsf{Enc}(\mathsf{pk}, w; r)$; $\pi' = \mathsf{Prove}'(\boldsymbol{\sigma}, (\phi, \mathsf{pk}, \boldsymbol{c}), (w, r))$ $\mathbf{return}\ (\boldsymbol{c}, \pi')$;
$\underline{\mathsf{Verify}(\boldsymbol{\sigma} = \boldsymbol{\sigma}' \cup \mathsf{pk}, \phi, \pi = (\boldsymbol{c}, \pi'))}$: $\mathbf{assert}\ \mathsf{Verify}'(\boldsymbol{\sigma}, (\phi, \mathsf{pk}, \boldsymbol{c}), \pi')$;
$\underline{\mathsf{Sim}(\boldsymbol{\sigma} = \boldsymbol{\sigma}' \cup \mathsf{pk}, \boldsymbol{\tau}, \phi)}$:
    $\boldsymbol{c} \leftarrow \mathsf{Enc}(\mathsf{pk}, 0; r)$ for $r \xleftarrow{\$} \mathbb{Z}_p^*$; $\pi' \leftarrow \mathsf{Sim}'(\boldsymbol{\sigma}', \boldsymbol{\tau}, (\phi, \mathsf{pk}, \boldsymbol{c}))$; $\mathbf{return}\ (\boldsymbol{c}, \phi)$;
$\underline{\mathsf{Ext}(\boldsymbol{\sigma}, \boldsymbol{\tau}_{ext}, \phi, \pi = (\boldsymbol{c}, \pi'))}$: $\mathbf{return}\ \mathsf{Dec}(\boldsymbol{\tau}_{ext}, \boldsymbol{c})$;
```

Fig. 2. The construction for black-box weak SE NIZK where $\mathsf{NIZK}' = (\mathsf{Setup}', \mathsf{Prove}', \mathsf{Verify}', \mathsf{Sim}')$ is a weak simulation sound NIZK and $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a IND-CPA secure cryptosystem.

PPT adversary $\mathcal{A}$ that can break black-box weak SE of NIZK. We can easily construct a PPT adversary $\mathcal{B}$ that can break weak simulation soundness of NIZK'. $\mathcal{B}$ gets $\boldsymbol{\sigma}'$ as an input and generates pk itself. Now $\mathcal{B}$ can run $\mathcal{A}(\boldsymbol{\sigma}' \cup \mathsf{pk})$ internally and whenever $\mathcal{A}$ makes a simulation query ϕ, $\mathcal{B}$ makes a simulation query $(\phi, \mathsf{pk}, \boldsymbol{c} = \mathsf{Enc}(\mathsf{pk}, 0))$ and gets back a proof π' which allows him to send $(\boldsymbol{c}, \pi')$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs $(\phi^*, (\boldsymbol{c}^*, \pi^*))$ such that ϕ^* has not been queried and either ϕ^* is an invalid statement or $\boldsymbol{c}$ does not encrypt the correct witness. Now $\mathcal{B}$ can output $((\phi^*, \boldsymbol{c}^*), \pi^*)$ which will break weak simulation soundness. $\square$

We can obtain good efficiency if we instantiate the above construction by taking $\mathsf{Groth16}$ as NIZK' and by using vector ElGamal as a cryptosystem (see Appendix F for details). We call this instantiation $\mathsf{Int\text{-}Groth16}$. In Section 5 we discuss further optimization of this construction.

Corollary 1. *$\mathsf{Int\text{-}Groth16}$ is a complete, black-box weak SE, and computational zero-knowledge NIZK argument.*

4.2 Black-box Weak SE with External Encryption

The disadvantage of the previous construction is that one needs to encode the extended relation as an arithmetic circuit, that is shown, e.g. in Hawk, to result in a considerably larger public parameters and a slower prover. Thus, we propose a second construction $\mathsf{Ext\text{-}Groth16}$ which is closely based on the SAVER cryptosystem [LCKO19] which in a sense gives ciphertexts as a public input to $\mathsf{Groth16}$. Having the encryption outside of the circuit allows us to have smaller circuit overhead which results in smaller CRS size and higher prover efficiency. As before, proof size is linear, and is dominated by the size of the encrypted witness (this is inevitable for black-box constructions, as discussed before [GW11]). The formal description is presented on Fig. 3. Roughly speaking, we reinstantiate SAVER, but also prove that the construction is black-box weak simulation extractable. Additionally we re-prove computational zero-knowledge under the weaker and more standard DDH assumption.

```
$\underline{\mathsf{Setup}(\mathcal{R}):}$
    $\boldsymbol{\tau}_1 = x, \alpha, \beta, \gamma, \delta \xleftarrow{\$} \mathbb{Z}_p^*$; $\boldsymbol{\tau}_2 = \{s_i\}_{i=1}^{l_w}, \{t_i\}_{i=0}^{l_w} \xleftarrow{\$} \mathbb{Z}_p^*$;
    $\boldsymbol{\sigma}_1 \leftarrow \left[\alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \left\{\frac{x^i t(x)}{\delta}\right\}_{i=0}^{n-2}, \{y_i(x)\}_{i=0}^{l+l_w}, \left\{\frac{q_i(x)}{\delta}\right\}_{i=l+l_w+1}^{m}\right]_1$;
    $\boldsymbol{\sigma}_2 \leftarrow \left[\beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}\right]_2$;
    $\mathsf{pk}_1 \leftarrow \left[\{\delta s_i\}_{i=1}^{l_w}, \{y_{l+i}(x) t_i\}_{i=1}^{l_w}, \delta(t_0 + \sum_{i=1}^{l_w} t_i s_i), \gamma(1 + \sum_{i=1}^{l_w} s_i)\right]_1$;
    $\mathsf{pk}_2 \leftarrow \left[\{t_i\}_{i=0}^{l_w}\right]_2$;
    $\textbf{return } (\boldsymbol{\sigma} = \boldsymbol{\sigma}_1 \cup \boldsymbol{\sigma}_2 \cup \mathsf{pk}_1 \cup \mathsf{pk}_2, \boldsymbol{\tau} = \boldsymbol{\tau}_1 \cup \boldsymbol{\tau}_2, \boldsymbol{\tau}_{\text{ext}} = \{s_i\}_{i=1}^{l_w})$;
$\underline{\mathsf{Prove}(\boldsymbol{\sigma}, \boldsymbol{\phi} = \phi_1 \dots \phi_l, \boldsymbol{w} = w_1 \dots w_{l_w} \dots w_{m-l}):}$
    $r, r_a, r_b \xleftarrow{\$} \mathbb{Z}_p^*$;
    $c_0 \leftarrow r[\delta]_1$; $c_i \leftarrow r[\delta s_i]_1 + w_i[y_{l+i}(x)]_1$ for $i \in [1 \dots l_w]$;
    $\psi \leftarrow r\left[\delta(t_0 + \sum_{j=1}^{l_w} t_j s_j)\right]_1 + \sum_{i=1}^{l_w} w_i [y_{l+i}(x) t_i]_1$;
    $[a]_1 \leftarrow \left[\alpha + \sum_{i=0}^{m} a_i u_i(x) + r_a \delta\right]_1$; $[b]_2 \leftarrow \left[\beta + \sum_{i=0}^{m} a_i v_i(x) + r_b \delta\right]_2$;
    $[c]_1 \leftarrow \left[\sum_{i=l+l_w+1}^{m} w_{i-l} \frac{q_i(x)}{\delta} + \frac{h(x)t(x)}{\delta} + a r_b + b r_a - r_a r_b \delta\right]_1 - r\left[\gamma\left(1 + \sum_{i=1}^{l_w} s_i\right)\right]_1$;
    $\textbf{return } (([a]_1, [b]_2, [c]_1), \mathcal{CT} = (c_0, \dots, c_{l_w}, \psi))$;
$\underline{\mathsf{Verify}(\boldsymbol{\sigma}, \boldsymbol{\phi} = \phi_1 \dots \phi_l, \pi = ((a, b, c), (c_0, \dots, c_{l_w}, \psi))):}$
    $\textbf{assert } \sum_{i=0}^{l_w} e(c_i, [t_i]_2) = e(\psi, H)$;
    $\textbf{assert } e(a, b) = e([\alpha]_1, [\beta]_2) + e(\sum_{i=0}^{l} \phi_i [y_i(x)]_1 + \sum_{i=0}^{l_w} c_i, [\gamma]_2) + e(c, [\delta]_2)$;
$\underline{\mathsf{Sim}(\boldsymbol{\tau}, \boldsymbol{\phi} = \phi_1 \dots \phi_l):}$
    $\mu, \nu, c_0, \dots, c_{l_w} \xleftarrow{\$} \mathbb{Z}_p^*$;
    $(a, b, c) \leftarrow \left([\mu]_1, [\nu]_2, \left[\frac{\mu\nu - \alpha\beta - \gamma(\sum_{i=0}^{l} \phi_i y_i(x) + \sum_{i=1}^{l_w} c_i)}{\delta}\right]_1\right)$;
    $\psi \leftarrow \left[\sum_{i=0}^{l_w} t_i c_i\right]_1$;
    $\textbf{return } ((a, b, c), \mathcal{CT} = ([c_0]_1, \dots, [c_{l_w}]_1, \psi))$;
$\underline{\mathsf{Ext}(\boldsymbol{\sigma}, \boldsymbol{\tau}_{\text{ext}} = \{s_i\}_{i=1}^{l_w}, \phi, \pi = (\cdot, (c_0, c_1, \dots, c_{l_w}, \cdot)):}$
    $\textbf{for } i \in [1 \dots l_w]$: $[y_i(x) w_i]_1 \leftarrow c_i - s_i c_0$; $w_i \leftarrow \mathsf{dlog}_{[y_i(x)]_1}\left([y_i(x) w_i]_1\right)$;
    $\textbf{return } w_1, \dots, w_{l_w}$;
$\underline{\mathsf{Rand}(\boldsymbol{\sigma}, \pi = ((a, b, c), (c_0, c_1, \dots, c_{l_w}, \phi)):}$
    $r_1, r_2, r' \xleftarrow{\$} \mathbb{Z}_p^*$;
    $c_0 \mapsto c_0 + r'[\delta]_1$; $c_i \mapsto c_i + r'[\delta s_i]$; $\psi \mapsto \psi + r'[\delta t_0 + \sum_{j=1}^{l_w} \delta t_j s_j]_1$;
    $a \mapsto (1/r_1) a$; $b \mapsto r_1 b + r_1 r_2 [\delta]_2$; $c \mapsto c + r_2 a - r'[\gamma\left(1 + \sum_{i=1}^{l_w} s_i\right)]_1$;
    $\textbf{return } ((a, b, c), (c_0, c_1, \dots, c_{l_w}, \phi))$;
```

Fig. 3. Ext-Groth16: the black-box-extractable SAVER-inspired variant of Groth16. The relation $\mathcal{R}$ must assert that inputs on witness input wires $l \dots l + l_w$ are small enough to be efficiently decryptable. $q_i(x)$ and $y_i(x)$ are as for Groth16, e.g. in Fig. 1.

Technical Details. As Ext-Groth16 is based on SAVER, we point out the important ways it is different from Groth16. First, we extend the CRS with the pk elements, similarly to how it is done in Int-Groth16 (since pk uses Groth16 trapdoors, it changes the security proof). Second, Groth16 itself is modified: while constructing the proof, element c has an additional coefficient, that is needed to balance out ciphertext randomness.

Crucially, Ext-Groth16 cannot achieve black-box strong SE, because it is proof malleable (and rerandomizable). First, the rerandomization of embedded Groth16 still works, because it does not interfere with the "ciphertext randomness cancelling term" of c. Second, ciphertexts are also rerandomizable: we can replace r with $r + r'$ additively in all c_i, in ψ and c (as shown in Fig. 3).

Another important distinction is that in order for the decryption to work efficiently (since it relies on solving discrete logarithm), plaintexts should be small enough. This is critical to guarantee the extraction — to prevent $\mathcal{A}$ from creating un-extractable proofs, we require the circuit itself to make range-checks on plaintext values. We account for the circuit growth in our efficiency evaluation, but in this section we assume the circuit transformation to be an implicit part of the construction, since this suffices for our security analysis.

Finally, we estimate the resulting performance parameters of Ext-Groth16. Construction CRS size (omitting constants) is $(m + 2n + 2l_w)$ $\mathbb{G}_1$, and $(n + l_w)$ $\mathbb{G}_2$. Proof size is $(l_w + 4)$ $\mathbb{G}_1$ and 1 $\mathbb{G}_2$, so $l_w + 2$ times more $\mathbb{G}_1$ than in Groth16. Prover time is (omitting constants) $(m + 3n - l + 2l_w)$ E_1 and n E_2. Verifier time is l E_1 and $(l_w + 5)$ P, so $l_w + 2$ pairings more than in Groth16.

Security. We give a direct proof for the security of Ext-Groth16, as opposed to relying on the security of a transformation as for Int-Groth16. We prove computational zero-knowledge under the standard DDH assumption, as compared to a decisional polynomial assumption introduced and used in SAVER. The weak SE proof is structurally similar to the proof of Theorem 1: that is, we show that either $\mathcal{A}$ reuses a simulated proof (potentially randomizing it), or it does not use simulated data at all, and in that case we can extract the witness. The crucial difference now is that extractor Ext is black-box and operates by decrypting the ciphertext. The proof of the following theorem is deferred to Appendix E.

Theorem 4. *The Ext-Groth16 NIZK argument in Fig. 3 achieves perfect completeness; computational zero-knowledge under the DDH assumption; and black-box weak SE against algebraic adversaries under linear independence of $U = \{u_i(X)\}_{i=0}^{l+l_w}$, and span independence between U and rest of $u_i(X)$.*

Lemma 1. *The Ext-Groth16 NIZK is rerandomizable with* Rand *in Fig. 3.*

Proof. Follows directly from rerandomizability of SAVER in [LCKO19]. □

5 Performance

In this section, we evaluate the efficiency of Int-Groth16 and Ext-Groth16. First, in Table 1, we give a high-level comparison of Groth16 and (the most efficient)

C∅C∅ black-box SE transformation [KZM+15, Section 4]. It shows the asymptotic dependence of the performance metrics on the witness size l_w and the blow-up of the QAP size due to the use of cryptographic primitives for the transformation. Enc_{l_w} denotes an encryption scheme with sufficiently large plaintext size to encrypt the witness. We note that even for Ext-Groth16 a small circuit modification is required, and therefore m grows by $2l_w$ bits, and n grows by l_w; additionally, l_w wires for Ext-Groth16 have 6 times less capacity than for Int-Groth16 and C0C0. Clearly, in Table 1, an overhead of C∅C∅ in CRS size and prover time is strictly bigger than in both constructions we suggest, due to the use of PRF and commitment scheme, and Ext-Groth16 encryption overhead (thus proof size and verification time) is bigger than in first two transformations because of the expansion factor.

Table 1. A comparison of Groth16 with the overhead of C∅C∅ framework and our constructions. Constants are omitted in the case of CRS size and prover's computation. $\mathbb{G}_1$ and $\mathbb{G}_2$: group elements, E: exponentiations and P: pairings. l_w is the number of secret input wires. $e(l_w) = |\mathsf{Enc}_{l_w}|$, $c(l_w) = e(l_w) + |\mathsf{Com}| + |\mathsf{Prf}|$, where $|\mathsf{Op}|$ denotes the number of *constraints* required for the operation $\mathsf{Op} \in \{\mathsf{Enc}_{l_w}, \mathsf{Com}, \mathsf{Prf}\}$. $e_i(l_w), c_i(l_w)$ denote an additional increase in *input wires* (counted in m, but not in n). k_e is Enc expansion factor, can be assumed ≤ 2. The highlighted cells indicate the best efficiency. Top-level parentheses between expressions and units are omitted for better readability.

Construction	Security	CRS	Proof	Prover	Verifier
Groth16, Sec. 3	KS, Weak WB-SE	$m + 2n\ \mathbb{G}_1$ $n\ \mathbb{G}_2$	$2\ \mathbb{G}_1$ $1\ \mathbb{G}_2$	$m + 3n - l\ E_1$ $n\ E_2$	$l\ E_1$ $3\ P$
Groth16 + [KZM+15]	Weak BB-SE	$+3c(l_w) + c_i(l_w)\ \mathbb{G}_1$ $+c(l_w)\ \mathbb{G}_2$	$+k_e l_w\ \mathbb{G}_1$	$+4c(l_w) + c_i(l_w) - O(l_w)\ E_1$ $+c(l_w)\ E_2$	$+k_e l_w\ E_1$
Int-Groth16, Sec. 4.1	Weak BB-SE	$+3e(l_w) + e_i(l_w)\ \mathbb{G}_1$ $+e(l_w)\ \mathbb{G}_2$	$+k_e l_w\ \mathbb{G}_1$	$+4e(l_w) + e_i(l_w) - O(l_w)\ E_1$ $+e(l_w)\ E_2$	$+k_e l_w\ E_1$
Ext-Groth16, Sec 4.2	Weak BB-SE	$+36l_w\ \mathbb{G}_1$ $+12l_w\ \mathbb{G}_2$	$+6l_w + 2\ \mathbb{G}_1$	$+42l_w\ E_1$ $+12l_w\ E_2$	$+6l_w + 2\ P$

We also estimate the *concrete* performance of our two black-box constructions, along the same four performance parameters defined in Table 2, as depending on the bit-size of the encrypted witness. For both NIZKs we will use a 255-bit BLS12-381 curve, defined over a 381 bit prime field. Let us assume that witness size is B_w bits, and it is provided in bit-decomposed form in the original circuit. We aim to optimize proof size, which is important for SNARKs, and thus will only consider encrypting secret inputs at the maximum possible capacity (e.g. we do not encrypt individual bits); the two approaches have different block capacities, so the number of plaintext (and ciphertext) blocks is different in both cases. For Int-Groth16, block size is 248 bits, where the 6 remaining bits are reserved for Koblitz [Kob87] message embedding padding. For Ext-Groth16 we split the plaintext in 43-bit blocks, thus assuming that we can solve 43-bit discrete logarithm for black-box extraction. This explains Ext-Groth16 expansion factor of $6 = \lceil 248/43 \rceil$. We base our circuit design estimates, which are especially relevant to Int-Groth16, on zcash implementation, description of which is provided in [HBHW20] (Section "Circuit Design").

We cover the concrete performance estimation and analysis in Appendix G.

Table 2. Overhead comparison of our constructions over plain Groth16. $\mathbb{G}^{\mathbb{J}}$ stands for bit-size of an encoded JubJub point, and $\mathbb{G}_i$ is the size of an encoded BLS12-381 point. Highlighted cells indicate efficiency improvement.

Construction	CRS	Proof	Prover	Verifier
Int-Groth16	$3286 + 16.4B_w\ \mathbb{G}_1$ $1010 + 5.1B_w\ \mathbb{G}_2$	$\left(\lceil \frac{B_w}{248} \rceil + 1\right) \mathbb{G}^{\mathbb{J}} + B_w$	$4296 + 21.6B_w\ E_1$ $1010 + 5.1B_w\ E_2$	$3\lceil \frac{B_w}{248} \rceil\ E_1$
Ext-Groth16	$0.14B_w\ \mathbb{G}_1$ $0.05B_w\ \mathbb{G}_2$	$\left(\lceil \frac{B_w}{43} \rceil + 2\right) \mathbb{G}_1$	$0.16B_w\ E_1$ $0.05B_w\ E_2$	$\left(\lceil \frac{B_w}{43} \rceil + 2\right)\ P$

Performance Comparison. Our estimates, summarized in Table 2, suggest that both constructions are quite efficient practically. Ext-Groth16 achieves better prover time and CRS size at the expense of slightly bigger proofs and verification time. CRS size and prover time of Ext-Groth16 incur a very small overhead, and are asymptotically *much* smaller than the same numbers for Int-Groth16, giving almost a $100 - 135\times$ performance gain. Hence, we focus our detailed analyses on the proof size and verifier time:

1. *Proof size.* Assuming that encoded BLS12-381 $\mathbb{G}_1$ takes 381 bits, and that JubJub point $\mathbb{G}^{\mathbb{J}}$ takes 256 bits, Int-Groth16 overhead is $\left(\lceil \frac{B_w}{248} \rceil + 1\right)256 + B_w \approx 2.03B_w + 256$ bits, and for Ext-Groth16 it is $\left(\lceil \frac{B_w}{43} \rceil + 2\right)381 \approx 8.86B_w + 762$ bits. Asymptotically, Int-Groth16 proof size is $\times 4.4$ times smaller.
2. *Verifier time.* To compare the increase in exponentiations in Int-Groth16 with the increase in pairings in Ext-Groth16, we use the estimation that micro benchmarks ([AB19, Fig 2], also consistent with [FLSZ17, Table 3] for BN-254) show pairings to be approximately $N = 35$ times slower than processing one element of a multi-exponentiation. Thus, the verification overhead of Int-Groth16 is small for practical witnesses, e.g. $1600 \cdot 3/248 \approx 20$ wires for encrypting 200 bytes, comparing to tens of thousands circuit constraints. The overhead of Ext-Groth16 therefore is about $70\times$ more than for Int-Groth16, although for real-world witnesses it takes less than just a few tens milliseconds, and becomes immaterial for bigger public input sizes.

6 Conclusion and Future Work

We prove two important theorems about [Gro16] and [LCKO19] enabling the composable analysis of provable secure protocols. We conjecture that both our white-box and black-box results generalize to other SNARKs. In fact, we first showed white-box weak SE in a modification of [GM17] with the second equation removed. We decided to focus on Groth16 as the most important SNARK in this family to give a targeted proof and performance analysis. Besides improving performance, we expect weak SE and proof randomization to also have positive cryptographic applications that would be impossible with strong SE — just as for Groth-Sahai proofs [GS08,BCC$^+$09].

Acknowledgements

We thank Mary Maller for encouraging discussions and technical feedback on the white-box weak-SE theorem. 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). Janno Siim was additionally supported by the Estonian Research Council grant PRG49. Most of the work was done while Janno Siim was affiliated with the University of Edinburgh. Karim Baghery has been supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0085, and by Cyber Security Research Flanders with reference number VR20192203.

References

AB19. Shahla Atapoor and Karim Baghery. Simulation extractability in Groth's zk-SNARK. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 336–354. Springer, 2019.

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 Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 118–136. Springer, Heidelberg, July 2019. doi:10.1007/978-3-030-23696-0_7.

BCC$^+$09. Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2009. doi:10.1007/978-3-642-03356-8_7.

BCG$^+$14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014. doi:10.1109/SP.2014.36.

BCG$^+$20. Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. ZEXE: Enabling decentralized private computation. In *2020 IEEE Symposium on Security and Privacy*, pages 947–964. IEEE Computer Society Press, May 2020. doi:10.1109/SP40000.2020.00050.

BG18. Sean Bowe and Ariel Gabizon. Making groth's zk-SNARK simulation extractable in the random oracle model. Cryptology ePrint Archive, Report 2018/187, 2018. `https://eprint.iacr.org/2018/187`.

BMRS20. Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. Coda: Decentralized cryptocurrency at scale. *IACR Cryptol. ePrint Arch.*, 2020:352, 2020. URL: `https://eprint.iacr.org/2020/352`.

BPR20. Karim Baghery, Zaira Pindado, and Carla Ràfols. Simulation extractable versions of groth's zk-SNARK revisited. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 453–461. Springer, Heidelberg, December 2020. `doi:10.1007/978-3-030-65411-5_22`.

BS20. Karim Baghery and Mahdi Sedaghat. Tiramisu: Black-box simulation extractable NIZKs in the updatable CRS model. Technical report, Cryptology ePrint Archive, Report 2020/474, 2020.

BV98. Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 59–71. Springer, Heidelberg, May / June 1998. `doi:10.1007/BFb0054117`.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. `doi: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 Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 683–699. ACM Press, October / November 2017. `doi:10.1145/3133956.3134025`.

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002. `doi:10.1145/509907.509980`.

DDO$^+$01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001. `doi:10.1007/3-540-44647-8_33`.

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 *2016 IEEE Symposium on Security and Privacy*, pages 235–254. IEEE Computer Society Press, May 2016. `doi:10.1109/SP.2016.22`.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. `doi:10.1007/978-3-319-96881-0_2`.

FKMV12. Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Heidelberg, December 2012. `doi:10.1007/978-3-642-34931-7_5`.

FLSZ17. Prastudy Fauzi, Helger Lipmaa, Janno Siim, and Michal Zajac. An efficient pairing-based shuffle argument. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 97–127. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70697-9_4.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_37.

GM17. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63715-0_20.

GOS06. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006. doi:10.1007/11761679_21.

GPS06. S.D. Galbraith, K.G. Paterson, and N.P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. http://eprint.iacr.org/2006/165.

Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006. doi:10.1007/11935230_29.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_11.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008. doi:10.1007/978-3-540-78967-3_24.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. doi:10.1145/1993636.1993651.

HBHW20. Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification, version 2020.1.13. https://github.com/zcash/zips/blob/master/protocol/protocol.pdf, 2020. [Online; accessed: 19/08/2020].

KKK20. Thomas Kerber, Aggelos Kiayias, and Markulf Kohlweiss. Kachina - foundations of private smart contracts. *IACR Cryptol. ePrint Arch.*, 2020:543, 2020. URL: https://eprint.iacr.org/2020/543.

KKKZ19. Thomas Kerber, Aggelos Kiayias, Markulf Kohlweiss, and Vassilis Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *2019 IEEE Symposium on Security and Privacy*, pages 157–174. IEEE Computer Society Press, May 2019. doi:10.1109/SP.2019.00063.

KLO19. Jihye Kim, Jiwon Lee, and Hyunok Oh. Qap-based simulation-extractable SNARK with a single verification. Technical report, Cryptology ePrint Archive, Report 2019/586, 2019.

KMS$^+$16. Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858. IEEE Computer Society Press, May 2016. `doi:10.1109/SP.2016.55`.

Kob87. Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

Kur02. Kaoru Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In David Naccache and Pascal Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 48–63. Springer, Heidelberg, February 2002. `doi:10.1007/3-540-45664-3_4`.

KZM$^+$15. Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. How to use SNARKs in universally composable protocols. Cryptology ePrint Archive, Report 2015/1093, 2015. `http://eprint.iacr.org/2015/1093`.

LCKO19. Jiwon Lee, Jaekyoung Choi, Jihye Kim, and Hyunok Oh. SAVER: Snark-friendly, additively-homomorphic, and verifiable encryption and decryption with rerandomization. Cryptology ePrint Archive, Report 2019/1270, 2019. `https://eprint.iacr.org/2019/1270`.

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

MS07. V. N. Muralidhara and Sandeep Sen. A result on the distribution of quadratic residues with applications to elliptic curve cryptography. In K. Srinathan, C. Pandu Rangan, and Moti Yung, editors, *INDOCRYPT 2007*, volume 4859 of *LNCS*, pages 48–57. Springer, Heidelberg, December 2007.

NT16. Assa Naveh and Eran Tromer. PhotoProof: Cryptographic image authentication for any set of permissible transformations. In *2016 IEEE Symposium on Security and Privacy*, pages 255–271. IEEE Computer Society Press, May 2016. `doi:10.1109/SP.2016.23`.

PV05. Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2005. `doi:10.1007/11593447_1`.

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. `doi:10.1109/SFFCS.1999.814628`.

SBG$^+$19. Samuel Steffen, Benjamin Bichsel, Mario Gersbach, Noa Melchior, Petar Tsankov, and Martin T. Vechev. zkay: Specifying and enforcing data privacy in smart contracts. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1759–1776. ACM Press, November 2019. `doi:10.1145/3319535.3363222`.

Sho04. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/2004/332`.

ZKP19. ZKProof. ZKProof community reference, version 0.2. `https://docs.zkproof.org/pages/reference/reference.pdf/`, December 2019. [Online; accessed 26/06/2020; Updated versions at `https://zkproof.org`].

A Additional Definitions

First, we give a standard definition of knowledge soundness (KS).

Definition 6 (Knowledge Soundness). *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 for* $\mathcal{R}_\lambda$ *we have*

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

We can also have a corresponding notion of weak simulation soundness which is implied by white-box and black-box weak simulation extractability.

Definition 7 (Weak Simulation-Soundness). *We say that* NIZK *is* weakly simulation-sound *if for any PPT adversary* $\mathcal{A}$ *and* $\mathcal{R}_\lambda$ *we have*

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

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* $\mathcal{L}_{\mathcal{R}_\lambda}$ *is the language corresponding to* $\mathcal{R}_\lambda$*.*

Finally, we remind the standard definition of computational zero-knowledge.

Definition 8 (Computational Zero-Knowledge). *We say that* NIZK *is computational zero-knowledge, if for any PPT adversary* $\mathcal{A}$ *and* $\mathcal{R}_\lambda$, $|\varepsilon_0-\varepsilon_1|=\mathrm{negl}(\lambda)$*, where* $\varepsilon_b=\Pr\left[(\boldsymbol{\sigma},\tau)\leftarrow\mathsf{Setup}(\mathcal{R}_\lambda):\mathcal{A}^{\mathcal{S}_{b,\boldsymbol{\sigma},\tau}}(\boldsymbol{\sigma})=1\right]$*. The oracle* $\mathcal{S}_{b,\boldsymbol{\sigma},\tau}$ *on input* (ϕ,w) *asserts that* $(\phi,w)\in\mathcal{R}_\lambda$ *and then returns* $\pi=\mathsf{Prove}(\boldsymbol{\sigma},\phi,w)$ *if* $b=0$*, and* $\pi=\mathsf{Sim}(\boldsymbol{\sigma},\tau,\phi)$ *if* $b=1$*.*

B Lemmas for Algebraic Proofs

The purpose of this section is to give general lemmas for pairing-based SNARKs that have their security based on algebraic assumption. This simplifies proofs for algebraic adversaries later on.

Lemma 2 (Schwartz-Zippel). *Let* $f\in\mathbb{F}[X_1,\dots,X_n]$ *be a non-zero polynomial of degree* $d\geq 0$ *over a field* $\mathbb{F}$*. Let* $S\subset\mathbb{F}$ *finite, and let* $\boldsymbol{x}=(x_1,\dots,x_n)$*. Then* $\Pr_{\boldsymbol{x}\leftarrow S^n}[f(\boldsymbol{x})=0]\leq d/|S|$*.*

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 $\boldsymbol{T}=T_1,\dots,T_n$ 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 set of polynomials $\{P_{\iota,i}(\boldsymbol{T})\}_{\iota,i}$.

Lemma 3 (Algebraic Verification Satisfiability). *Let $\boldsymbol{E} = (E_{1,1}, \dots, E_{1,m_1}, E_{2,1}, \dots, E_{2,m_2})$ 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*[5].

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

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

*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 $\mathcal{A}$.*

Proof (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 $[\boldsymbol{e}]_{1,2} \xleftarrow{\$} \mathcal{A}(\boldsymbol{\sigma})$; $K \leftarrow \mathcal{X}^{\mathsf{alg}}_{\mathcal{A}}(\mathsf{trans}_{\mathcal{A}})$, 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 $\pi = [\boldsymbol{e}]_{1,2}$ independently 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 $\Pr_{\boldsymbol{e} \leftarrow \mathcal{A}(\dots)}[V(\boldsymbol{e}) = 0 \mid V'(\boldsymbol{T}) \neq 0] < \mathsf{negl}(\lambda)$ 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

[5] 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 $[t_{\iota,i}]_\iota$.

non-negligible, which means that V can be satisfied by a prover. Then:

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

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 non-trivially, possibly extracting knowledge about the trapdoor, which allowed it to satisfy 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 parameterized 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. □

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 generalize 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.

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 3, although it slightly increases the required degree of (d_1, d_2)-**dlog**[6].

C Proof of Weak SE for Groth16

Before we start the weak SE proof we present a re-phrased knowledge soundness proof, on top of which we will later build the main theorem.

Below we use the following notation. 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 $P(\boldsymbol{X}) = \sum_M P_{[M]} M$.

[6] 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 $d_1 = 2n-1$.

Theorem 5 ([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 3. When $\mathcal{A}$ presents the proof $\pi = ([a]_1, [b]_2, [c]_1)$ that satisfies the verification equation, 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 3.

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

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

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 [7], thus x^i will appear in coefficients. We start with examining

[7] 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

the following equations, listed by monomials they are produced by, and by the terms of the verification equation they are extracted from:

$$\begin{aligned}
\alpha\beta \text{ in } AB - \alpha\beta &: A_1 B_1 = 1 \implies A_1 \neq 0, B_1 \neq 0 \\
\beta^2 \text{ in } AB &: A_2 B_1 = 0 \implies A_2 = 0 \\
\alpha\gamma &: A_1 B_2 = 0 \implies B_2 = 0 \\
\beta^2/\delta &: \Big(\sum_{i=l+1}^{m} A_{6,i} u_i(x)\Big) B_1 = 0 \implies \sum_{i=l+1}^{m} A_{6,i} u_i(x) = 0 \\
\beta\alpha/\delta &: \Big(\sum_{i=l+1}^{m} A_{6,i} v_i(x)\Big) B_1 = 0 \implies \sum_{i=l+1}^{m} A_{6,i} v_i(x) = 0 \\
\beta/\delta \text{ in } AB &: \Big(\sum_{i=l+1}^{m} A_{6,i} w_i(x) + \sum_{i=0}^{n-2} A_{7,i} x^i t(x)\Big) B_1 + \\
&\quad \Big(\sum_{i=l+1}^{m} A_{6,i} u_i(x)\Big)\Big(\sum_{i=0}^{n-1} B_{4,i} x^i\Big) = 0 \ \wedge \\
1/\delta &: \Big(\sum_{i=l+1}^{m} A_{6,i} w_i(x) + \sum_{i=0}^{n-2} A_{7,i} x^i t(x)\Big)\Big(\sum_{i=0}^{n-1} B_{4,i} x^i\Big) = 0 \\
&\implies \sum_{i=0}^{n-2} A_{7,i} x^i t(x) = 0 \wedge \sum_{i=l+1}^{m} A_{6,i} w_i(x) = 0
\end{aligned}$$

If $(\sum_{i=0}^{n-1} B_{4,i} x^i) = 0$ then from β/δ we have $\sum_{i=l+1}^{m} A_{6,i} w_i(x) + \sum_{i=0}^{n-2} A_{7,i} x^i t(x) = 0$, and otherwise from $1/\delta$ we have $\sum_{i=l+1}^{m} A_{6,i} w_i(x) + \sum_{i=0}^{n-2} A_{7,i} x^i t(x) = 0$. Now, since the sums have different spans of x^i powers, $\sum_{i=0}^{n-2} A_{7,i} x^i t(x) = 0$ and $\sum_{i=l+1}^{m} A_{6,i} w_i(x) = 0$.

$$\begin{aligned}
\beta^2/\gamma \text{ in } AB &: \Big(\sum_{i=0}^{l} A_{5,i} u_i(x)\Big) B_1 = 0 \implies \sum_{i=0}^{l} A_{5,i} u_i(x) = 0 \\
\beta\alpha/\gamma &: \Big(\sum_{i=0}^{l} A_{5,i} v_i(x)\Big) B_1 = 0 \implies \sum_{i=0}^{l} A_{5,i} v_i(x) = 0 \\
\beta/\gamma &: \Big(\sum_{i=0}^{l} A_{5,i} w_i(x)\Big) B_1 + \Big(\sum_{i=0}^{l} A_{5,i} u_i(x)\Big)\Big(\sum_{i=0}^{n-1} B_{4,i} x^i\Big) = 0 \ \wedge \\
1/\gamma &: \Big(\sum_{i=0}^{l} A_{5,i} w_i(x)\Big)\Big(\sum_{i=0}^{n-1} B_{4,i} x^i\Big) = 0
\end{aligned}$$

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.

$$\implies \sum_{i=0}^{l} A_{5,i} w_i(x) = 0 \text{ from } \beta/\gamma \text{ and } 1/\gamma \text{ as with } 1/\delta$$

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:

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

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

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

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

Substituting the first two equations into the left hand side of the third one, using that $A_1 B_1 = 1$:

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

Because a_0 is always 1 and A_1 and B_1 are nonzero, 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. $\square$

Finally, we give a proof for Theorem 1 which shows that Groth16 has white-box weak SE.

*Proof (**Proof of Theorem 1**).* 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 and B (C follows the same pattern as A) from the verification equation as the linear combination of the public CRS and new simulated proof elements:

$$A = \cdots + \sum_{i=1}^{q} A_{8,i}\mu_i + \sum_{i=1}^{q} A_{9,i} \frac{\mu_i\nu_i - \alpha\beta - \sum_{j=0}^{l} a_{i,j}(\beta u_j(x) + \alpha v_j(x) + w_j(x))}{\delta}$$

$$B = \cdots + \sum_{i=1}^{q} B_{5,i}\nu_i$$

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 coefficients of the following monomials (affected by simulated proofs):

$$\alpha\beta \text{ in } AB - C\delta : A_1B_1 - \sum_{i=1}^{q} A_{9,i}B_3 + \sum_{i=1}^{q} C_{9,i} = 1 \qquad \mu_i\beta \text{ in } AB : A_{8,i}B_1 = 0$$

$$\mu_i\nu_j (i \neq j) \text{ in } AB : A_{8,i}B_{5,j} = 0 \qquad \mu_i\gamma \text{ in } AB : A_{8,i}B_2 = 0$$

$$\mu_i\nu_i \text{ in } AB - C\delta : A_{9,i}B_3 + A_{8,i}B_{5,i} - C_{9,i} = 0 \qquad \mu_i\delta \text{ in } AB - C\delta : A_{8,i}B_3 - C_{8,i} = 0$$

$$\mu_i\nu_i\nu_j/\delta \text{ in } AB : A_{9,i}B_{5,j} = 0 \qquad \nu_i\alpha \text{ in } AB : B_{5,i}A_1 = 0$$

$$\mu_i\nu_i\beta/\delta \text{ in } AB : A_{9,i}B_1 = 0 \qquad \nu_i\beta \text{ in } AB : B_{5,i}A_2 = 0$$

$$\nu_i\delta \text{ in } AB : B_{5,i}A_3 = 0$$

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 simulated 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 μ_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).

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

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

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

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:

$$\nu_k \text{ in } AB: \Big(\sum_{i=0}^{n-1} A_{4,i} x^i\Big) B_{5,k} = 0 \Longrightarrow \sum_{i=0}^{n-1} A_{4,i} x^i = 0$$
$$\mu_k \text{ in } AB: \Big(\sum_{i=0}^{n-1} B_{4,i} x^i\Big) A_{8,k} = 0 \Longrightarrow \sum_{i=0}^{n-1} B_{4,i} x^i = 0$$

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[8] 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. □

D Randomizability of Groth16

*Proof (**Proof of Theorem 2**).*
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 $\{\mathsf{Rand}(\pi)\}_\lambda = \{(a', b', c')\}_\lambda$, where π is perhaps not honestly generated, but necessarily verifies. In honestly generated proofs, first two values a, b are independently uniform, and the third element of the tuple is defined from them.

By examining Rand, we immediately see that r_1 makes $a' = r_1 a$ uniform, and that $b' = r_1 b + r_1 r_2 [\delta]_2$ is also independent since $r_1 r_2 \delta$ is uniform because of r_2. Thus we obtain two uniform distributions, and this is true irrespectively of the original distribution of π. Since Rand is correct, the modified proof also verifies. Hence in both distributions the first two tuple elements are uniform, and the third depends on them in the same way, defined by Groth16 verification equation. □

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

Proof (***Proof of Observation 1***).

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:

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

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:

$$A = A_{8,k}\mu_k \qquad B = \frac{1}{A_{8,k}}\nu_k + \frac{C_{8,k}}{A_{8,k}}\delta$$
$$C = \sum_{i=0}^{n-1} C_{4,i}x^i + \sum_{i=0}^{l} C_{5,i}y_i(x) + \sum_{i=l+1}^{m} C_{6,i}\frac{q_i(x)}{\delta} + C_{8,k}\mu_k + \frac{\mu_k\nu_k - \alpha\beta - \sum_{j=0}^{l} a_{k,j}q_j(x)}{\delta}$$

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 $u_i(X)$ for instance and witness wires, and since the verification equation only includes the instance-related sum (formally, we view the monomial β equation; the end of Theorem 1 proof explains the technique). What we left with is precisely the well-known randomization Rand, where $r_1 = 1/A_{8,k}$, and $r_2 = C_{8,k}$:

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

□

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

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

Fig. 4. 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.

Observation 2. 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 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[9] of $V(\boldsymbol{C})$ is as presented in Figure 4.

Proof. 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 $B_1 = 1/A_1$.

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

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

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

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

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

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:

$$\begin{aligned}
\beta &: \Big(\sum_{i=0}^{n-1} A_{4,i}x^i\Big) = A_1\Big(\sum_{i=0}^{l} a_iu_i(x) + \sum_{i=l+1}^{m} C_{6,i}u_i(x)\Big)\\
\alpha &: \Big(\sum_{i=0}^{n-1} B_{4,i}x^i\Big) = \frac{1}{A_1}(\sum_{i=0}^{l} a_iv_i(x) + \sum_{i=l+1}^{m} C_{6,i}v_i(x)\Big)
\end{aligned}$$

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

$$A = A_1\alpha + A_3\delta + A_1\sum_{i=0}^{m} a_iu_i(x) \qquad B = \frac{1}{A_1}\beta + B_3\delta + \frac{1}{A_1}\sum_{i=0}^{m} a_iv_i(x)$$

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

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

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

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:

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

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

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. □

E Security Proofs for Ext-Groth16

*Proof (**Proof of Theorem 4, Perfect Completeness**).* The validity of the statement is ensured by straightforward verification of the simulated ciphertext satisfying both verification equations. □

*Proof (**Proof of Theorem 4, Computational Zero-Knowledge**).* Let $\mathcal{A}$ be an arbitrary PPT adversary that makes q queries to the simulation oracle in the ZK game. We consider a sequence of games and prove that in each game adversary's advantage changes at most by a negligible amount. Let us denote the probability that $\mathcal{A}$ outputs 1 in Game_x by ε_x and let ε_{DDH} denote the maximum distinguishing advantage in the DDH game.

$\underline{\mathsf{Game}_0}$: This is the original game in Definition 8 when $b = 0$. That is, $\mathcal{A}$ can query the oracle $\mathcal{S}_{0,\boldsymbol{\sigma},\tau}$ with inputs $(\phi, w) \in R$ and gets back proofs $\pi = \mathsf{Prove}(\boldsymbol{\sigma}, \phi, w)$.

$\underline{\mathsf{Game}_1}$: In this game, we change the oracle $\mathcal{S}_{0,\boldsymbol{\sigma},\tau}$ to $\mathcal{S}'_{0,\boldsymbol{\sigma},\tau}$ in Fig. 5 that uses the trapdoor τ to simulate a, b, c, and ψ. Elements in the grey box in the figure are changed compared to $\mathcal{S}_{0,\boldsymbol{\sigma},\tau}$.

Let us argue that the probability that $\mathcal{A}$ outputs 1 in the Game_0 is the same as in the Game_1, i.e., $\varepsilon_0 = \varepsilon_1$. Firstly, $c_0, \ldots, c_{l_w}$ are computed the same way in both games. Since $\psi = \sum_{i=0}^{l_w} t_i c_i = t_0[r\delta]_1 + \sum_{i=1}^{l_w} t_i([r\delta s_i + w_i y_{l+i}(x)]_1) = r[\delta(t_0 + \sum_{i=1}^{l_w} t_i s_i)]_1 + \sum_{i=1}^{l_w} w_i [y_{l+i} t_i]_1$, then in both games ψ is uniqely determined by $c_0, \ldots, c_{l_w}$. Finally, a, b are uniformly random in both games and c is the unique value determined by $c_0, \ldots, c_{l_w}, a$ and b. It is easy to verify that c in Fig. 5 does indeed satisfy the verification equation. Hence, output of $\mathcal{S}'_{0,\boldsymbol{\sigma},\tau}$ has the same distribution as the output of $\mathcal{S}_{0,\boldsymbol{\sigma},\tau}$. From this it follows that $\varepsilon_0 = \varepsilon_1$.

$\underline{\mathsf{Game}_{2:(1,1)}}$: For convenience, let us denote the ciphertext elements of different queries by $c_{i,j}$ where $i \in [0, l_w]$ and $j \in [1, q]$. We change the oracle in the previous game such that $c_{1,1}$ is sampled randomly.

```
$\underline{\mathcal{S}'_{0,\boldsymbol{\sigma},\tau}(\phi = \phi_1 \dots \phi_l, w = w_1 \dots w_{m-l})}$:
    $\textbf{if } (\phi, w) \notin \mathcal{R} \textbf{ return } \bot$
    $\mu, \nu, r \xleftarrow{\$} \mathbb{Z}_p^*;\ c_0 \leftarrow r[\delta]_1$
    $c_i \leftarrow r[\delta s_i]_1 + w_i[y_{l+i}(x)]_1 \text{ for } i \in [1 \dots l_w]$
    $(a, b, c) \leftarrow \left([\mu]_1, [\nu]_2, \left[\frac{\mu\nu - \alpha\beta - \gamma\sum_{i=0}^{l}\phi_i y_i(x)}{\delta}\right]_1 + \sum_{i=1}^{l_w}(\frac{c_i}{\delta})\right),\ \ \psi = \sum_{i=0}^{l_w} t_i c_i$
    $\textbf{return } ((a, b, c), \mathcal{CT} \leftarrow (c_0, \dots, c_{l_w}, \psi))$
```

Fig. 5. Simulation oracle $\mathcal{S}'_{0,\boldsymbol{\sigma},\tau}$ in Game_1

We show that $|\varepsilon_1 - \varepsilon_2|$ is bounded by the probability of breaking the DDH assumption. Let us construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to distinguish DDH tuples. The adversary $\mathcal{B}$ gets as an input $[z_x, z_y, z]_1$ where $z_x, z_y \xleftarrow{\$} \mathbb{Z}_p$ and either $z = z_x z_y$ or $z \xleftarrow{\$} \mathbb{Z}_p$. Next, $\mathcal{B}$ samples $\boldsymbol{\sigma}$ and τ except that $[s_1]_1 = [z_x]_1$ and thus that element of τ is unknown. The adversary $\mathcal{B}$ continues by running $\mathcal{A}$ on the input $\boldsymbol{\sigma}$ while simulating the query oracle. The query oracle behavies like $\mathcal{S}'_{0,\boldsymbol{\sigma},\tau}$ except on the first query it outputs $c_{0,1} = \delta[z_x]_1$, $c_{1,1} = \delta[z]_1 + w_1[y_{l+1}(x)]_1$, $c_{1,2} = (\delta s_2)[z_x]_1 + w_2[y_{l+2}(x)]_1$, $\dots$, $c_{1,l_w} = (\delta s_{l_w})[z_x]_1 + w_{l_w}[y_{l+l_w}(x)]_1$. Note that if z is uniformly random then $c_{1,1}$ is uniformly random as in $\mathsf{Game}_{2:(1,1)}$, but when $z = z_x z_y$, then $c_{1,1}$ is a valid ciphetext element as in Game_1, where z_x takes the role of r, and z_y of s_1. Therefore, $|\varepsilon_1 - \varepsilon_{2:(1,1)}| \le \varepsilon_{DDH}$.

$\underline{\mathsf{Game}_{2:(i,j)} \text{ for } i \in [2 \dots l_w], j \in [1 \dots q]}$: We continue with a similar strategy as in $\mathsf{Game}_{2:(1,1)}$. Namely, we change the oracle of the previous game by sampling $c_{i,j}$ uniformly randomly. We use the same reduction idea as in $\mathsf{Game}_{2:(1,1)}$ and show that $|\varepsilon_{2:(i-1,j)} - \varepsilon_{2:(i,j)}| \le \varepsilon_{DDH}$ (or $|\varepsilon_{2:(l_w,j-1)} - \varepsilon_{2:(1,j)}| \le \varepsilon_{DDH}$).

Finally, $\mathsf{Game}_{2:(l_w,q)}$ is the original ZK game where $\mathcal{A}$ has an oracle access to the simulator presented in Fig. 3, which produces all $c_{i,j}$ uniformly random; this is equivalent to honest encryption of a random message. It follows that the advantage that $\mathcal{A}$ breaks ZK is bounded by $l_w \cdot q \cdot \varepsilon_{DDH}$. □

Proof (***Proof of Theorem 4, Weak BB SE***). The knowledge soundness (KS) theorem of [LCKO19] shows how to reduce KS of the SAVER scheme (with two verification equations) to the KS of Groth16. This theorem is also structured as a reduction, but to a weak white-box SE of Groth16 that we have proved in Section 3.

Additionally to the set of elements $\mathcal{A}$ sees in the proof of Theorem 1, we have two more: (1) the CRS is extended with one embedded public key, hence we have elements that depend on the t_i and s_i trapdoors; (2) simulation queries now also produce random ciphertexts $c_{i,j}$ (also, simulated C depends on these ciphertexts, which changes $C_{9,i}$ element).

We write out the representation of A and B (C, Ψ, C_i follow the same pattern as A) from the verification equation as the linear combination of the public CRS and new simulated proof elements:

$$
\begin{aligned}
A = {} & A_1\alpha + A_2\beta + A_3\delta + \sum_{i=0}^{n-1} A_{4,i}x^i + \sum_{i=0}^{l+l_w} A_{5,i}y_i(x) + \sum_{l+l_w+1}^{m} A_{6,i}\frac{q_i(x)}{\delta} \\
& + \sum_{i=0}^{n-2} A_{7,i}\frac{x^i t(x)}{\delta} + \\
& + \sum_{i=1}^{q} \Big(A_{8,i}\mu_i + A_{9,i}\Big(\frac{\mu_i\nu_i - \alpha\beta - \gamma(\sum_{j=0}^{l}\phi_{i,j}y_j(x) + \sum_{j=r}^{l_w} c_{i,j})}{\delta}\Big)\Big) \\
& + \sum_{i=1}^{l_w} A_{10,i}\delta s_i + \sum_{i=1}^{l_w} A_{11,i}t_i y_{l+i}(x) + A_{12}\delta(t_0 + \sum_{i=1}^{l_w} t_i s_i) + A_{13}\gamma(1 + \sum_{i=1}^{l_w} s_i) + \\
& + \sum_{i=1}^{q} \Big(\sum_{j=0}^{l_w} A_{14,i,j}c_{i,j} + A_{15,i}\sum_{j=0}^{l_w} t_j c_{i,j}) \Big)
\end{aligned}
$$

$$
B = B_1\beta + B_2\gamma + B_3\delta + \sum_{i=0}^{n-1} B_{4,i}x^i + \sum_{i=1}^{q} B_{5,i}\nu_i + \sum_{i=0}^{l_w} B_{6,i}t_i
$$

We will refer to the terms $A_1 \dots A_{9,i}$ and $B_1 \dots B_{5,i}$ as "first category" (since they are used in the SE proof), and the other terms are, correspondingly, "second category". We use the same indexing for the first category coefficients as in the SE proof for compatibility; the only difference is that there are fewer $A_{6,i}$ coefficients, and $A_{5,i}$ ranges to $l + l_w$ and not l. Technically, this change is merely syntactical: we could assume secret inputs are part of the (hidden) instance, which would leave the coefficients as they were before (by setting $l = l + l_w$).

We will show that it is possible to extract the witness from the coefficients an algebraic $\mathcal{A}$ returns for the ciphertexts. At the same time, Ext in Fig. 3 is black-box. The security proof will use the white-box extracted coefficients, but they are equal to those returned by Ext because of correctness of the encryption scheme.

We now analyse the first verification equation of $\mathsf{Ext\text{-}Groth16}$:

$$
\prod_{i=0}^{l_w}[C_i]_1[t_i]_2 = [\Psi]_1[1]_2 \quad \text{or, in exponent form:} \quad C_0t_0 + \dots + C_{l_w}t_{l_w} = \Psi
$$

It is immediately clear that Ψ can only be composed of elements that contain t_i, since they are in the immutable part of the left hand side:

$$
\prod_{i=0}^{l_w} C_i t_i = \sum_{i=1}^{l_w} \Psi_{11,i}t_i y_{l+i}(x) + \Psi_{12}\delta(t_0 + \sum_{i=1}^{l_w} t_i s_i) + \sum_{i=1}^{q} \Psi_{15,i}(\sum_{j=0}^{l_w} t_j c_{i,j})
$$

Now, we derive the restrictions on the ciphertext coefficients C_i. Going through the other elements of the equation, to balance properly, each C_i must consist of either: (1) some strictly other t_j (clearly we cannot have t_i^2 in the equation; in particular, $C_{i,12} = C_{i,15,j} = 0$ for all $j \in [1 \dots l]$ because of that); or (2) $y_{l+i}(x)$ for $i > 0$; or (3) δ for $i = 0$ and $s_i\delta$ for $i > 0$; or (4) simulated ciphertexts $\{c_{j,i}\}_{j=1}^{q}$. Substituting it into the equation:

$$\Big(C_{0,3}\delta + \sum_{i=1}^{l_w} C_{0,11,i} t_i y_{l+i}(x) + \sum_{i=1}^{q}\sum_{j=0}^{l_w} C_{0,14,i,j} c_{i,j}\Big) t_0$$
$$+ \sum_{j=1}^{l_w} \Big(C_{j,10,j} s_j \delta + C_{j,5,j} y_{l+j}(x) + \sum_{i=1, i\neq j}^{l_w} C_{j,11,i} t_i y_{l+i}(x) + \sum_{i=1}^{q}\sum_{k=0}^{l_w} C_{j,14,i,k} c_{i,k}\Big) t_j$$
$$= \sum_{i=1}^{l_w} \Psi_{11,i} t_i y_{l+i}(x) + \Psi_{12}\delta(t_0 + \sum_{i=1}^{l_w} t_i s_i) + \sum_{i=1}^{q} \Psi_{15,i}(\sum_{j=0}^{l_w} t_j c_{i,j})$$

From δt_0 we obtain $C_{0,3} = \Psi_{12}$, and for $i > 0$ from $\delta t_j s_j$ we get $\Psi_{12} = C_{j,10,j}$. Now, looking at $t_j y_{l+j}(x)$ (in fact, on $t_j x/\gamma, \alpha t_j x/\gamma, \beta t_j x/\gamma$), we also get $C_{j,5,j} = \Psi_{11,j}$. From each $t_0 c_{i,j}$ we derive that $C_{0,14,i,0} = \Psi_{15,i}$, and all other $C_{0,14,i,j} = 0$; similarly only $C_{j,14,i,j} = \Psi_{15,i}$. Finally, notice that $t_j t_0$ for $j > 0$ cannot be balanced by anything from C_j ($C_{j,11,i}$ start from $i = 1$) or Ψ, so all $C_{0,11,i} = 0$. Applying these changes, we derive:

$$\Big(\Psi_{12}\delta + \sum_{i=1}^{q} \Psi_{15,i} c_{i,0}\Big) t_0$$
$$+ \sum_{j=1}^{l_w} \Big(\Psi_{12} s_j \delta + \Psi_{11,j} y_{l+j}(x) + \sum_{i=1, i\neq j}^{l_w} C_{j,11,i} t_i y_{l+i}(x) + \sum_{i=1}^{q} \Psi_{15,i} c_{i,j}\Big) t_j$$
$$= \sum_{i=1}^{l_w} \Psi_{11,i} t_i y_{l+i}(x) + \Psi_{12}\delta(t_0 + \sum_{i=1}^{l_w} t_i s_i) + \sum_{i=1}^{q} \Psi_{15,i}(\sum_{j=0}^{l_w} t_j c_{i,j})$$

The coefficients of the resulting equation represent the following logic: (1) original honest ciphertexts ($\Psi_{11,j}$ are encrypted witness wires, and Ψ_{12} is randomness), (2) homomorphically added simulation ciphertexts ($\Psi_{15,i}$ for $i = 1 \dots q$), and (3) linear combination on the left hand side (nonzero $C_{j,11,i}$).

We now argue that all $C_{j,11,i} = 0$, and thus no nontrivial linear combination is possible. Assuming the contrary, and analysing monomial $t_{k_1} t_{k_2}$ for some pair of positive indices $k_1 \neq k_2$ (both $\in [1 \dots l_w]$), we have $C_{k_1,11,k_2} y_{l+k_2}(x) + C_{k_2,11,k_1} y_{l+k_1}(x) = 0$. This, in turn, implies, simultaneously, $C_{k_1,11,k_2} f_{l+k_2}(x) + C_{k_2,11,k_1} f_{l+k_1}(x) = 0$, for $f_i(X) = v_i(X), u_i(X), w_i(X)$ (viewing $\alpha x, \beta x, x$). But we assumed $\{u_i(X)\}_{i=l+1}^{l_w}$ to be linearly independent, and therefore all $C_{j,11,i} = 0$.

The resulting view on the equation is now:

$$\Big(\Psi_{12}\delta + \sum_{i=1}^{q} \Psi_{15,i} c_{i,0}\Big) t_0 + \sum_{j=1}^{l_w} \Big(\Psi_{12} s_j \delta + \Psi_{11,j} y_{l+j}(x) + \sum_{i=1}^{q} \Psi_{15,i} c_{i,j}\Big) t_j$$

$$= \sum_{i=1}^{l_w} \Psi_{11,i} t_i y_{l+i}(x) + \Psi_{12}\delta(t_0 + \sum_{i=1}^{l_w} t_i s_i) + \sum_{i=1}^{q} \Psi_{15,i}(\sum_{j=0}^{l_w} t_j c_{i,j})$$

As we can see, it deviates from the proof of SAVER KS in exactly one aspect: $\mathcal{A}$ can combine its message encryption with the simulated zero-ciphertexts homomorphically. This does not give $\mathcal{A}$ any real power: we will show that this combination cannot satisfy the *second* verification equation, because $\mathcal{A}$ cannot produce the "ciphertext randomness cancelling value" for C.

After showing how exactly C_i are restricted, our next step is substituting their general form into the second equation which corresponds to the verification equation of Groth16:

$$AB - \alpha\beta - \Bigg(\sum_{i=0}^{l} \phi_i y_i(x) + \Big(\Psi_{12}\delta + \sum_{i=1}^{q} \Psi_{15,i} c_{i,0}\Big) + \sum_{j=1}^{l_w} \Big(\Psi_{12} s_j \delta + \Psi_{11,j} y_{l+j}(x) + \sum_{i=1}^{q} \Psi_{15,i} c_{i,j}\Big)\Bigg)\gamma - C\delta = 0$$

We now follow the SE proof reduction: the block of 11 equations from which it starts remains the same. The equation extracted from $\alpha\beta$ is not affected by the change of terms since no additional $\alpha\beta$ terms are created either by second category coefficients, or by the ciphertext terms. The other 10 equations depend either on μ_i or ν_i $(\mu_i\nu_j, \mu_i, \nu_i, \ldots, \nu_i\delta)$, and they are exactly the same in our case too, since (1) they do not contain $A_{5,i}$ and $A_{6,i}$, (2) the second category monomials do not contain μ_i or ν_i, and (3) the ciphertext coefficients do not either. Hence, in both cases all $A_{9,i} = 0$, and we follow the branching of the SE proof:

– *Non-simulation case.* All the simulation elements are zero: $A_{9,i} = A_{8,i} = B_{5,i} = C_{8,i} = C_{9,i} = 0$, and thus we have as before $A_1 B_1 = 1$. From β^2 and $\alpha\delta$, we get $A_2 = B_2 = 0$.
  From Equation $(c_{i,0}\gamma)$: $A_{9,i} B_{4,0} + A_{14,i,0} B_2 - \Psi_{15,i} - C_{9,i} = 0$, so $\Psi_{15,i} = 0$ — this means $\mathcal{A}$ cannot add simulated ciphertexts into a non-simulated one. Indeed, to balance out simulated $c_{i,j}$ $\mathcal{A}$ would need to add the cancelling coefficient to C, but it cannot do that since it is part of $C_{9,i}$ which is zero.
  We easily cancel all the second category terms from A, B. From Equation $(\beta\delta s_i)$ we have $A_{10,i} = 0$, from monomials $\beta t_i x^i$ we get $\sum_{i=1}^{l_w} A_{11,i} t_i y_{l+i}(x) = 0$ (coefficients may be nonzero since forming a linear combination may be possible). From Equation $(\delta\beta t_0)$, $A_{12} B_1 = 0$, so $A_{12} = 0$. From Equation $(\gamma\beta)$: $A_2 B_2 + A_{13} B_1 = 0$, so $A_{13} = 0$. At the same time, from Equation $(\beta c_{i,j})$: $A_{14,i,j} B_1 = 0$, and from Equation $(\beta c_{i,0} t_0)$: $A_{15,i} B_1 = 0$, so all $A_{14,i,j} = A_{15,i} = 0$. Considering Equation (αt_i) we get $A_1 B_{6,i} = 0$, so $B_{6,i} = 0$.
  To characterise Ψ_{12} we must look at Equation $(\delta\gamma)$: $A_3 B_2 + A_{13} B_3 - \Psi_{12} - C_{13} = 0$. Since $B_2 = A_{13} = 0$, we derive $C_{13} = -\Psi_{12}$ — that is, the cancelling

coefficient in C must be balanced out by the ciphertext randomness, which is in line with honest proof generation logic.
Other second category terms in C also cancel: $C_{10,i} = 0$ because of $\delta^2 s_i$; $C_{11,i} = 0$ from $\delta t_i x^i$; $C_{12,i} = 0$ because of $\delta^2 t_0$; and $C_{14,ij}$ with $C_{15,i}$ are zero from $\delta c_{i,j}$ and $\delta c_{i,j} t_j$ correspondingly.
Now the verification equation looks like as if $\mathcal{A}$ uses a single honestly constructed ciphertext:

$$AB - \alpha\beta - \Big(\sum_{i=0}^{l} \phi_i y_i(x) + \Psi_{12}\delta + \sum_{j=1}^{l_w} \Big(\Psi_{12} s_j \delta + \Psi_{11,j} y_{l+j}(x)\Big)\Big)\gamma + C\delta = 0$$

Moreover, as we showed, Ψ_{12} (ciphertext cancelling term coefficient) is cancelled out by C_{13} — the only nonzero from the second category coefficients. So from here we can reduce to the basic Groth16, with the only difference that now $A_{5,i}$ has more wires and $A_{6,i}$ has less. We show that this minor change does not significantly affect the proof of Groth16 KS.
For all the equations in the KS proof until we get to critical equations (that is, $(\beta^2/\delta, \beta\alpha/\delta, \beta/\delta, 1/\delta, \beta^2/\gamma, \beta\alpha/\gamma, \beta/\gamma, 1/\gamma)$) the only change that happens is that whenever a sum with $A_{5,i}$ appears it now spans to $l + l_w$ not to l, and whenever a sum with $A_{6,i}$ appears, it goes from $l + l_w + 1$ to m, not from $l+1$. This is easy to verify, since the only new monomials that we have are $\delta\gamma$ (with Ψ_{12} and with C_{13}; is not part of the monomials listed), $\delta\gamma s_i$ (similarly), and $\Psi_{11,j} y_{l+j}(x)\gamma$ only affect critical equations. And other second category elements are zero.
Looking at the third critical coefficient (corresponding to powers of x only) we see almost the same equation as in the KS proof, except now $\Psi_{11,i}$ are instead of first l_w wires of $C_{6,i}$. No other new terms are added, and coefficients with B_2, $A_{6,i}$ and $A_{7,i}$ are cancelled as before, which gives:

$$\Big(\sum_{i=0}^{n-1} A_{4,i} x^i\Big)\Big(\sum_{i=0}^{n-1} B_{4,i} x^i\Big) = \sum_{i=0}^{l} \phi_i w_i(x) + \\ \sum_{i=l+1}^{l+l_w} \Psi_{11,i-l} w_i(x) + \sum_{i=l+l_w+1}^{m} C_{6,i} w_i(x) + \sum_{i=0}^{n-2} C_{7,i} x^i t(x)$$

To argue that $A_{4,i}$ and $B_{4,i}$ form $u_i(x)$ and $v_i(x)$ sets, we, as in the KS proof, look at α and β:

$$\beta: \quad \Big(\sum_{i=0}^{n-1} A_{4,i} x^i\Big) B_1 = \sum_{i=0}^{l} \phi_i u_i(x) + \sum_{i=l+1}^{l+l_w} \Psi_{11,i-l} u_i(x) + \sum_{i=l+l_w}^{m} C_{6,i} u_i(x)$$

$$\alpha: \quad \Big(\sum_{i=0}^{n-1} B_{4,i} x^i\Big) A_1 = \sum_{i=0}^{l} \phi_i v_i(x) + \sum_{i=l+1}^{l+l_w} \Psi_{11,i-l} v_i(x) + \sum_{i=l+l_w}^{m} C_{6,i} v_i(x)$$

Therefore we trivially conclude, substituting the last two equations into the previous one, and extracting from $\Psi_{11,j}$, as we extracted from $C_{6,j}$ instead in the KS case.

– *Simulation Case.* This branch is characterised by $\mathcal{A}$ reusing the simulated proof number k with $C_{9,k} = 1$; as we showed in Groth16 weak SE proof, all other $A_{8,i}, B_{5,i}$ and $C_{9,i}$ are zero. From the very same block of 11 equations we derive, as before: $A_1 = A_2 = A_3 = B_1 = B_2 = 0$.
  Since simulation "ciphertext cancelling" terms are embedded with simulated C, and only $C_{9,k}$ is nonzero, $\mathcal{A}$ cannot use any other set of ciphertexts than $\{c_{k,i}\}_i$. Formally, we show it by looking at Equation ($\gamma c_{i,0}$): $A_{9,i}B_{4,0} + A_{14,i,0}B_2 - \Psi_{15,i} + C_{9,i} = 0$, so we conclude that $\Psi_{16,i} = 0$ for $i \neq k$, and for $i = k$ since all $A_{9,i} = 0$ we get $\Psi_{15,k} = A_{9,k}B_{4,0} + C_{9,k} = 0 + 1 = 1$. That is, $\mathcal{A}$ uses exactly one simulated ciphertext vector, unmodified.
  We now cancel all the second category terms for A and B, looking at combinations of coefficients in A with nonzero $B_{5,k}(\nu_k)$ and the coefficients of B with $A_{8,k}(\mu_k)$; all they are extracted from AB only. From Equation ($\delta s_i \nu_k$): $A_{10,i}B_{5,k} = 0$, so $A_{10,i} = 0$. Looking at monomials $\nu_k t_i \alpha x, \nu_k t_i \beta x, \nu_k t_i x$ simultaneously: $A_{11,i} t_i y_{l+i}(x) B_{5,k} \nu_k = 0$, hence $A_{11,i} = 0$. From Equation ($\delta t_0 \nu_k$), $A_{12}B_{5,k} = 0$, so $A_{12} = 0$. From Equation ($\gamma \nu_k$): $A_{13}B_{5,k} = 0$, so $A_{13} = 0$. Monomials with $c_{i,j}\nu_k$ do only appear in $A_{14,i,j}, A_{15,i}$ since $A_{9,i} = 0$, so from Equations $c_{i,j}\nu_k, c_{i,j}t_j\nu_k$ all these coefficients are zero in a similar manner. Finally, we cancel $B_{6,i}$ by looking at $t_i \mu_k$ which gives us $A_{8,k}B_{6,i} = 0$.
  Regarding Ψ_{12}, as in the non-simulation case, we look at Equation ($\delta\gamma$): $A_{13}B_3 - \Psi_{12} - C_{13} = 0$, which simplifies to $\Psi_{12} = -C_{13}$. This means that $\mathcal{A}$ can indeed add its own randomness to C, because it can cancel it out exactly with C_{13}.
  Except for C_{13} we can cancel all the other second category terms of C. $C_{10,i}$ is zero because $\delta^2 s_i$ are only balanced by $A_{10,i}B_3$ but $A_{10,i} = 0$; the same thing happens to $C_{11,i}$ because of $\delta t_i(\alpha x + \beta x + x)$, to C_{12} ($\delta^2 t_0$ and $\delta^2 t_i s_i$), to $C_{14,i,j}$ ($\delta c_{i,j}$), and to $C_{15,i}$ ($\delta c_{i,j} t_j$).
  We have now cancelled all the second category terms of A, B, C except for C_{13} balancing out Ψ_{12}. Now it is possible to proceed with the reduction exactly as in the second branch of the Groth16 weak SE proof, having in mind the similar difference with $A_{5,i}, A_{6,i}$ elements explained in the first branch of the current proof. In particular, we argue that $\Psi_{11,j}$ cancel from the third critical equation:

$$0 = \sum_{i=0}^{l} a_i u_i(x) + \sum_{i=l+1}^{l+l_w} \Psi_{11,i-l} u_i(x) + \sum_{i=l+l_w+1}^{m} C_{6,i} u_i(x) - \sum_{i=0}^{l} a_{k,i} u_i(x)$$

  Similarly to Groth16 weak SE, $\sum_{i=l+l_w+1}^{m} C_{6,i} u_i(x) = 0$ because it is linearly independent from all the $0 \ldots l + l_w$ input wires. Then, because input wires are independent, all $\Psi_{11,i-1} = 0$ (which forbids adding nonzero honest ciphertexts), and $a_i = a_{k,i}$. Hence, $\mathcal{A}$ has reused the simulated proof number k, but potentially with ciphertext randomization (Ψ_{12}) additionally to the (A, B, C) randomization of Groth16. $\square$

F Vector ElGamal Cryptosystem

A public key cryptosystem is a triple of efficient algorithms $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ where $\mathsf{KGen}(1^\lambda)$ outputs a public key pk and a secret key sk, Enc takes as an input a public pk and a message m and outputs a ciphertext c, and Dec takes in the secret key sk and a ciphertext c and outputs the message m. We require that $m = \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m))$ for any valid pk, sk, and m. A standard privacy property for public cryptosystems is INDistinguishability under Chosen Plaintext Attacks (IND-CPA) which intuitively says that an efficient adversary has a negligible advantage of distinguishing $c_1 = \mathsf{Enc}(\mathsf{pk}, m_1)$ from $c_2 = \mathsf{Enc}(\mathsf{pk}, m_2)$ where m_1, m_2 are chosen by the adversary.

We describe a common variant of ElGamal cryptosystem that can be used to encrypt a vector of group elements. Let the message space be $\mathbb{G}^n$ for some integer n. The vector ElGamal cryptosystem works as follows:

- $\mathsf{KGen}(1^\lambda)$: Samples $s_1, \ldots, s_n \xleftarrow{\$} \mathbb{Z}_p$ and returns $\mathsf{pk} \leftarrow [s_1, \ldots, s_n]$ and $\mathsf{sk} \leftarrow \{s_i\}_{i=1}^n$.
- $\mathsf{Enc}(\mathsf{pk}, \{[m_i]\}_{i=1}^n)$: Samples $r \xleftarrow{\$} \mathbb{Z}_p$ and returns $\boldsymbol{c} \leftarrow [r, rs_1 + m_1, \ldots, rs_n + m_n]$.
- $\mathsf{Dec}(\mathsf{sk}, [c_0, \ldots, c_n])$: Computes $[m_i] = [c_i] - s_i[c_0]$ for $i = 1, \ldots, n$ and returns $\{[m_i]\}_{i=1}^n$.

It is also possible to have $\mathbb{Z}_p^n$ as a message space, but then messages have to be small enough to compute the discrete logarithm. Both variants are known to be IND-CPA secure [Kur02] under the well-known DDH assumption described below.

Definition 9 (Decisional Diffie-Hellman assumption). *Let $\mathbb{G}$ be a cyclic group with a generator $G = [1]$. We say that the DDH assumption holds in $\mathbb{G}$, if for all PPT $\mathcal{A}$, $\mathsf{Adv}_{\mathbb{G},\mathcal{A}}^{DDH} \triangleq |\varepsilon_0 - \varepsilon_1| = \mathrm{negl}(\lambda)$, where $\varepsilon_b \triangleq \Pr\left[x, y, z \xleftarrow{\$} \mathbb{Z}_p^* : \mathcal{A}([x], [y], [xy + bz]) = 1\right]$.*

G Concrete Performance Analysis

We cover in this section the concrete performance analysis that was left out from Section 5. We base our circuit size estimates on the Zcash specification [HBHW20], and assume that all both constructions are using BLS12-381 curve.

Performance of $\mathsf{Int\text{-}Groth16}$. The performance overhead of $\mathsf{Int\text{-}Groth16}$ compared to $\mathsf{Groth16}$ depends mostly on the increase in circuit size — we must implement the encryption scheme itself, and also the infrastructure that converts the input (plaintext) data to the desired form, which we discuss first. We remind that JubJub forms a 252 bit group over 255 bit prime field, which is equal to the group size of BLS, allowing seamless integration of one into another.

Plaintext embedding into the JubJub curve — that is, converting plaintext blocks into JubJub points — is the main technical challenge, which we solve using the approach of Koblitz [Kob87] to overcome it. To embed a plaintext block w_i of bit-length $(254 - \log_2 \kappa)$ into the curve we reserve a padding p_i for a nonce of length $\log_2 \kappa$: with probability $1 - 2^{\kappa}$ the concatenation $w_i \| p_i$ is a valid x-coordinate for some p. Because of how lengths are chosen, $m_i \| p_i$ is always smaller than the prime field of JubJub. For practical purposes it is enough to reserve 6 bits for the padding, which gives $\kappa = 64$, leaving 248 bits for the message block, and thus $l_w = \lceil B_w / 248 \rceil$. To avoid issues with completeness — since now it is possible, with negligible probability, that some w_i does not have a suitable padding — we allow a fallback mechanism [MS07], in which a random blinding b_i is chosen, and the algorithm is repeated for $w_i + b_i$, and b_i is attached to the ciphertext in clear. To avoid attacks where $\mathcal{A}$ finds non-encodable witnesses, we generate this b_i every single time (otherwise the presence of nonzero b_i may leak something about the message). From the circuit side, one extra wire per plaintext block is required to contain b_i, but it takes no extra mul-gates, since any gate that uses w_i can use $w_i + b_i$ for free. We will denote this procedure by $\mathsf{Embed}(w_i, b_i, p_i) = ((w_i \oplus b_i) \parallel p_i, y)$, where y coordinate is computed from x. Instead of computing y in the circuit, we pass it through the intermediate secret wires, and just check that (x, y) is on the curve, which takes just 4 constraints.

The second issue is that we must verify that circuit inputs corresponding to w_i are of a right bit-size in order to guarantee correctness of decryption (used in extraction). The standard way of solving this is to represent the values as bit-vectors of the needed size. That is, for each encrypted element $e = w_i$ we supply its bit decomposition $\{e_i\}_{i=0}^{n-1}$ explicitly and assert that $e = \sum 2^i e_i$, which certifies that e is a n bit value. This takes n gates to ensure each e_i is a binary value, and then one gate is needed to combine them all. Converting from bit-decomposed representation to any other base is inexpensive, and takes one gate per digit in the chosen base, so we assume that application (as opposed to the encryption-correctness check) is taking each w_i in bit-decomposed representation. To simplify the comparison, we assume that this representation is used in the original non-transformed circuit too, so we omit boolean-checks on w_i from both estimates.

We now discuss the encryption scheme itself. To verify the ciphertext $c_0, c_1, \ldots,$ c_{l_w} we must check that $c_0 = r[1]_1$ and $c_i = r[s_i]_1 + \mathsf{Embed}(w_i, b_i, p_i)$. All the JubJub exponentiations here take 750 multiplication gates for each full-bit exponent — we assume that public key points are embedded into the circuit, thus we use an algorithm for fixed-base exponentiation.

- Checking c_0 requires 254 constraints for bit-range check, 750 gates for fixed-base exponentiation, 2 constraints to compare both coordinates of c_0 to the computed ones, and to perform $\mathsf{onCurve}$ check for c_0 coordinates (4 gates); together it takes 1010 constraints. The number of *additional input wires* that we introduce (for r and c_i) is 256, which we include in computation of m but not n.

– For the c_i checks we must additionally perform embedding and sum the embedded point with $r[s_i]$. For the embedding, our circuit accepts bit-decomposed w_i, b_i, p_i and y (second coordinate of embedded point). Bit-checking b_i, p_i (w_i is excluded, r is bit-checked once and counted for c_0 check already) takes $248 \cdot 2 + 6$ constraints. In order to optimize l we will pass b_i as field elements through public inputs and perform an equality-check with the bit-decomposed witness-passed b_i; this takes 1 constraint. Next, we must bit-add w_i and b_i, which takes 250 constraints. We can now pack the bits corresponding to $x = (w_i \oplus b_i) \parallel p_i$ into a field element with a single gate, and it takes 4 gates to verify $\mathsf{onCurve}(x, y)$. We sum the value with $r[s_i]$ (this takes 750 constraints to compute), which takes extra 6 constraints. On-curve check for c_i and comparison of c_i with the computed points takes 6 gates. In total, we get 1273 constraints. The number of additional input wires is 258 (b_i, p_i, y, c_i).

Combining it all together, n increases by $\Delta_n = 1010 + 1273 l_w$, m increases by $\Delta_m = \Delta_n + 256 + 258 l_w = 1266 + 1531 l_w$, and l increases by $\Delta_l = 3 l_w$ because of public blinding factors (l_w field elements) and two values for c_i.

We now analyse the NIZK parameters we compare along:

1. *CRS size.* We get extra $\Delta_m + 2\Delta_n = 3286 + 4077 l_w$ $\mathbb{G}_1$, and $\Delta_n = 1010 + 1273 l_w$ $\mathbb{G}_2$. Converting it to B_w, extra $3286 + 16.4 B_w$ $\mathbb{G}_1$ and $1010 + 5.1 B_w$ $\mathbb{G}_2$.
2. *Proof size.* Using ElGamal we have $l_w + 1$ points per message block, and l_w 248-bit blinding factors. This results in additional $\lceil B_w/248 \rceil + 1$ $\mathbb{G}_1$ plus B_w bits.
3. *Prover time.* $\Delta_m + 3\Delta_n - \Delta_l = 4296 + 5347 l_w = 4296 + 21.6 B_w$ E_1. As in the CRS, $1010 + 5.1 B_w$ E_2.
4. *Verifier time.* Extra $\Delta_l = 3 l_w \approx 0.012 B_w$ exponentiations, plus time to decode c_i (finding second point coordinate), which we ignore in our comparison.

Performance of $\mathsf{Ext\text{-}Groth16}$. As we mentioned before, efficient black-box extraction from $\mathsf{Ext\text{-}Groth16}$ is only possible if encrypted plaintext values are small enough, since the decryption algorithm needs to solve DLP for each ciphertext element. We assume that it is feasible to solve 43 bit DLP, which splits every 128 bits into 3 blocks.

Compared to $\mathsf{Groth16}$, $\mathsf{Ext\text{-}Groth16}$ has two types of overhead: on the first, structural layer, it has additional CRS elements (for the public key), ciphertext proof elements, prover exponentiations, and verifier pairings; on the second infrastructural layer, the circuit should be changed to assert that encrypted wire values fit into 43 bits, and then to convert from this representation to the desired one. We will denote the number of secret input wires by $l'_w = \lceil B_w/43 \rceil$ to distinguish it from the number of wires l_w in the more efficienly-packed $\mathsf{Int\text{-}Groth16}$.

Regarding infrastructure, since we compare to the circuit which already uses binary-decomposed witness, all the bits are checked to be binary, so the only real overhead is to pack them into field values and compare to the plaintext values that are plugged-in externally. Each comparison takes just one constraint, so we have extra $\Delta_n = l_w$ constraints. We also have an additional input wires for ciphertexts, so $\Delta_m = \Delta_n + l_w = 2l_w$. Although we connect the ciphertexts externally, formally they are not counted as public inputs, so $\Delta_l = 0$.

This gives, for the four parameters:

1. *CRS size.* $\Delta_m + 2\Delta_n + 2l_w = 6l_w \approx 0.14B_w\ \mathbb{G}_1$. For second group elements, $\Delta_n + l_w = 2l_w = 0.05B_w\ \mathbb{G}_2$.
2. *Proof size.* We produce $l'_w + 2$ extra ciphertext points in $\mathbb{G}_1$.
3. *Prover time.* For E_1, $\Delta_m + 3\Delta_n - \Delta_l + 2l_w = 7l_w = 0.16B_w\ E_1$. The overhead for E_2 is the same as of $\mathbb{G}_2$ in CRS size.
4. *Verifier time.* We need to compute $l'_w + 2$ more pairings than in Groth16, and no additional exponentiations, since $\Delta_l = 0$.