An extended abstract of this work appears in *PKC'18*. This is the full version.

Subversion-zero-knowledge SNARKs

Georg Fuchsbauer[1]

Abstract

Subversion zero knowledge for non-interactive proof systems demands that zero knowledge (ZK) be maintained even when the common reference string (CRS) is chosen maliciously. SNARKs are proof systems with succinct proofs, which are at the core of the cryptocurrency Zcash, whose anonymity relies on ZK-SNARKs; they are also used for ZK contingent payments in Bitcoin.

We show that under a plausible hardness assumption, the most efficient SNARK schemes proposed in the literature, including the one underlying Zcash and contingent payments, satisfy subversion ZK or can be made to at very little cost. In particular, we prove subversion ZK of the original SNARKs by Gennaro et al. and the almost optimal construction by Groth; for the Pinocchio scheme implemented in libsnark we show that it suffices to add 4 group elements to the CRS. We also argue informally that Zcash is anonymous even if its parameters were set up maliciously.

Keywords: Zero knowledge, SNARKs, parameter subversion, Zcash, Bitcoin contingent payments.

1 Introduction

One of the primary motivations for succinct non-interactive arguments (SNARG) was verifiable computation. Consider a client that outsources resource-intensive computation to a powerful server, which attaches a *proof* to the result, so the client is convinced that it was computed correctly. For this to be meaningful, verification of such a proof must be considerably more efficient than performing the computation in the first place. SNARG systems provide such proofs and an impressive line of research has led to more and more efficient systems with proofs of size less than a kilobyte that can be verified in milliseconds. The reason why SNARGs are not used in outsourcing of computation is that computing a proof for complex computations is still not practical. (For example, a proof in Zcash, which is for a very simple statement, takes minutes to compute on a PC.)

Zero-knowledge (ZK) SNARGs are used when some inputs to the computation come from the prover (the server in our example), who wants to keep its inputs private. ZK systems guarantee that a proof does not reveal more about private inputs than what can be inferred from the result of the computation. If the proofs prove knowledge of the private inputs, they are called SNARKs. ZK-SNARKs are already deployed, for example in Zcash [Zca], which is a cryptocurrency like Bitcoin [Nak09], based on the Zerocash protocol [BCG+14a]. As opposed to Bitcoin, where all transactions are public, Zcash payments are fully anonymous and protect the users' privacy. Zcash achieves this by using SNARK proofs that are zero-knowledge.

[1] Inria, École normale supérieure, CNRS, PSL Research University, France. georg.fuchsbauer@ens.fr, www.di.ens.fr/~fuchsbau. The author is supported by the French ANR EfTrEC project (ANR-16-CE39-0002).

Zero-knowledge contingent payments use SNARKs for *fair exchange* of information against payments over the Bitcoin network, assuming that the information can be verified (in the sense that it can be formalized as the witness of an NP statement), e.g. solutions to a Sudoku puzzle. Bitcoin's scripting language defines *Pay-to-PubkeyHash* transactions, which are bound to a hash value y and can be redeemed by exhibiting a preimage, i.e., some x s.t. $H(x) = y$. In a contingent payment Alice, the seller, chooses a key k, encrypts the information she is offering as c under k and sends c together with $y := H(k)$ to Bob, the buyer. Bob makes a transaction to y. To redeem it, Alice must publish the preimage k, which then allows Bob to decrypt c and obtain the purchased information. To prevent Alice from cheating, she must prove that c encrypts the desired information under a preimage of y, for which she can use SNARKs. Zero-knowledge guarantees that no information is leaked before being paid.

The main drawback of SNARKs is that they require system parameters that must be generated in a trusted way. In particular, whoever knows the randomness used when setting them up can convince verifiers of false statements (violating *soundness* of the system), which for Zerocash translates to counterfeiting money. The authors of Zerocash write: "[D]ue to the zk-SNARK, our construction requires a one-time trusted setup of public parameters. The trust affects soundness of the proofs, though anonymity continues to hold even if the setup is corrupted by a malicious party." [BCG+14a]. The last statement is then not elaborated any further.

For ZK contingent payments (ZKCP) the parameters are generated by the buyer, which prevents the seller from cheating. However, Campanelli, Gennaro, Goldfeder and Nizzardo [CGGN17] recently showed that the *buyer* can cheat in the reference implementation of ZKCP, which allows for selling the solution to a Sudoku puzzle. By maliciously setting up the parameters, the buyer can learn information about the solution from the SNARK proof sent by the seller before paying. This shows that not only soundness but also zero knowledge of SNARKs breaks down in the face of parameter subversion.

In this work we look at whether zero knowledge can be salvaged when the parameters are set up maliciously and analyze the most efficient SNARK constructions in the literature, including the one [BCTV14] that underlies Zcash and ZKCP. We base our analyses on the theoretical framework introduced by Bellare et al. [BFS16], who formalized the notion of *subversion zero knowledge*.

Zero-knowledge proofs. A zero-knowledge proof [GMR89] is a protocol between a prover and a verifier that allows the former to convince the latter of the validity of a statement without revealing anything else. ZK proofs are an important building block for cryptographic schemes as they allow to assert that computations were done correctly while respecting the user's privacy. The three main properties of a ZK proof system are that a proof for a valid statement computed according to the protocol should convince a verifier (completeness); but there is no way that a malicious prover can convince a verifier of false statements (soundness); and nothing but the truth of the statement is revealed (zero knowledge).

In *non-interactive* ZK proofs [BFM88], the prover only sends one message (the proof) to the verifier. NIZK systems rely on a *common reference string* (CRS) to which both prover and verifier have access and which must be set up in a trusted way (for SNARKs the CRS is often called *parameters*). Without such a CRS, NIZK systems are not possible [GO94].

NIZK proof systems exist for every NP language [BFM88, BDMP91]. A language L is an NP language if it can be defined via a polynomial-time computable relation R: a statement x is in L iff there exists a *witness* w of length polynomial in the length of x such that $R(x, w) = \mathsf{true}$. In verifiable computation a server's private input would be a witness. For ZK contingent payments, the ciphertext c, the hash value y and the Sudoku challenge are the statement. The witness is the plaintext of c (the Sudoku solution) and the encryption key k.

Zero knowledge is formalized via a *simulator* that generates a CRS in which it can embed a *trapdoor*. The trapdoor must allow the simulator to produce proofs without a witness for the proven statement. ZK requires that there exists a simulator whose simulated CRSs and proofs are computationally indistinguishable from real ones. If both types are distributed equivalently then we have *perfect* ZK. Groth, Ostrovsky and Sahai [GOS06b, GOS06a, Gro06, GS08] constructed NIZK proof systems based on groups equipped with a *pairing*, i.e., an efficiently computable bilinear map. They gave the first perfect ZK system for all NP languages and very efficient schemes for specific languages based on standard cryptographic hardness assumptions.

SNARKs. Another line of work considered the size of proofs from a theoretical point of view, leading to schemes with a proof size that is sublinear in the length of the proved statement [Mic00]. SNARGs are succinct non-interactive arguments, where *succinct* means that the proof length only depends (polynomially) on the security parameter. They are *arguments* (as opposed to proofs) because soundness only holds against efficient provers. This is the best achievable notion, since SNARGs are perfect-ZK, which implies that every CRS has a trapdoor. SNAR**K**s are succinct non-interactive arguments *of knowledge*, for which a valid proofs implies that the prover knows the witness.

The first NIZK system with proofs whose size is independent of the proven statement (and its witness) was given by Groth [Gro10] using bilinear groups; it was later improved by Lipmaa [Lip12]. Gennaro, Gentry, Parno and Raykova [GGPR13] introduced the notion of a quadratic span program (QSP), showed how to efficiently convert any boolean circuit into a QSP and then constructed a SNARK system for QSPs whose proofs consist of 8 elements of a bilinear group. They gave another construction based on quadratic arithmetic programs (QAP), which represent *arithmetic* circuits, whose inputs are elements from a finite field $\mathbb{F}$ and whose gates add or multiply $\mathbb{F}$ elements. QAPs are preferred in practice due to their greater efficiency. As circuit satisfiability is NP-complete, SNARKs exist for all NP languages.

Parno, Howell, Gentry and Raykova [PHGR13] improved on [GGPR13], making the conversion from circuits to QAPs more efficient and reducing the proof size by one group element. They implemented their scheme and named it "Pinocchio". Ben-Sasson et al. [BCG$^+$13, BCTV14] improve the conversion of actual program code to QAPs, reduce the size of SNARK parameters and implement their results as *libsnark* [BCG$^+$14b]. The size of SNARK proofs for boolean circuits was then further reduced by Danezis, Fournet, Groth and Kohlweiss [DFGK14], who modified QSP to *square* span programs (SSP) and built a system for them whose proofs consist of only 4 group elements.

Recently, Groth [Gro16] presented the most efficient SNARK construction to date, which is for arithmetic circuits and whose proofs consist of only 3 group elements (and require 3 pairings to verify). All previous bilinear-group-based SNARKs are proven under strong cryptographic assumptions (*knowledge* assumptions), for which there is evidence that they might be unavoidable [GW11, BCCT12]. Starting from Bitansky et al.'s [BCI$^+$13] *linear interactive proof* framework, Groth [Gro16] achieves his result by proving security directly in the generic-group model [Sho97] (which implies all previously considered assumptions).

He also shows that SNARKs over asymmetric bilinear groups must contain elements from both source groups, meaning that the proof size of his construction is only one element short of the optimal size. Recently, Fuchsbauer, Kiltz and Loss [FKL17] proved Groth's scheme secure under a "q-type" variant of the discrete log assumption in the *algebraic group model*. In this model adversaries can only output group elements if they were obtained by applying the group operation to previously received group elements.

Subversion-resistance. The Snowden revelations documented the NSA's efforts to subvert standards, for which an illustrative example is the NSA-designed and ISO-standardized *Dual EC* random number generator. Its parameters include two elliptic-curve points, whose respective discrete logarithms can act as a backdoor that can be exploited to break TLS [CNE+14]. NIZK systems are particularly prone to parameter subversion, since their CRS must be subvertible *by design*: zero knowledge requires that an honest CRS is indistinguishable from a backdoored CRS, where the backdoor is the trapdoor used to simulate proofs. For SNARKs the parameters always contain a backdoor and anyone knowing it can simulate proofs for false statements, which means breaking soundness.

Motivated by this, Bellare, Fuchsbauer and Scafuro [BFS16] ask what security can be maintained for NIZKs when its trusted parameters are subverted. They formalize different notions of resistance to CRS subversion and investigate their achievability. They define *subversion soundness* (S-SND), meaning that no adversary can generate a (malicious) CRS together with a valid proof π for a false statement x.

They also give a subversion-resistant analogue for zero knowledge. Recall that ZK assumes that there exists a CRS simulator $\mathsf{Sim.crs}$, which returns a simulated CRS crs' and an associated simulation trapdoor td, and a proof simulator $\mathsf{Sim.pf}$ that outputs proofs on input a valid instance x and td, such that no efficient adversary can distinguish the following: being given crs' and an oracle implementing $\mathsf{Sim.pf}$, or an honest crs and an oracle returning honestly computed proofs. Subversion ZK (S-ZK) requires that for any adversary X creating a malicious CRS crs in any way it likes using randomness (coins) r, there exists a simulator $\mathsf{Sim_X.crs}$ returning a simulated CRS crs' with trapdoor td *together with simulated coins* r', as well as a proof simulator $\mathsf{Sim_X.pf}$, such that no adversary can distinguish the following: being given crs' *and* r' and a $\mathsf{Sim_X.pf}$ oracle, or a crs output by X, together with the used coins r and an honest proof oracle. The authors also define a subversion-resistant notion (S-WI) of witness-indistinguishability [FLS90] (see Sections 2.3 and 2.4).

Following [GO94], Bellare et al. [BFS16] first show that S-SND cannot be achieved together with (standard) ZK for non-trivial languages (for trivial ones the verifier needs no proof to check validity of statements). This is because ZK allows breaking soundness by subverting the CRS. They then show that S-SND can be achieved together with S-WI. Their main result is a construction that achieves both S-ZK (and thus S-WI) and SND.

BFS's S-ZK scheme. To achieve S-ZK, a simulator must be able to simulate proofs under a CRS output by a subvertor, so it cannot simply embed a trapdoor as in standard ZK. Bellare et al. [BFS16] base S-ZK on a knowledge assumption, which is the type of assumption on which security (in particular, knowledge soundness) of SNARKs relies. It states that an algorithm can only produce an output of a certain form if it knows some underlying information. This is formalized by requiring the existence of an extractor that extracts this information from the algorithm. In their scheme this information acts as the simulation trapdoor, which under their knowledge assumption can be obtained from a subvertor outputting a CRS.

Concretely, they assume that for a bilinear group $(\mathbb{G}, +)$ with a generator P any algorithm that outputs a *Diffie-Hellman* tuple (P, s_1P, s_2P, s_1s_2P) for some s_1, s_2, must know *either* s_1 *or* s_2. They call their assumption *Diffie-Hellman knowledge-of-exponent assumption* (DH-KEA) and note that a tuple (P, S_1, S_2, S_3) of this form can be verified via a (symmetric) bilinear map $\mathbf{e}$ by checking $\mathbf{e}(S_3, P) = \mathbf{e}(S_1, S_2)$. A question that arises is: who chooses the group $\mathbb{G}$ in their scheme? Bellare et al. address this by making the group $\mathbb{G}$ part of the scheme specification. This begs the question whether the subversion risk has not simply been shifted from the CRS to the choice of the group. They argue that the group generation algorithm is deterministic and public, so users can create the group themselves, and it is thus *reproducible*, whereas the CRS is inherently not.

Parameter setup in practice. A way to avoid the problem of generating a trusted CRS for NIZK systems is by proving its security in the *random-oracle model* (ROM) [BR93]. Instead of a CRS, all parties are assumed to have access to a truly random function (which is modeled as an oracle returning random values). In practice the random oracle is replaced by a cryptographic hash function and a proof in the ROM can be viewed as a security heuristic for the resulting scheme.

For NIZK systems whose CRS is a uniform random string, e.g. PCP-based constructions like [BSBC+17] recently, one can in practice set the CRS to a common random-looking public value such as the digits of π or the output of a standardized hash function on a fixed input. This intuitively guarantees that no one has embedded a trapdoor. For the Groth-Sahai proof system [GS08] the CRS consists of random elements of an elliptic-curve group; they can be set up by hashing a common random string directly into the elliptic curve [BF01, BCI+10].

For practical SNARKs the situation is different: there are no CRS-less constructions in the random-oracle model and the CRS is highly structured. The parameters typically contain elements of the form $(P, \tau P, \tau^2 P)$, where P is a generator of a group $\mathbb{G}$ and τ is a random value. Soundness completely breaks down if the value τ is known to anyone. Unfortunately, there is no known way of creating such a triple obliviously, that is, without knowing the value τ.

Our techniques. In order to show subversion zero knowledge of SNARK schemes, we assume that computing elements $(P, \tau P, \tau^2 P)$ cannot be done without knowing τ. (Looking ahead, we actually make a weaker assumption in asymmetric bilinear groups by requiring the adversary to return $(P_1, \tau P_1, \tau^2 P_1) \in \mathbb{G}_1^3$ as well as $(P_2, \tau P_2) \in \mathbb{G}_2^2$, which makes the structure of the triple verifiable using the bilinear map.) Under this assumption, which we call square knowledge of exponent (SKE) assumption (Definition 2.14), we then prove subversion ZK of five relevant SNARK constructions from the literature or slight variants of them.

As an additional sanity check, we prove that SKE holds in the generic group model (Theorem 2.16). Following Groth [Gro16], we assume that the bilinear group description is part of the specification of the language for which the proof system is defined (and not part of the CRS as in [BFS16]). Following his previous work [DFGK14], we let the CRS generation algorithm sample *random* group generators (in contrast to [BFS16], which assumes a fixed group generator). This intuitively leads to weaker assumptions required to prove soundness.

To show subversion zero knowledge of existing SNARK schemes, we proceed as follows. Standard zero knowledge holds because the randomness used to compute the CRS allows the simulator to produce proofs that are distributed equivalently to honestly generated proofs under the (honestly computed) CRS. However, for S-ZK this must hold even for a CRS that was computed in any arbitrary way. While we cannot guarantee that the CRS subvertor used random values when computing the CRS, we first show how to verify that the *structure* of the CRS is as prescribed. (For the asymmetric Pinocchio scheme [BCTV14] this requires us to extend the CRS slightly.)

Another difference between standard and subversion ZK is that in the former the simulator creates the CRS and thus knows the simulation trapdoor, whereas for S-ZK the CRS is produced by the subvertor, so it might not be clear how proofs can be simulated at all. Now if the CRS contains elements $(P, \tau P, \tau^2 P)$, whose correct structure can be verified via the pairing, then under our SKE assumption we can extract the value τ. SKE thus allows the simulator to obtain parts of the randomness even from a maliciously generated CRS. Unfortunately, the simulation trapdoor typically contains *other* values that the S-ZK simulator cannot extract.

Our next step is then to demonstrate that proofs can be simulated using τ only, or to show how under our assumption more values can be extracted that then enable simulation. Our final step is to show that if a CRS passes the verification procedure we define, then proofs that were simulated using the partial trapdoor are distributed like real proofs. This shows that the analyzed scheme is S-ZK under our SKE assumption. While knowledge assumptions are strong assumptions, they seem unavoidable since S-ZK implies 2-move interactive ZK by letting the verifier create the CRS. And such schemes require extractability assumptions [BCPR14].

Since simulated proofs are by definition independent of a witness, our results imply that under a verified, but possibly malicious, CRS, proofs for different witnesses are equally distributed. As a corollary we thereby obtain that all SNARKs we consider satisfy subversion witness indistinguishability *unconditionally* (i.e., no assumptions required).

We note that Ben-Sasson et al. [BCG+15] also consider making a CRS verifiable. Their goal is to protect *soundness* against subversion by sampling the secret values underlying a CRS in a distributed way. Only if all participants in the CRS-creation protocol collude can they break soundness. To guarantee a correctly distributed CRS, the participant(s) must prove adherence to the protocol via NIZK proofs [Sch91, FS87] secure in the random-oracle model. The protocol thus returns *verifiable* SNARK parameters. The parameters used for Zcash were set up using this multiparty protocol, which was recently detailed by Bowe, Gabizon and Green [BGG17].

Our Results

As already discussed, SNARKs are not subversion-sound because their CRS contains the simulation trapdoor. In this work we look at subversion resistance of their zero-knowledge property and investigate several SNARK constructions from the literature that are based on bilinear groups. In particular,

1. the first QSP-based and 2. QAP-based constructions [GGPR13];
3. optimized Pinocchio [BCTV14] as implemented in libsnark [BCG+14b]; and
4. and 5. the two most efficient (SSP- and QAP-based) constructions by Groth et al. [DFGK14, Gro16].

We make the (reasonable) assumption that a privacy-conscious prover (whose protection is the goal of zero knowledge) first checks whether the CRS looks plausible (to whatever extent this is possible) before publishing a proof with respect to it. All of our results implicitly make this assumption.

We start with the first SNARK construction for QAPs by Gennaro, Gentry, Parno and Raykova [GGPR13] and show how to verify that the CRS is correctly formed. We then show that under the square knowledge of exponent (SKE) assumption their construction satisfies subversion zero knowledge as defined in [BFS16]. The same holds for their QSP-based SNARK.

We next turn to the optimized version of Pinocchio over asymmetric bilinear groups due to Ben-Sasson, Chiesa, Tromer and Virza [BCTV14]. For this construction we show that adding 4 group elements to the CRS makes it efficiently checkable. We then prove that the scheme with this slightly extended CRS satisfies subversion zero knowledge under SKE, whereas the original scheme, which is implemented in libsnark [BCG+14b], succumbs to a parameter-subversion attack [CGGN17]. For the SNARK by Danezis, Fournet, Groth and Kohlweiss [DFGK14], we show that CRS well-formedness can be efficiently verified without modifying the CRS and that S-ZK holds analogously to Pinocchio.

Finally, we consider the most efficient SNARK scheme by Groth [Gro16] and again show that the scheme is *already* subversion-zero-knowledge under SKE. Proving this is more involved than for the previous schemes, since the value τ, for which $P, \tau P, \tau^2 P, \ldots$ are contained in the CRS does not suffice to simulate proofs, as for the previous schemes. We show that, using SKE twice, another value can be extracted, which together with τ then enables proof simulation. As corollaries, we get that S-WI holds unconditionally for all considered schemes.

CONCURRENT WORK. Campanelli, Gennaro, Goldfeder and Nizzardo [CGGN17] show that Pinocchio as implemented in libsnark [BCG+14b] is not subversion-zero-knowledge by exhibiting an attack. As countermeasures they propose to instead use one of the older SNARKs by Gennaro et al. [GGPR13], as they allow verification of CRS well-formedness, which yields witness indistinguishability. They admit that for applications for which there is only *one* witness, like selling a Sudoku solution, WI is vacuous (as any protocol satisfies WI).

They refer to Bellare et al.'s [BFS16] S-ZK system and conjecture that "the techniques extend to the original QSP/QAP protocol in [GGPR13]" (which we proved rigorously). Moreover, "[i]t is however not clear if those techniques extend to Pinocchio" and "it would require major changes in the current implementation of ZKCP protocols". (We show that it suffices to add 4 group elements to the CRS and perform the checks of well-formedness.) They recommend following the Zcash approach [BCG+15, BGG17] and using an interactive protocol that lets the prover and verifier compute the CRS together.

In other concurrent work Abdolmaleki, Baghery, Lipmaa and Zajac [ABLZ17] present a S-ZK variant of Groth's SNARK [Gro16]. They need to modify the scheme, thereby reducing efficiency, and they prove their result under a stronger assumption. In particular, they extend the CRS by $2d$ group elements (where d is the number of multiplication gates in the circuit that represents the relation). Their assumption states that any adversary that for generators $P_1 \in \mathbb{G}_1^*$ and $P_2 \in \mathbb{G}_2^*$ outputs a pair of the form (sP_1, sP_2) must know s. As they note, their assumption is false in groups with a symmetric ("Type-1") bilinear map as well as in asymmetric groups of Type 2, whereas our SKE assumption holds generically in all bilinear group settings. They claim security of their scheme under their own definition of S-ZK, which is a statistical notion, in contrast to original computational S-ZK notion [BFS16], which we consider.[1]

PRACTICAL IMPLICATIONS OF OUR RESULTS. We show that for all analyzed schemes except asymmetric Pinocchio, it suffices to verify the parameters once in order to guarantee subversion zero knowledge. Any already deployed parameters can thus be continued to be used after verification. Subversion-ZK of Pinocchio can be obtained by adding 4 group elements to the CRS.

For Pinocchio-based ZK contingent payments this means that the scheme can be made secure by slightly augmenting the size of the parameters and having the seller verify them. No additional interaction between seller and buyer (as recommended by Campanelli et al. [CGGN17]) is thus required. Of course, admitting additional interaction could lead to more efficient schemes than using the (costly) CRS verification.

The SNARK parameters used in Zcash have been computed by running the multi-party protocol from [BCG+15, BGG17] and verifiability of this process is achieved via random-oracle NIZK proofs. Let us define a CRS subvertor that runs this protocol, playing the roles of all parties, and outputs the resulting CRS which includes the ROM proofs. Since the latter guarantee well-formedness of the CRS, under SKE there exists an efficient extractor that can extract the simulation trapdoor from this CRS subvertor. Using the trapdoor, proofs can be simulated (as specified in Section 5). We thus conclude that, assuming users verify the consistency of the CRS, Zcash provides a subversion-resistant form of anonymity in the random-oracle model under the SKE assumption with respect to the bilinear group used by Zcash. Thus, even if all parties involved in creating the parameters were malicious, Zcash is still anonymous.

We content ourselves with the above argument, as a formal proof would be beyond the scope

[1] It is not clear how their scheme can achieve statistical S-ZK, considering that the success of the simulator relies on a *computational* assumption. They also claim that their notion is stronger because they let the subvertor X pass "extra information" to the adversary A, whereas A "only" receives X's coins r in [BFS16]. But A can itself compute any such information from r.

of this paper. Subsequently to our results Bowe et al. [BGG17] proved that their protocol is S-ZK with a polynomially small (not negligible) simulation error in the random-oracle model without making knowledge assumptions.

2 Definitions

2.1 Notation

If x is a (binary) string then $|x|$ is its length. If S is a finite set then $|S|$ denotes its size and $s \leftarrow\$ S$ denotes picking an element uniformly from S and assigning it to s. We denote by $\lambda \in \mathbb{N}$ the security parameter and by 1^λ its unary representation.

Algorithms are randomized unless otherwise indicated. "PT" stands for "polynomial time", whether for randomized or deterministic algorithms. By $y \leftarrow A(x_1, \ldots; r)$ we denote the operation of running algorithm A on inputs $x_1, \ldots$ and coins r and letting y denote the output. By $y \leftarrow\$ A(x_1, \ldots)$, we denote letting $y \leftarrow A(x_1, \ldots; r)$ for random r. We denote by $[A(x_1, \ldots)]$ the set of points that have positive probability of being output by A on inputs $x_1, \ldots$

For our security definitions we use the code-based game playing framework [BR06]. A game G (e.g. Figure 1) usually depends on a scheme and executes one or more adversaries. It defines oracles for the adversaries as procedures. The game eventually returns a boolean. We let Pr[G] denote the probability that G returns true.

We recall the standard notions of soundness, knowledge-soundness, witness-indistinguishability and zero knowledge for NIZKs, which assume the CRS is trusted and then give their subversion-resistant counterparts that were introduced in [BFS16]. We mainly follow their exposition and start with the syntax.

2.2 NP Relations and NI Systems

NP relations. Consider $R\colon \{0,1\}^* \times \{0,1\}^* \to \{\mathsf{true}, \mathsf{false}\}$. For $x \in \{0,1\}^*$ we let $R(x) = \{\, w \mid R(x,w) = \mathsf{true} \,\}$ be the *witness set* of x. R is an **NP** relation if it is PT and there is a polynomial P_R such that every w in $R(x)$ has length at most $P_R(|x|)$ for all x. We let $L(R) = \{\, x \mid R(x) \neq \emptyset \,\}$ be the *language* associated to R. We will consider relations output by a PT *relation generator* Rg (which may also output some auxiliary information z that is given to the adversary). We assume λ can be deduced from $R \in [\mathsf{Rg}(1^\lambda)]$ and note that definitions from [BFS16], which are for one *fixed* relation R, are easily recovered by defining $\mathsf{Rg}(1^\lambda) := (1^\lambda, R)$.

NI systems. A non-interactive (NI) system Π for relation generator Rg specifies the following PT algorithms. Via $crs \leftarrow\$ \Pi.\mathsf{Pg}(R)$ one generates a common reference string crs. Via $\pi \leftarrow\$ \Pi.\mathsf{P}(R, crs, x, w)$ the honest prover, given x and $w \in R(x)$, generates a proof π that $x \in L(R)$. Via $d \leftarrow \Pi.\mathsf{V}(R, crs, x, \pi)$ a verifier can produce a decision $d \in \{\mathsf{true}, \mathsf{false}\}$ indicating whether π is a valid proof that $x \in L(R)$. We require (perfect) completeness, that is, for all $\lambda \in \mathbb{N}$, all $R \in [\mathsf{Rg}(1^\lambda)]$, all $crs \in [\Pi.\mathsf{Pg}(R)]$, all $x \in L(R)$, all $w \in R(x)$ and all $\pi \in [\Pi.\mathsf{P}(R, crs, x, w)]$ we have $\Pi.\mathsf{V}(R, crs, x, \pi) = \mathsf{true}$. We also assume that $\Pi.\mathsf{V}$ returns false if any of its arguments is $\bot$.

2.3 Standard Notions: SND, KSND, WI and ZK

Soundness. Soundness means that it is hard to create a valid proof for any $x \notin L(R)$. We consider computational soundness as opposed to a statistical one, which is usually sufficient for applications, and which is the notion achieved by SNARGs.

Definition 2.1 (SND) *An NI system* Π *for relation generator* Rg *is* sound *if* $\mathbf{Adv}^{\mathrm{snd}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\cdot)$ *is negligible for all PT adversaries* A*, where* $\mathbf{Adv}^{\mathrm{snd}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda) = \Pr[\mathrm{SND}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda)]$ *and game* SND *is specified in Figure 1.*

<u>Knowledge soundness.</u> This strengthening of soundness [BG93] means that a prover that outputs a valid proof must know the witness. Formally, there exists an extractor that can extract the witness from the prover. The notion implies soundness, since for a proof of a wrong statement there exists no witness.

Definition 2.2 (KSND) *An NI system* Π *for relation generator* Rg *is* knowledge-sound *if for all PT adversaries* A *there exists a PT extractor* E *such that* $\mathbf{Adv}^{\mathrm{ksnd}}_{\Pi,\mathsf{Rg},\mathsf{A},\mathsf{E}}(\cdot)$ *is negligible, where* $\mathbf{Adv}^{\mathrm{ksnd}}_{\Pi,\mathsf{Rg},\mathsf{A},\mathsf{E}}(\lambda) = \Pr[\mathrm{KSND}_{\Pi,\mathsf{Rg},\mathsf{A},\mathsf{E}}(\lambda)]$ *and game* KSND *is specified in Figure 1.*

Note that (as for the following two notions) the output of game KSND is *efficiently computable*, which is not the case for SND, since membership in $L(R)$ may not be efficiently decidable. This can be an issue when proving security of more complex systems that use a system Π as a building block.

<u>WI.</u> Witness-indistinguishability [FLS90] requires that proofs for the same statement using different witnesses are indistinguishable. The adversary can adaptively request multiple proofs for statements x under one of two witnesses w_0, w_1; it receives proofs under w_b for a challenge bit b which it must guess.

Definition 2.3 (WI) *An NI system* Π *for* Rg *is* witness-indistinguishable *if* $\mathbf{Adv}^{\mathrm{wi}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\cdot)$ *is negligible for all PT adversaries* A*, where* $\mathbf{Adv}^{\mathrm{wi}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda) = 2\Pr[\mathrm{WI}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda)] - 1$ *and game* WI *is specified in Figure 1.*

<u>ZK.</u> Zero knowledge [GMR89] means that no information apart from the fact that $x \in L(R)$ is leaked by the proof. It is formalized by requiring that a simulator, who can create the CRS, can compute proofs without being given a witness, so that CRS and proofs are indistinguishable from real ones. In particular, the distinguisher A can adaptively request proofs by supplying an instance and a valid witness for it. The proof is produced either by the honest prover using the witness, or by the simulator. The adversary outputs a guess b' as to whether the proofs were real or simulated.

Definition 2.4 (ZK) *An NI system* Π *for* Rg *is* zero-knowledge *if* Π *specifies additional PT algorithms* $\Pi.\mathsf{Sim.crs}$ *and* $\Pi.\mathsf{Sim.pf}$ *such that* $\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\cdot)$ *is negligible for all PT adversaries* A*, where* $\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda) = 2\Pr[\mathrm{ZK}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda)] - 1$ *and game* ZK *is specified in Figure 1.*

An NI system Π is *statistical* zero-knowledge if the above holds for all (not necessarily PT) adversaries A. It is *perfect* zero-knowledge if $\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\cdot) \equiv 0$.

2.4 Notions for Subverted CRS: S-SND, S-KSND, S-WI and S-ZK

For all notions considered in the previous section the CRS is assumed to be honestly generated. Bellare et al. [BFS16] ask what happens when the CRS is maliciously generated and define subversion-resistant analogues S-SND, S-WI and S-ZK, in which the adversary chooses the CRS. The following three definitions are from [BFS16].

<u>Subversion soundness.</u> Subversion soundness asks that if the adversary creates a CRS in any way it likes, it is still unable to prove false statements under it. We accordingly modify the soundness game SND by letting the adversary choose *crs* in addition to x and π.

Game $\mathrm{SND}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda)$ $R \leftarrow_{\$} \mathsf{Rg}(1^\lambda)$ $crs \leftarrow_{\$} \Pi.\mathsf{Pg}(R)$ $(x, \pi) \leftarrow_{\$} \mathsf{A}(R, crs)$ Return $\big(x \notin L(R)$ and $\Pi.\mathsf{V}(R, crs, x, \pi)\big)$	Game $\text{S-SND}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda)$ $R \leftarrow_{\$} \mathsf{Rg}(R)$ $(crs, x, \pi) \leftarrow_{\$} \mathsf{A}(R)$ Return $\big(x \notin L(R)$ and $\Pi.\mathsf{V}(R, crs, x, \pi)\big)$
Game $\mathrm{KSND}_{\Pi,\mathsf{Rg},\mathsf{A},\mathsf{E}}(\lambda)$ $R \leftarrow_{\$} \mathsf{Rg}(1^\lambda)$ $crs \leftarrow_{\$} \Pi.\mathsf{Pg}(R)$; $r \leftarrow_{\$} \{0,1\}^{\mathsf{A.rl}(\lambda)}$ $(x, \pi) \leftarrow \mathsf{A}(R, crs; r)$ $w \leftarrow_{\$} \mathsf{E}(R, crs, r)$ Return $\big(R(x, w) = \mathsf{false}$ and $\Pi.\mathsf{V}(R, crs, x, \pi)\big)$	Game $\text{S-KSND}_{\Pi,\mathsf{Rg},\mathsf{A},\mathsf{E}}(\lambda)$ $R \leftarrow_{\$} \mathsf{Rg}(1^\lambda)$ $r \leftarrow_{\$} \{0,1\}^{\mathsf{A.rl}(\lambda)}$ $(crs, x, \pi) \leftarrow \mathsf{A}(R; r)$ $w \leftarrow_{\$} \mathsf{E}(R, r)$ Return $\big(R(x, w) = \mathsf{false}$ and $\Pi.\mathsf{V}(R, crs, x, \pi)\big)$
Game $\mathrm{WI}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda)$ $b \leftarrow_{\$} \{0,1\}$; $R \leftarrow_{\$} \mathsf{Rg}(1^\lambda)$ $crs \leftarrow_{\$} \Pi.\mathsf{Pg}(R)$ $b' \leftarrow_{\$} \mathsf{A}^{\text{Prove}}(R, crs)$ Return $(b = b')$ $\text{Prove}(x, w_0, w_1)$ If $R(x, w_0) = \mathsf{false}$ or $R(x, w_1) = \mathsf{false}$ then return $\perp$ $\pi \leftarrow_{\$} \Pi.\mathsf{P}(R, crs, x, w_b)$ Return π	Game $\text{S-WI}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda)$ $b \leftarrow_{\$} \{0,1\}$; $R \leftarrow_{\$} \mathsf{Rg}(1^\lambda)$ $(crs, st) \leftarrow_{\$} \mathsf{A}(R)$ $b' \leftarrow_{\$} \mathsf{A}^{\text{Prove}}(R, crs, st)$ Return $(b = b')$ $\text{Prove}(x, w_0, w_1)$ If $R(x, w_0) = \mathsf{false}$ or $R(x, w_1) = \mathsf{false}$ then return $\perp$ $\pi \leftarrow_{\$} \Pi.\mathsf{P}(R, crs, x, w_b)$ Return π
Game $\mathrm{ZK}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda)$ $b \leftarrow_{\$} \{0,1\}$; $R \leftarrow_{\$} \mathsf{Rg}(1^\lambda)$ $crs_1 \leftarrow_{\$} \Pi.\mathsf{Pg}(R)$ $(crs_0, td) \leftarrow_{\$} \Pi.\mathsf{Sim.crs}(R)$ $b' \leftarrow_{\$} \mathsf{A}^{\text{Prove}}(R, crs_b)$ Return $(b = b')$ $\text{Prove}(x, w)$ If $R(x, w) = \mathsf{false}$ then return $\perp$ If $b = 1$ then $\pi \leftarrow_{\$} \Pi.\mathsf{P}(R, crs_1, x, w)$ Else $\pi \leftarrow_{\$} \Pi.\mathsf{Sim.pf}(R, crs_0, td, x)$ Return π	Game $\text{S-ZK}_{\Pi,\mathsf{Rg},\mathsf{X},\mathsf{S},\mathsf{A}}(\lambda)$ $b \leftarrow_{\$} \{0,1\}$; $R \leftarrow_{\$} \mathsf{Rg}(1^\lambda)$ $r_1 \leftarrow_{\$} \{0,1\}^{\mathsf{X.rl}(\lambda)}$; $crs_1 \leftarrow \mathsf{X}(R; r_1)$ $(crs_0, r_0, td) \leftarrow_{\$} \mathsf{S.crs}(R)$ $b' \leftarrow_{\$} \mathsf{A}^{\text{Prove}}(R, crs_b, r_b)$ Return $(b = b')$ $\text{Prove}(x, w)$ If $R(x, w) = \mathsf{false}$ then return $\perp$ If $b = 1$ then $\pi \leftarrow_{\$} \Pi.\mathsf{P}(R, crs_1, x, w)$ Else $\pi \leftarrow_{\$} \mathsf{S.pf}(R, crs_0, td, x)$ Return π

Figure 1: Games defining soundness, knowledge-soundness, witness-indistinguishability and zero knowledge (left) and their subversion-resistant counterparts (right) for an NI system Π.

Definition 2.5 (S-SND) *An NI system* Π *for generator* Rg *is* subversion-sound *if* $\mathbf{Adv}^{\text{s-snd}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\cdot)$ *is negligible for all PT adversaries* A*, where* $\mathbf{Adv}^{\text{s-snd}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda) = \Pr[\text{S-SND}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda)]$ *and game* S-SND *is specified in Figure 1.*

Subversion WI. Subversion WI demands that even when the subvertor creates a CRS in any way it likes, it can still not decide which of two witnesses of its choice were used to create a proof. The adversary is modeled as a two-stage algorithm: it first outputs a CRS *crs* along with state information (which could e.g. contain a trapdoor associated to *crs*) passed to the second stage. The second stage is then defined like for the honest-CRS game WI, where via its Prove oracle, the adversary can adaptively query proofs for instances under one of two witnesses.

Definition 2.6 (S-WI) *An NI system* Π *for generator* Rg *is* subversion-witness-indistinguishable *if* $\mathbf{Adv}^{\text{s-wi}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\cdot)$ *is negligible for all PT adversaries* A*, where* $\mathbf{Adv}^{\text{s-wi}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda) = 2\Pr[\text{S-WI}_{\Pi,\mathsf{Rg},\mathsf{A}}(\lambda)] - 1$ *and game* S-WI *is specified in Figure 1. An NI system* Π *is* perfect *S-WI if* $\mathbf{Adv}^{\text{s-wi}}_{\Pi,\mathsf{Rg},\mathsf{A}}(\cdot) \equiv 0$*.*

Subversion ZK. This notion considers a CRS subvertor X that returns an arbitrarily formed CRS. Subversion ZK now asks that for any such X there exists a simulator that is able to simulate (1) the full view of the CRS subvertor, *including its coins*, and (2) proofs for adaptively chosen instances without knowing the witnesses. The simulator consists of $\mathsf{S.crs}$, which returns a CRS, coins for X and a trapdoor which is then used by its second stage $\mathsf{S.pf}$ to simulate proofs. The adversary's task is to decide whether it is given a real CRS and the coins used to produce it, and real proofs (case $b = 1$); or whether it is given a simulated CRS and coins, and simulated proofs (case $b = 0$).

Definition 2.7 (S-ZK) *An NI system* Π *for generator* Rg *is* subversion-zero-knowledge *if for all PT CRS subvertors* X *there exists a PT simulator* $\mathsf{S} = (\mathsf{S.crs}, \mathsf{S.pf})$ *such that for all PT adversaries* A *the function* $\mathbf{Adv}^{\text{s-zk}}_{\Pi,\mathsf{Rg},\mathsf{X},\mathsf{S},\mathsf{A}}(\cdot)$ *is negligible, where* $\mathbf{Adv}^{\text{s-zk}}_{\Pi,\mathsf{Rg},\mathsf{X},\mathsf{S},\mathsf{A}}(\lambda) = 2\Pr[\text{S-ZK}_{\Pi,\mathsf{Rg},\mathsf{X},\mathsf{S},\mathsf{A}}(\lambda)] - 1$ *and game* S-ZK *is specified in Figure 1.*

The definition is akin to zero knowledge for interactive proof systems [GMR89], when interpreting the CRS as the verifier's first message. The simulator must produce a full view of the verifier (including coins and a transcript of its interaction with the Prove oracle). On the other hand, to imply ZK of NI systems, the simulator needs to produce the CRS *before* learning the statements for which it must simulate proofs. Moreover, the simulator can depend on X but not on A.

Subversion KSND. For completeness we give a subversion-resistant analogue for knowledge soundness (not considered in [BFS16]), as this is the relevant notion for SNARKs. We modify game KSND and let the adversary choose *crs* in addition to x and π. We are not aware of any construction that achieves S-KSND and some form of WI.

Definition 2.8 (S-KSND) *An NI system* Π *for relation generator* Rg *is* subversion-knowledge-sound *if for all PT adversaries* A *there exists a PT extractor* E *such that* $\mathbf{Adv}^{\text{s-ksnd}}_{\Pi,\mathsf{Rg},\mathsf{A},\mathsf{E}}(\cdot)$ *is negligible, where* $\mathbf{Adv}^{\text{s-ksnd}}_{\Pi,\mathsf{Rg},\mathsf{A},\mathsf{E}}(\lambda) = \Pr[\text{S-KSND}_{\Pi,\mathsf{Rg},\mathsf{A},\mathsf{E}}(\lambda)]$ *and game* S-KSND *is specified in Figure 1.*

2.5 Bilinear Groups and Assumptions

Bilinear groups. The SNARK constructions we consider are based on bilinear groups, for which we introduce a new type of knowledge-of-exponent assumption. We distinguish between asymmetric and symmetric groups.

Definition 2.9 *An asymmetric-bilinear-group generator* aGen *is a PT algorithm that takes input a security parameter* 1^λ *and outputs a description of a bilinear group* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e})$ *with the following properties:*

$\underline{\text{GAME PDH}_{q,\mathsf{aGen},\mathsf{A}}(\lambda)}$

$Gr = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}) \leftarrow\!\!\$ \ \mathsf{aGen}(1^\lambda)$; $P_1 \leftarrow\!\!\$ \ \mathbb{G}_1^*$; $P_2 \leftarrow\!\!\$ \ \mathbb{G}_2^*$; $s \leftarrow\!\!\$ \ \mathbb{Z}_p^*$

$Y \leftarrow\!\!\$ \ \mathsf{A}\big(Gr, P_1, P_2, sP_1, sP_2, \ldots, s^q P_1, s^q P_2, s^{q+2}P_1, s^{q+2}P_2, \ldots, s^{2q}P_1, s^{2q}P_2\big)$

Return $(Y = s^{q+1}P_1)$

$\underline{\text{GAME TSDH}_{q,\mathsf{aGen},\mathsf{A}}(\lambda)}$

$Gr = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}) \leftarrow\!\!\$ \ \mathsf{aGen}(1^\lambda)$; $P_1 \leftarrow\!\!\$ \ \mathbb{G}_1^*$; $P_2 \leftarrow\!\!\$ \ \mathbb{G}_2^*$; $s \leftarrow\!\!\$ \ \mathbb{Z}_p^*$

$(r, Y) \leftarrow\!\!\$ \ \mathsf{A}\big(Gr, P_1, P_2, sP_1, sP_2, \ldots, s^q P_1, s^q P_2, \big)$

Return $\big(r \in \mathbb{Z}_p \setminus \{s\}$ and $Y = \mathbf{e}(P_1, P_2)^{1/(s-r)}\big)$

$\underline{\text{GAME PKE}_{q,\mathsf{aGen},\mathsf{Z},\mathsf{A},\mathsf{E}}(\lambda)}$

$Gr = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}) \leftarrow\!\!\$ \ \mathsf{aGen}(1^\lambda)$; $P_1 \leftarrow\!\!\$ \ \mathbb{G}_1^*$; $P_2 \leftarrow\!\!\$ \ \mathbb{G}_2^*$; $s \leftarrow\!\!\$ \ \mathbb{Z}_p^*$

$r \leftarrow\!\!\$ \ \{0,1\}^{\mathsf{A.rl}(\lambda)}$

$z \leftarrow\!\!\$ \ \mathsf{Z}(Gr, P_1, sP_1, \ldots, s^q P_1)$

$(V, W) \leftarrow \mathsf{A}\big(Gr, P_1, P_2, sP_1, sP_2 \ldots, s^q P_1, s^q P_2, z; r\big)$

$(a_0, \ldots, a_q) \leftarrow\!\!\$ \ \mathsf{E}\big(Gr, P_1, P_2, sP_1, sP_2, \ldots, s^q P_1, s^q P_2, z, r\big)$

Return $\big(\mathbf{e}(V, P_2) = \mathbf{e}(P_1, W)$ and $V \neq (\sum_{i=0}^q a_i s^i)P_1\big)$

Figure 2: Games defining assumptions q-PDH, q-TSDH and q-PKE

- *p is a prime of length λ;*
- *$(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ and $(\mathbb{G}_T, \cdot)$ are groups of order p;*
- *$\mathbf{e}\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map, that is, for all $a, b \in \mathbb{Z}_p$ and $S \in \mathbb{G}_1$, $T \in \mathbb{G}_2$ we have: $\mathbf{e}(aS, bT) = \mathbf{e}(S, T)^{ab}$;*
- *$\mathbf{e}$ is non-degenerate, that is, for $P_1 \in \mathbb{G}_1^*$ and $P_2 \in \mathbb{G}_2^*$ (i.e., P_1 and P_2 are generators) $\mathbf{e}(P_1, P_2)$ generates $\mathbb{G}_T$.*

Moreover, we assume that group operations and the bilinear map can be computed efficiently, membership of the groups and equality of group elements can be decided efficiently, and group generators can be sampled efficiently.

A symmetric-bilinear-group *generator* sGen *returns a bilinear group with $\mathbb{G}_1 = \mathbb{G}_2$, which we denote by $\mathbb{G}$, and with a symmetric non-degenerate bilinear map $\mathbf{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.*

<u>Assumptions.</u> We recall the assumptions under which SNARKs in the literature were proven sound. The following assumptions are from [DFGK14], who adapted PDH from [Gro10] to asymmetric groups, and TSDH from [BB04, Gen04].

Definition 2.10 (q-PDH) *The* $q(\lambda)$-power Diffie-Hellman *assumption holds for an asymmetric group generator* aGen *if* $\mathbf{Adv}^{\mathrm{pdh}}_{q,\mathsf{aGen},\mathsf{A}}(\cdot)$ *is negligible for all PT adversaries* A, *where* $\mathbf{Adv}^{\mathrm{pdh}}_{q,\mathsf{aGen},\mathsf{A}}(\lambda) = \Pr[\mathrm{PDH}_{q,\mathsf{aGen},\mathsf{A}}(\lambda)]$ *and* PDH *is defined in Figure 2.*

The q-PDH assumption for symmetric group generators sGen is defined analogously by letting $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2$ (A thus only receives $2q$ group elements).

Definition 2.11 (q-TSDH) *The* $q(\lambda)$-target-group strong Diffie-Hellman *assumption holds for an asymmetric group generator* aGen *if* $\mathbf{Adv}^{\mathrm{tsdh}}_{q,\mathsf{aGen},\mathsf{A}}(\cdot)$ *is negligible for all PT adversaries* A*, where* $\mathbf{Adv}^{\mathrm{tsdh}}_{q,\mathsf{aGen},\mathsf{A}}(\lambda) = \Pr[\mathrm{TSDH}_{q,\mathsf{aGen},\mathsf{A}}(\lambda)]$ *and* TSDH *is defined in Figure 2.*

The q-TSDH assumption for symmetric group generators sGen is defined analogously by letting $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2$ (A thus only receives $q+1$ group elements).

KEA. The knowledge-of-exponent assumption [Dam92, HT98, BP04] in a group $\mathbb{G}$ states that an algorithm A that is given two random generators $P, Q \in \mathbb{G}^*$ and outputs (cP, cQ) must know c. This is formalized by requiring that there exists an extractor for A which given A's coins outputs c. This has been considered in the bilinear-group setting [AF07] where A's output (cP, cQ) can be verified by using the bilinear map. Generalizations of KEA were made by Groth [Gro10], who assumes that for every A that on input $(P, Q, sP, sQ, s^2P, s^2Q, \ldots, s^qP, s^qQ)$ returns (cP, cQ) an extractor can extract $(a_0, \ldots, a_q)$ such that $c = \sum_{i=0}^{q} a_i s^i$. Danezis et al. [DFGK14] port Groth's assumption to asymmetric groups as follows.

Definition 2.12 (q-PKE) *The* $q(\lambda)$-power knowledge of exponent *assumption holds for* aGen *w.r.t. the class* $\mathcal{A}ux$ *of auxiliary input generators if for every PT* $\mathsf{Z} \in \mathcal{A}ux$ *and PT adversary* A *there exists a PT extractor* E *s.t.* $\mathbf{Adv}^{\mathrm{pke}}_{q,\mathsf{aGen},\mathsf{Z},\mathsf{A},\mathsf{E}}(\cdot)$ *is negligible, where* $\mathbf{Adv}^{\mathrm{pke}}_{q,\mathsf{aGen},\mathsf{Z},\mathsf{A},\mathsf{E}}(\lambda) = \Pr[\mathrm{PKE}_{q,\mathsf{aGen},\mathsf{Z},\mathsf{A},\mathsf{E}}(\lambda)]$ *and* PKE *is defined in Figure 2.*

The q-PKE assumption for symmetric generators sGen is defined by letting $\mathbb{G}_1 = \mathbb{G}_2$ but *again choosing* $P_1, P_2 \leftarrow_{\$} \mathbb{G}^*$ (A thus again receives $2q+2$ group elements).

Bellare et al. [BFS16] consider deterministically generated groups (whereas for SNARK systems the group will be part of the relation R output by a relation generator Rg). They therefore need to define all other assumptions, such as DLin [BBS04], with respect to this fixed group. BFS introduce a new type of KEA, called DH-KEA, which assumes that if A outputs a Diffie-Hellman (DH) tuple (sP, tP, stP) w.r.t. the fixed P, then A must know either s or t. The auxiliary input given to A are two additional random generators H_0, H_1. The intuition is that while an adversary may produce one group element without knowing its discrete logarithm by hashing into the elliptic curve [BF01, SvdW06, BCI+10], it seems hard to produce a DH tuple without knowing at least one of the logarithms.

Definition 2.13 (DH-KEA) *Let* $\mathsf{detSGen}$ *be a deterministic group generator. The* Diffie-Hellman knowledge of exponent *assumption holds for* $\mathsf{detSGen}$ *if for every PT* A *there exists a PT* E *s.t.* $\mathbf{Adv}^{\mathrm{dhke}}_{\mathsf{detSGen},\mathsf{A},\mathsf{E}}(\cdot)$ *is negligible, where* $\mathbf{Adv}^{\mathrm{dhke}}_{\mathsf{detSGen},\mathsf{A},\mathsf{E}}(\lambda) = \Pr[\mathrm{DHKE}_{\mathsf{detSGen},\mathsf{A},\mathsf{E}}(\lambda)]$ *and* DHKE *defined in Figure 3.*

SKE. We now consider a weakening of DH-KEA where we prescribe $s = t$; that is, if A on input P outputs a pair (sP, s^2P) then E extracts s. This assumption is weaker than (i.e., implied by) DH-KEA. As we consider groups with randomly sampled generators, we let A choose the generator P itself and assume that there exists an extractor that extracts s when A outputs a tuple (P, sP, s^2P). This allows us to choose a random generator when setting up parameters of a scheme. The security of such schemes then follows from assumptions such as PDH, as defined above, where the generators are chosen randomly.

Definition 2.14 (SKE) *Let* sGen *be a symmetric-group generator. The* square knowledge of exponent *assumption holds for* sGen *if for every PT* A *there exists a PT* E *s.t.* $\mathbf{Adv}^{\mathrm{ske}}_{\mathsf{sGen},\mathsf{A},\mathsf{E}}(\cdot)$ *is negligible, where* $\mathbf{Adv}^{\mathrm{ske}}_{\mathsf{sGen},\mathsf{A},\mathsf{E}}(\lambda) = \Pr[\mathrm{SKE}_{\mathsf{sGen},\mathsf{A},\mathsf{E}}(\lambda)]$ *with* SKE *is defined in Figure 3.*

GAME $\mathrm{DHKE}_{\mathsf{detSGen},\mathsf{A},\mathsf{E}}(\lambda)$

$(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, P) \leftarrow \mathsf{detSGen}(1^\lambda)$; $H_0, H_1 \leftarrow\$ \mathbb{G}$; $r \leftarrow\$ \{0,1\}^{\mathsf{A.rl}(\lambda)}$

$(S_0, S_1, S_2) \leftarrow \mathsf{A}(1^\lambda, H_0, H_1; r)$; $s \leftarrow\$ \mathsf{E}(1^\lambda, H_0, H_1, r)$

Return $\big(\mathbf{e}(S_0, S_1) = \mathbf{e}(P, S_2)$ and $sP \neq S_0$ and $sP \neq S_1\big)$

GAME $\mathrm{SKE}_{\mathsf{sGen},\mathsf{A},\mathsf{E}}(\lambda)$ (for symmetric groups)

$Gr = (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow\$ \mathsf{sGen}(1^\lambda)$; $r \leftarrow\$ \{0,1\}^{\mathsf{A.rl}(\lambda)}$

$(S_0, S_1, S_2) \leftarrow \mathsf{A}(Gr; r)$

$s \leftarrow\$ \mathsf{E}(Gr, r)$

Return $\big(\mathbf{e}(S_1, S_1) = \mathbf{e}(S_0, S_2)$ and $sS_0 \neq S_1\big)$

GAME $\mathrm{SKE}_{\mathsf{aGen},\mathsf{A},\mathsf{E}}(\lambda)$ (for asymmetric groups)

$Gr = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}) \leftarrow\$ \mathsf{aGen}(1^\lambda)$; $r \leftarrow\$ \{0,1\}^{\mathsf{A.rl}(\lambda)}$

$(S_0, S_1, S_2, T_0, T_1) \leftarrow \mathsf{A}(Gr; r)$

$s \leftarrow\$ \mathsf{E}(Gr, r)$

Return $\big(\mathbf{e}(S_1, T_0) = \mathbf{e}(S_0, T_1)$ and $\mathbf{e}(S_1, T_1) = \mathbf{e}(S_2, T_0)$ and $sS_0 \neq S_1\big)$

Figure 3: Games defining knowledge-of-exponent assumptions

SKE FOR ASYMMETRIC GROUPS. For asymmetric bilinear-group generators, we make assumption SKE in the first source group $\mathbb{G}_1$. Unlike for symmetric groups, a tuple $(S_0, sS_0, s^2S_0) \in \mathbb{G}_1^3$ is not verifiable via an asymmetric pairing. To make it verifiable, we *weaken* the assumption and require A to additionally output a $\mathbb{G}_2$-element T_0 as well as $T_1 = sT_0$, which enables verification (as done in game $\mathrm{SKE}_{\mathsf{aGen}}$).

Definition 2.15 (SKE) *Let* aGen *be an asymmetric-group generator. The SKE assumption holds for* aGen *in the first source group if for every PT* A *there exists a PT* E *s.t.* $\mathbf{Adv}^{\mathrm{ske}}_{\mathsf{aGen},\mathsf{A},\mathsf{E}}(\cdot)$ *is negligible, where* $\mathbf{Adv}^{\mathrm{ske}}_{\mathsf{aGen},\mathsf{A},\mathsf{E}}(\lambda) = \Pr[\mathrm{SKE}_{\mathsf{aGen},\mathsf{A},\mathsf{E}}(\lambda)]$ *and* SKE *is defined in Figure 3.*

We note that in addition to verifiability these additional elements T_0 and T_1 actually add to the plausibility of the assumption for asymmetric groups. Even if outputting S_2 was not required, one could argue that the following *stronger* assumption holds in Type-3 bilinear groups, in which DDH holds in $\mathbb{G}_1$ and in $\mathbb{G}_2$: it is hard to compute $(S_0, S_1, T_0, T_1) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2$ with $\mathbf{e}(S_1, T_0) = \mathbf{e}(S_0, T_1)$ without knowing the logarithms of S_1 to base S_0 (or equivalently T_1 to base T_0):[2] an adversary might choose S_0 and S_1 obliviously by hashing into the group; but if it was able to compute from them the respective T_0 and T_1 then this would break DDH in $\mathbb{G}_1$. (Given a DDH challenge $(S_0, S_1 = s_1S_0, S_2 = s_2S_0, R)$, compute T_0 and T_1 as above; then we have $R = s_1s_2S_0$ iff $\mathbf{e}(R, T_0) = \mathbf{e}(S_2, T_1)$.) Of course, this argument breaks down if there is an efficiently computable homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ or vice versa.

Finally, we note that q-PKE with $q = 0$ does not imply SKE, since a PKE adversary must return (V, W) which is a multiple of the received (P_1, P_2), while an SKE adversary can choose the "basis" (S_0, T_0) itself. The converse does not hold either (SKE$\not\Rightarrow$PKE), since an SKE adversary must return $S_2 = s^2S_0$.

[2] When fixing the generators S_0 and T_0, this corresponds to the assumption made by Abdolmaleki et al. [ABLZ17] to show S-ZK of their SNARK.

2.6 SKE in the Generic-Group Model

We show that SKE holds in the generic-group model. We show it for symmetric generic groups, which implies the result for asymmetric groups (where the adversary has less power). As [BFS16] did for DH-KEA, we reflect hashing into elliptic curves by providing the adversary with an additional generic operation: it can create new group elements without knowing their discrete logarithms (which are not known to the extractor either).

Theorem 2.16 *SKE, as defined in Definition 2.14, holds in the generic-group model with hashing into the group.*

In the proof of the theorem we will use the following lemma, which we prove first.

Lemma 2.17 *Let $\mathbb{F}$ be a field and let $A, B, C \in \mathbb{F}[X_1, \ldots, X_k]$, with degree of A, B and C at most 1. If $A \cdot C = B^2$ then for some $s \in \mathbb{F}$: $B = s \cdot A$.*

Proof. Let $\alpha_i, \beta_i, \gamma_i$, for $0 \leq i \leq k$, denote the coefficients of X_i (where $X_0 := 1$) in A, B, C, respectively. If $A = 0$ then $B = 0$ and the theorem follows. Assume thus $A \neq 0$; Define $j := \min\{i \in [0, k] : \alpha_j \neq 0\}$ and $s := \beta_j \cdot \alpha_j^{-1}$.

To prove the lemma, we will now show that for all $i \in [0, k]$:

$$\beta_i = s \cdot \alpha_i \ . \tag{1}$$

From $A \cdot C = B^2$ we have

$$L(\vec{X}) := \big(\beta_0 + \textstyle\sum_{i=1}^k \beta_i X_i\big)^2 - \big(\alpha_0 + \textstyle\sum_{i=1}^k \alpha_i X_i\big)\big(\gamma_0 + \textstyle\sum_{i=1}^k \gamma_i X_i\big) = 0 \ . \tag{2}$$

From $L(0, \ldots, 0) = 0$, we get: (I) $\beta_0^2 = \alpha_0 \gamma_0$, which implies that Eq. (1) holds for $i = 0$: either $\alpha_0 = 0$, then from (I): $\beta_0 = 0$; or $\alpha_0 \neq 0$, then $j = 0$ and Eq. (1) holds as well.

Let now $i \in [1, k]$ be arbitrarily fixed and let e_i denote the vector $(0, \ldots, 0, 1, 0, \ldots, 0)$ with 1 at position i. Consider $L(e_i) = 0$, which together with (I) yields

$$2\beta_0\beta_i + \beta_i^2 - \alpha_0\gamma_i - \alpha_i\gamma_0 - \alpha_i\gamma_i = 0 \ . \tag{3}$$

Similarly, from $L(2e_i) = 0$, we have $4\beta_0\beta_i + 4\beta_i^2 - 2\alpha_0\gamma_i - 2\alpha_i\gamma_0 - 4\alpha_i\gamma_i = 0$, which after subtracting Eq. (3) twice yields: (II) $\beta_i^2 = \alpha_i\gamma_i$. If $\alpha_i = 0$ then $\beta_i = 0$, which shows Eq. (1). For the remainder let us assume $\alpha_i \neq 0$.

Plugging (II) into Eq. (3) yields: (III) $2\beta_0\beta_i = \alpha_0\gamma_i - \alpha_i\gamma_0$.

If $\alpha_0 \neq 0$ then $j = 0$ and plugging (I) and (II) into (III) yields

$$2\beta_0\beta_i - \alpha_0\alpha_i^{-1}\beta_i^2 - \alpha_i\alpha_0^{-1}\beta_0^2 = 0 \ .$$

Solving for β_i yields the unique solution $\beta_i = \beta_0\alpha_0^{-1}\alpha_i$, which shows Eq. (1) for the case $\alpha_0 \neq 0$.

Let us now assume $\alpha_0 = 0$. By (I) we have $\beta_0 = 0$. If $i = j$ then Eq. (1) holds by definition of s. Assume $i \neq j$. From $L(e_i + e_j)$ we have (since $\alpha_0 = \beta_0 = 0$):

$$0 \ = \ \beta_i^2 + \beta_j^2 + 2\beta_i\beta_j - \alpha_i\gamma_0 - \alpha_i\gamma_i - \alpha_i\gamma_j - \alpha_j\gamma_0 - \alpha_j\gamma_i - \alpha_j\gamma_j \ = \ 2\beta_i\beta_j - \alpha_i\gamma_j - \alpha_j\gamma_i \ ,$$

where we used (II) and $\alpha_i\gamma_0 = \alpha_j\gamma_0 = 0$ (which follows from (III) and $\alpha_0 = \beta_0 = 0$). Together with (II) the latter yields $2\beta_i\beta_j - \alpha_i\alpha_j^{-1}\beta_j^2 - \alpha_j\alpha_i^{-1}\beta_i^2 = 0$. Solving for β_i yields the unique solution $\beta_i = \beta_j\alpha_j^{-1}\alpha_i$, which concludes the proof. ∎

Proof of Theorem 2.16. In the "traditional" generic-group model, group elements are represented by random strings and an adversary A only has access to operations on them (addition of elements in $\mathbb{G}$, multiplication of elements in $\mathbb{G}_T$ and pairing of elements in $\mathbb{G}$) via oracles. In particular, A can only produce new $\mathbb{G}$ elements by adding received elements.

We also need to reflect the fact that by "hashing into the group", A can create a new group element *without knowing its discrete logarithm w.r.t. one of the received elements.* We extend the generic-group model and provide the adversary with an additional operation, namely to request a new group element "independently of the received ones". (And neither the adversary nor the extractor we construct knows its discrete logarithm.)

For SKE the adversary A receives the group element P and needs to output (S_0, S_1, S_2) where for some s, t: $S_0 = tP$, $S_1 = sS_0 = stP$ and $S_2 = s^2 S_0 = s^2 tP$. The adversary can produce these group elements by combining the received generator P with newly generated ("hashed") group elements that it has requested. We represent the latter as $x_i P$, for $i = 1, \ldots k$, for some k. The extractor keeps track of the group operations performed by A and thus knows

$$\alpha_0, \ldots, \alpha_k, \beta_0, \ldots, \beta_k, \gamma_0, \ldots, \gamma_k \in \mathbb{Z}_p \tag{4}$$

such that A's output (S_0, S_1, S_2) is of the form

$$S_0 = \alpha_0 P + \textstyle\sum_{i=1}^k \alpha_i (x_i P) \qquad S_1 = \beta_0 P + \sum_{i=1}^k \beta_i (x_i P) \qquad S_2 = \gamma_0 P + \sum_{i=1}^k \gamma_i (x_i P)$$

Note that the extractor does however not know $x := (x_1, \ldots, x_k)$.

Assume the adversary wins and $\mathbf{e}(S_1, S_1) = \mathbf{e}(S_0, S_2)$. Taking the logarithms of the latter yields

$$\big(\beta_0 + \textstyle\sum_{i=1}^k \beta_i x_i\big)^2 - \big(\alpha_0 + \sum_{i=1}^k \alpha_i x_i\big)\big(\gamma_0 + \sum_{i=1}^k \gamma_i x_i\big) = 0 \ . \tag{5}$$

Since the adversary has no information about $x_1, \ldots, x_k$ (except for a negligible information leak by comparing group elements, which we ignore), the values in Eq. (4) are generated independently of $x_1, \ldots, x_k$. By the Schwartz-Zippel lemma the probability that Eq. (5) holds when $x_1, \ldots, x_k$ are randomly chosen is negligible, except if the left-hand side corresponds to the zero polynomial. With overwhelming probability we thus have

$$B(\vec{X})^2 - A(\vec{X}) \cdot C(\vec{X}) = 0$$

with

$$A(\vec{X}) = \alpha_0 + \textstyle\sum_{i=1}^k \alpha_i X_i \qquad B(\vec{X}) = \beta_0 + \sum_{i=1}^k \beta_i X_i \qquad C(\vec{X}) = \gamma_0 + \sum_{i=1}^k \gamma_i X_i$$

By Lemma 2.17 we have that $B = s\,A$ for some $s \in \mathbb{F}$. The extractor computes and returns s, which is correct since $S_1 = B(\vec{x})P = sA(\vec{x})P = s\,S_0$. ▮

3 SNARKs

We start with a formal definition of SNARGs and SNARKs.

Definition 3.1 (SNARG) *An NI system* $\Pi = (\Pi.\mathsf{Pg}, \Pi.\mathsf{P}, \Pi.\mathsf{V})$ *is a* succinct non-interactive argument *for relation generator* Rg *if it is complete and sound, as in Definition 2.1, and moreover succinct, meaning that for all* $\lambda \in \mathbb{N}$*, all* $R \in [\mathsf{Rg}(1^\lambda)]$*, all* $crs \in [\Pi.\mathsf{Pg}(R)]$*, all* $x \in L(R)$*, all* $w \in R(x)$ *and all* $\pi \in [\Pi.\mathsf{P}(1^\lambda, crs, x, w)]$ *we have* $|\pi| = \mathrm{poly}(\lambda)$ *and* $\Pi.\mathsf{V}(1^\lambda, crs, x, \pi)$ *runs in time* $\mathrm{poly}(\lambda + |x|)$.

Definition 3.2 (SNARK) *A SNARG* Π *is a* succinct non-interactive argument of knowledge *if it satisfies knowledge soundness, as in Definition 2.2.*

Gennaro, Gentry, Parno and Raykova [GGPR13] base their SNARK constructions on *quadratic programs*. In particular, they show how to convert any boolean circuit into a quadratic span program and any arithmetic circuit into a quadratic arithmetic program (QAP).

Definition 3.3 (QAP) *A* quadratic arithmetic program *over a field* $\mathbb{F}$ *is a tuple of the form*

$$\left(\mathbb{F}, n, \{A_i(X), B_i(X), C_i(X)\}_{i=0}^m, Z(X)\right) ,$$

where $A_i(X), B_i(X), C_i(X), Z(X) \in \mathbb{F}[X]$*, which define a language of statements* $(s_1, \ldots, s_n) \in \mathbb{F}^n$ *and witnesses* $(s_{n+1}, \ldots, s_m) \in \mathbb{F}^{m-n}$ *such that*

$$\Big(A_0(X) + \sum_{i=1}^m s_i A_i(X)\Big) \cdot \Big(B_0(X) + \sum_{i=1}^m s_i B_i(X)\Big) = C_0(X) + \sum_{i=1}^m s_i C_i(X) + H(X) \cdot Z(X) \ , \quad (6)$$

for some degree-$(d-2)$ *quotient polynomial* $H(X)$*, where* d *is the degree of* $Z(X)$ *(we assume the degrees of all* $A_i(X), B_i(X), C_i(X)$ *are at most* $d-1$*).*

Definition 3.4 (Strong QAP) *A* strong QAP *is a QAP such that for any* $(r_1, \ldots, r_m, s_1, \ldots, s_m, t_1, \ldots, t_m) \in \mathbb{F}^{3m}$ *for which* $Z(X)$ *divides*

$$\left(A_0(X) + \textstyle\sum_{i=1}^m r_i A_i(X)\right) \cdot \left(B_0(X) + \textstyle\sum_{i=1}^m s_i B_i(X)\right) - C_0(X) + \textstyle\sum_{i=1}^m t_i C_i(X) \ , \quad (7)$$

it must be the case that $(r_1, \ldots, r_m) = (s_1, \ldots, s_m) = (t_1, \ldots, t_m)$*.*

All of the discussed SNARK constructions are for QAPs defined over a bilinear group. We will thus consider relation generators Rg of the following form:

Definition 3.5 (QAP relation) *A QAP relation generator* Rg *is a PT algorithm that on input* 1^λ *returns a relation description of the following form:*

$$R = \left(Gr, n, \vec{A}, \vec{B}, \vec{C}, Z\right) \ \text{ where } Gr \text{ is a bilinear group whose order } p \text{ defines } \mathbb{F} := \mathbb{Z}_p \text{ and}$$
$$\vec{A}, \vec{B}, \vec{C} \in \left(\mathbb{F}^{(d-1)}[X]\right)^{(m+1)}, \ Z \in \mathbb{F}^{(d)}[X], \ n \le m \ . \quad (8)$$

For $x \in \mathbb{F}^n$ *and* $w \in \mathbb{F}^{m-n}$ *we define* $R(x, w) = \mathsf{true}$ *iff there exists* $H(X) \in \mathbb{F}[X]$ *so that Eq. (6) holds for* $s := x \,\|\, w$ *(where "*$\|$*" denotes concatenation).*

4 GGPR's QAP-Based SNARK

Gennaro et al. [GGPR13] presented the first zero-knowledge SNARK construction for arithmetic circuits that are expressed as quadratic arithmetic programs. They separate the CRS into a (long) part pk, used to compute proofs, and a (short) part vk, used to verify them. Their construction is detailed in Figure 4. As it is defined over symmetric bilinear groups, we assume that Gr returned by Rg is symmetric.

We define procedure CRS VERIFICATION, which a prover runs on a CRS before using it the first time, as follows:

CRS VERIFICATION. On input (R, vk, pk), let $\{a_{i,j}\}$, $\{b_{i,j}\}$, $\{c_{i,j}\}$, $\{z_k\}$ denote the coefficients of $A_i(X)$, $B_i(X)$, $C_i(X)$ and $Z(X)$, respectively, that are contained in R, for $0 \le i \le m$ and $0 \le j \le d-1$ and $0 \le k \le d$.

KEY GENERATION. On input R as in Eq. (8) representing a QAP for a symmetric group Gr do the following:

1. Sample $P \leftarrow_{\$} \mathbb{G}^*$ and $\tau, \alpha, \beta_A, \beta_B, \beta_C \leftarrow_{\$} \mathbb{F}$, conditioned on $Z(\tau) \neq 0$ and $\gamma \leftarrow_{\$} \mathbb{F}^*$.
2. Set $vk = (P_1, P_2, vk_A, vk_{B,0}, vk_{C,0}, vk_Z, vk_\alpha, vk_\gamma, vk''_{A,\gamma}, vk''_{B,\gamma}, vk''_{C,\gamma})$ where

$$\{vk_{A,i}\}_{i=0}^n := \{A_i(\tau)P\}_{i=0}^n \qquad vk_{B,0} := B_0(\tau)P \qquad vk_{C,0} := C_0(\tau)P$$
$$vk_Z := Z(\tau)P \qquad vk_\alpha := \alpha P \qquad vk_\gamma := \gamma P$$
$$vk''_{A,\gamma} := \beta_A\gamma P \qquad vk''_{B,\gamma} := \beta_B\gamma P \qquad vk''_{C,\gamma} := \beta_C\gamma P$$

3. Set $pk = (pk_A, pk'_A, pk''_A, pk''_{Z,A}, pk_B, pk'_B, pk''_B, pk''_{Z,B}, pk_C, pk'_C, pk''_C, pk''_{Z,C}, pk_H, pk'_H, pk_Z, pk'_Z)$,

$$\begin{array}{llll} \text{where for } i = n+1, \ldots, m: & pk_{A,i} := A_i(\tau)P & pk'_{A,i} := A_i(\tau)\alpha P & pk''_{A,i} := A_i(\tau)\beta_A P \\ \text{for } i = 1, \ldots, m: & pk_{B,i} := B_i(\tau)P & pk'_{B,i} := B_i(\tau)\alpha P & pk''_{B,i} := B_i(\tau)\beta_B P \\ & pk_{C,i} := C_i(\tau)P & pk'_{C,i} := C_i(\tau)\alpha P & pk''_{C,i} := C_i(\tau)\beta_C P \\ \text{for } i = 0, \ldots, d: & pk_{H,i} := \tau^i P & pk'_{H,i} := \tau^i\alpha P & \\ \text{and moreover} & pk_Z := Z(\tau)P & pk'_Z := Z(\tau)\alpha P & \\ & pk_{A,0} := A_0(\tau)P & pk'_{A,0} := A_0(\tau)\alpha P & pk''_{Z,A} := Z(\tau)\beta_A P \\ & pk_{B,0} := B_0(\tau)P & pk'_{B,0} := B_0(\tau)\alpha P & pk''_{Z,B} := Z(\tau)\beta_B P \\ & pk_{C,0} := C_0(\tau)P & pk'_{C,0} := C_0(\tau)\alpha P & pk''_{Z,C} := Z(\tau)\beta_C P \end{array}$$

4. Return $crs := (vk, pk)$.

PROVE. On input R, (vk, pk) and $\vec{s} \in \mathbb{F}^m$ s.t. Eq. (6) is satisfied for some $H'(X) \in \mathbb{F}[X]$:

1. **If** (R, vk, pk) **does not pass** CRS VERIFICATION **then return** $\bot$**.**
2. Sample $\delta_A, \delta_B, \delta_C \leftarrow_{\$} \mathbb{F}$ and define
$$A(X) := A_0(X) + \textstyle\sum_{i=1}^m s_i A_i(X) + \delta_A Z(X)$$
$$B(X) := B_0(X) + \textstyle\sum_{i=1}^m s_i B_i(X) + \delta_B Z(X)$$
$$C(X) := C_0(X) + \textstyle\sum_{i=1}^m s_i C_i(X) + \delta_C Z(X)$$
3. Compute $H(X)$ s.t. $A(X)B(X) - C(X) = H(X)Z(X)$ and let $(h_0, \ldots, h_d) \in \mathbb{F}^{d+1}$ be its coefficients. (If $H'(X)$ satisfies Eq. (6) then $H(X) = H'(X) + \delta_A B(X) + \delta_B A(X) - \delta_A\delta_B Z(X) - \delta_C$.)
4. Define
$$\pi_A := \textstyle\sum_{i=n+1}^m s_i pk_{A,i} + \delta_A\, pk_Z \qquad \pi'_A := \textstyle\sum_{i=n+1}^m s_i pk'_{A,i} + \delta_A\, pk'_Z$$
$$\pi_B := \textstyle\sum_{i=1}^m s_i pk_{B,i} + \delta_B\, pk_Z \qquad \pi'_B := \textstyle\sum_{i=1}^m s_i pk'_{B,i} + \delta_B\, pk'_Z$$
$$\pi_C := \textstyle\sum_{i=1}^m s_i pk_{C,i} + \delta_C\, pk_Z \qquad \pi'_C := \textstyle\sum_{i=1}^m s_i pk'_{C,i} + \delta_C\, pk'_Z$$
$$\pi_H := \textstyle\sum_{i=1}^d h_i pk_{H,i} \qquad \pi'_H := \textstyle\sum_{i=1}^d h_i pk'_{H,i}$$
$$\pi_K := \textstyle\sum_{i=n+1}^m s_i pk''_{A,i} + \delta_A pk''_{Z,A} + \sum_{i=1}^m s_i pk''_{B,i} + \delta_B pk''_{Z,B} + \sum_{i=1}^m s_i pk''_{C,i} + \delta_C pk''_{Z,C}$$
5. Return $\pi := (\pi_A, \pi'_A, \pi_B, \pi'_B, \pi_C, \pi'_C, \pi_H, \pi'_H, \pi_K)$.

VERIFY. On input R, vk, $\vec{x} \in \mathbb{F}^n$ and proof $\pi \in \mathbb{G}^9$:

1. Compute $vk_x := vk_{A,0} + \sum_{i=1}^n x_i vk_{A,i}$.
2. Check validity of π'_A, π'_B, π'_C and π'_H: $\mathbf{e}(\pi'_H, P) = \mathbf{e}(\pi_H, vk_\alpha)$
$$\mathbf{e}(\pi'_A, P) = \mathbf{e}(\pi_A, vk_\alpha) \qquad \mathbf{e}(\pi'_B, P) = \mathbf{e}(\pi_B, vk_\alpha) \qquad \mathbf{e}(\pi'_C, P) = \mathbf{e}(\pi_C, vk_\alpha)$$
3. Check same coefficients were used via π_K: $\mathbf{e}(\pi_K, vk_\gamma) = \mathbf{e}(\pi_A, vk''_{A,\gamma}) \cdot \mathbf{e}(\pi_B, vk''_{B,\gamma}) \cdot \mathbf{e}(\pi_C, vk''_{C,\gamma})$
4. Check QAP is satisfied: $\mathbf{e}(vk_x + \pi_A, vk_{B,0} + \pi_B) = \mathbf{e}(\pi_H, vk_Z) \cdot \mathbf{e}(vk_{C,0} + \pi_C, P)$
5. If all checks in 2.–4. succeeded then return `true` and otherwise `false`.

Figure 4: The original QAP-based SNARK [GGPR13] with CRS verification (in bold)

1. Check $P \neq 0_{\mathbb{G}}$.
2. Check correct choice of τ, γ: $vk_Z \neq 0_{\mathbb{G}}$ and $vk_\gamma \neq 0_{\mathbb{G}}$.
3. Check consistency of pk_H and pk'_H: $P = pk_{H,0}$ and

$$\begin{aligned} &\text{for } i = 1, \ldots, d: && \mathbf{e}(pk_{H,i}, P) = \mathbf{e}(pk_{H,i-1}, pk_{H,1}) \\ &\text{for } i = 0, \ldots, d: && \mathbf{e}(pk'_{H,i}, P) = \mathbf{e}(pk_{H,i}, vk_\alpha) \end{aligned}$$

4. Check consistency of vk:

$$\text{for } i = 0, \ldots, n: \ vk_{A,i} = \textstyle\sum_{j=0}^{d-1} a_{i,j} pk_{H,j}$$
$$vk_{B,0} = \textstyle\sum_{j=0}^{d-1} b_{0,j} pk_{H,j} \qquad vk_{C,0} = \textstyle\sum_{j=0}^{d-1} c_{0,j} pk_{H,j} \qquad vk_Z = \textstyle\sum_{j=0}^{d} z_j pk_{H,j}$$

5. Check consistency of the remaining pk elements: for $i = n+1, \ldots, m$:

$$pk_{A,i} = \textstyle\sum_{j=0}^{d-1} a_{i,j} pk_{H,j} \quad \mathbf{e}(pk'_{A,i}, P) = \mathbf{e}(pk_{A,i}, vk_\alpha) \quad \mathbf{e}(pk''_{A,i}, vk_\gamma) = \mathbf{e}(pk_{A,i}, vk''_{A,\gamma})$$

   for $i = 1, \ldots, m$:

$$\begin{aligned} pk_{B,i} &= \textstyle\sum_{j=0}^{d-1} b_{i,j} pk_{H,j} \quad \mathbf{e}(pk'_{B,i}, P) = \mathbf{e}(pk_{B,i}, vk_\alpha) \quad \mathbf{e}(pk''_{B,i}, vk_\gamma) = \mathbf{e}(pk_{B,i}, vk''_{B,\gamma}) \\ pk_{C,i} &= \textstyle\sum_{j=0}^{d-1} c_{i,j} pk_{H,j} \quad \mathbf{e}(pk'_{C,i}, P) = \mathbf{e}(pk_{C,i}, vk_\alpha) \quad \mathbf{e}(pk''_{C,i}, vk_\gamma) = \mathbf{e}(pk_{C,i}, vk''_{C,\gamma}) \end{aligned}$$

   and moreover:

$$\begin{aligned} pk_Z &= \textstyle\sum_{j=0}^{d} z_i pk_{H,j} && \mathbf{e}(pk'_Z, P) = \mathbf{e}(pk_Z, vk_\alpha) \\ pk_{A,0} &= vk_{A,0} && \mathbf{e}(pk'_{A,0}, P) = \mathbf{e}(pk_{A,0}, vk_\alpha) \quad \mathbf{e}(pk''_{Z,A}, vk_\gamma) = \mathbf{e}(pk_Z, vk''_{A,\gamma}) \\ pk_{B,0} &= vk_{B,0} && \mathbf{e}(pk'_{B,0}, P) = \mathbf{e}(pk_{B,0}, vk_\alpha) \quad \mathbf{e}(pk''_{Z,B}, vk_\gamma) = \mathbf{e}(pk_Z, vk''_{B,\gamma}) \\ pk_{C,0} &= vk_{C,0} && \mathbf{e}(pk'_{C,0}, P) = \mathbf{e}(pk_{C,0}, vk_\alpha) \quad \mathbf{e}(pk''_{Z,C}, vk_\gamma) = \mathbf{e}(pk_Z, vk''_{C,\gamma}) \end{aligned}$$

6. If all checks in 2.–5. succeeded then return true and otherwise false.

Standard security. Adding CRS verification to Gennaro et al.'s scheme does not alter its security as proved in [GGPR13]. In fact, knowledge soundness is a notion that is independent of the prove algorithm Π.P and it follows by inspection that an honestly computed CRS satisfies verification.

Theorem 4.1 ([GGPR13]) *Let* Rg *be a relation generator that on input* 1^λ *returns a QAP of degree at most* $d(\lambda)$ *over a symmetric group* Gr*. Define a group generator* sGen *that returns the first output* Gr *of* Rg *and let* $q := \max\{2d-1, d+2\}$*. If the* q*-PDH and the* d*-PKE assumptions hold for* sGen *then the scheme in Figure 4 for* Rg *is knowledge-sound. Moreover, it is statistical zero-knowledge.*

CRS verifiability. We show that a CRS that passes verification is constructed as in KEY GENERATION, that is, that exist values $\tau, \alpha, \beta_A, \beta_B, \beta_C \in \mathbb{F}$ such that the conditions in Item 1. of KEY GENERATION are satisfied and vk and pk are as in Items 2. and 3. Let $\tau, \alpha, \xi_A, \xi_B, \xi_C, \gamma \in \mathbb{F}$ be the discrete logarithms of the elements $pk_{H,1}$, vk_α, $vk''_{A,\gamma}$, $vk''_{B,\gamma}$, $vk''_{C,\gamma}$ and vk_γ, respectively. By Check 2. in CRS VERIFICATION we have that $\gamma \neq 0$. Define $\beta_A := x_A \gamma^{-1}$, $\beta_B := x_B \gamma^{-1}$, $\beta_C := x_C \gamma^{-1}$.

Check 3. ensures that pk_H and pk_H are correctly computed w.r.t. τ and α and Check 4. ensures that $\{vk_{A,i}\}_{i=0}^{n}$, $vk_{B,0}$ and $vk_{C,0}$ are correctly computed w.r.t. τ.

Check 5. ensures that $\{pk_{A,i}, pk'_{A,i}, pk''_{A,i}\}_{i=n+1}^{m}$ are correctly computed w.r.t. τ, α and β_A; and $\{pk_{B,i}, pk'_{B,i}, pk''_{B,i}, pk_{C,i}, pk'_{C,i}, pk''_{C,i}\}_{i=1}^{m}$ are correctly computed w.r.t. τ, α, β_B and β_C. Moreover, it checks that pk_Z, pk'_Z, $pk_{A,0}$, $pk'_{A,0}$, $pk''_{Z,A}$, $pk_{B,0}$, $pk'_{B,0}$, $pk''_{Z,B}$, $pk_{C,0}$, $pk'_{C,0}$ and $pk''_{Z,C}$ are also of the correct form.

Trapdoor extraction. In order to prove subversion zero knowledge, we construct a simulator $(\Pi.\mathsf{Sim.crs}, \Pi.\mathsf{Sim.pf})$ for any CRS subvertor. Let X be a CRS subvertor that returns (vk, pk). Define $\mathsf{X}'(1^\lambda; r)$ that runs $(vk, pk) \leftarrow \mathsf{X}(1^\lambda; r)$, parses vk and pk as in Figure 4 and returns $(pk_{H,0}, pk_{H,1}, pk_{H,2})$. By SKE (Definition 2.14) there exists a PT algorithm $\mathsf{E}_{\mathsf{X}'}$ such that if for some $P \in \mathbb{G}$, $\tau \in \mathbb{F}$: $pk_{H,0} = P$, $pk_{H,1} = \tau P$, $pk_{H,2} = \tau^2 P$ then with overwhelming probability $\mathsf{E}_{\mathsf{X}'}$ extracts τ. Using $\mathsf{E}_{\mathsf{X}'}$ we define the CRS simulator $\mathsf{S.crs}$ as follows: On input 1^λ do the following:

1. Sample randomness for X: $r \leftarrow_{\$} \{0,1\}^{\mathsf{X.rl}(\lambda)}$.
2. Run $(vk, pk) \leftarrow \mathsf{X}(1^\lambda; r)$.
3. If (R, vk, pk) passes verification then $\tau \leftarrow_{\$} \mathsf{E}_{\mathsf{X}'}(1^\lambda, r)$; else $\tau \leftarrow \perp$
4. Return $((vk, pk), r, \tau)$.

Proof simulation. Given (vk, pk), trapdoor τ and a statement $x \in \mathbb{F}^n$, the proof simulator $\mathsf{S.pf}$ is defined as follows:

1. If $\tau = \perp$ then return $\perp$.
2. Use τ to compute $Z(\tau)$ (which in a verified CRS is non-zero). Compute the following "simulation keys":

$$sk_A := Z(\tau)^{-1} pk''_{Z,A} \qquad sk_B := Z(\tau)^{-1} pk''_{Z,B} \qquad sk_C := Z(\tau)^{-1} pk''_{Z,C}$$

(For a valid CRS, we have $sk_A = \beta_A P$ and $sk_B = \beta_B P$ and $sk_C = \beta_C P$.)

3. Define $v_x := \sum_{j=0}^{d-1} a_{0,j}\tau^j + \sum_{i=1}^{n} x_i \sum_{j=0}^{d-1} a_{i,j}\tau^j$. Set $vk_x := v_x P$ and $vk'_x := v_x vk_\alpha$.
4. Choose $a, b, c \leftarrow_{\$} \mathbb{F}$ and define the proof $\pi := (\pi_A, \pi'_A, \pi_B, \pi'_B, \pi_C, \pi'_C, \pi_K, \pi_H)$ as follows:

$$\begin{aligned}
\pi_A &:= (a - v_x)P = aP - vk_x & \pi'_A &:= (a - v_x) vk_\alpha \\
\pi_B &:= (b - B_0(\tau))P = bP - vk_{B,0} & \pi'_B &:= (b - B_0(\tau)) vk_\alpha \\
\pi_C &:= (c - C_0(\tau))P = cP - vk_{C,0} & \pi'_C &:= (c - C_0(\tau)) vk_\alpha \\
\pi_H &:= Z(\tau)^{-1}(ab - c)P & \pi'_H &:= Z(\tau)^{-1}(ab - c) vk_\alpha \\
\pi_K &:= (a - v_x) sk_A + (b - B_0(\tau)) sk_B + (c - C_0(\tau)) sk_C & &
\end{aligned}$$

Theorem 4.2 *Let* Rg *be a relation generator that outputs strong QAPs and implicitly defines a symmetric bilinear-group generator* sGen*. If SKE holds for* sGen *then the GGPR QAP-based SNARK [GGPR13] with CRS verification given in Figure 4 for* Rg *satisfies subversion zero knowledge.*

Proof. Consider $(vk, pk) \leftarrow \mathsf{X}(1^\lambda; r)$ and let E denote the event that (R, vk, pk) passes CRS VERIFICATION (in which case X returns $(P, \tau P, \tau^2 P)$) but $\mathsf{E}_{\mathsf{X}'}$ fails to extract τ. Since a correct (vk, pk) satisfies $\mathbf{e}(pk_{H,1}, pk_{H,1}) = \mathbf{e}(pk_{H,0}, pk_{H,2})$, by assumption SKE the probability of E is negligible. It suffices thus to show that, conditioned on E not happening, the probability that A outputs 1 in game S-ZK when $b = 0$ is the same as when $b = 1$.

If (vk, pk) does not pass verification then $\tau = \bot$ and both prover and proof simulator return $\bot$.

If (vk, pk) verifies then (because of $\neg E$) $\mathsf{E}_{\mathsf{X}'}$ extracts τ. We show that the outputs of the prover and the proof simulator are distributed equivalently. Above we showed that if the CRS verifies then there exist $\tau, \alpha, \beta_A, \beta_B, \beta_C, \gamma \in \mathbb{F}$ with $Z(\tau) \neq 0$ and $\gamma \neq 0$ such that vk and pk are defined as in Items 2. and 3. in Key generation.

Moreover, in a real proof the elements $\delta_A Z(\tau)P$ in π_A and $\delta_B Z(\tau)P$ in π_B and $\delta_C Z(\tau)P$ in π_C make π_A, π_B and π_C uniformly random. For a fixed vk and π_A, π_B and π_C, the equations in 2. of Verify uniquely determine π'_A, π'_B and π'_C, and the equations in 3. and 4. uniquely determine π_K and π_H (since $vk_\gamma \neq 0_{\mathbb{G}}$ and $vk_Z \neq 0_{\mathbb{G}}$).

In a simulated proof π_A, π_B and π_C are also uniformly random, so it suffices to show that the remaining proof elements satisfy the verification equations:

$$\begin{aligned}
\mathbf{e}(\pi'_A, P) &= \mathbf{e}\big((a - v_x)\alpha P,\ P\big) = \mathbf{e}(\pi_A, vk_\alpha) \\
\mathbf{e}(\pi'_B, P) &= \mathbf{e}\big((b - B_0(\tau))\alpha P,\ P\big) = \mathbf{e}(\pi_B, vk_\alpha) \\
\mathbf{e}(\pi'_C, P) &= \mathbf{e}\big((c - C_0(\tau))\alpha P,\ P\big) = \mathbf{e}(\pi_C, vk_\alpha) \\
\mathbf{e}(\pi_K, vk_\gamma) &= \mathbf{e}\big((a - v_x)\beta_A P + (b - B_0(\tau))\beta_B P + (c - C_0(\tau))\beta_C P,\ \gamma P\big) \\
&= \mathbf{e}(\pi_A, vk''_{A,\gamma}) \cdot \mathbf{e}(\pi_B, vk''_{B,\gamma}) \cdot \mathbf{e}(\pi_C, vk''_{C,\gamma}) \\
\mathbf{e}(\pi_H, vk_Z) &= \mathbf{e}\big(Z(\tau)^{-1}(ab - c)P, Z(\tau)P\big) = \mathbf{e}\big(aP, bP\big) \cdot \mathbf{e}\big(cP, P\big)^{-1} \\
&= \mathbf{e}(vk_x + \pi_A, vk_{B,0} + \pi_B) \cdot \mathbf{e}(vk_{C,0} + \pi_C, P)^{-1}
\end{aligned}$$

This concludes the proof. ∎

Corollary 4.3 *Let* Rg *be a relation generator that outputs strong QAPs. The GGPR QAP-based SNARK [GGPR13] with CRS verification (Figure 4) for* Rg *satisfies perfect subversion witness indistinguishability.*

Proof. In Theorem 4.2 we showed that proofs under a (possibly maliciously generated but) valid CRS are uniform group elements subject to satisfying the verification equation. Proofs using different witnesses are thus equally distributed. ∎

GGPR's QSP-Based SNARK

Gennaro et al. [GGPR13] also introduced (strong) quadratic span programs (QSP) and show how to efficiently convert any *boolean* circuit into an equivalent strong QSP. Strong QSPs are defined similarly to QAPs (Definition 3.3) except that there are no polynomials $C_i(X)$ and the coefficients can be different (like $(r_1, \ldots, r_m)$ and $(s_1, \ldots, s_m)$ in Eq. (7)). Moreover the statement $x \in \{0,1\}^{n'}$ with $n = 2n'$ is mapped to $\vec{r}$ and $\vec{s}$ as follows: for $i \in \{1, \ldots, n'\}$: $r_{2i} = s_{2i} := x_i$ and $r_{2i-1} = s_{2i-1} := 1 - x_i$.

The first SNARK construction in [GGPR13] is based on strong QSPs and is obtained by setting $C_i(X) :\equiv 0$ for all i in the QAP-based one above. It is straightforward to verify that all our results for the QAP-based construction carry over to the QSP-based SNARK.

5 Asymmetric Pinocchio

Pinocchio [PHGR13] is one of the central SNARK systems. Ben-Sasson, Chiesa, Tromer and Virza [BCTV14] proposed a variant in asymmetric groups for which they also shorten the verification key. Their system is implemented in libsnark [BCG+14b] and underlies Zcash.

Campanelli et al. [CGGN17] show that the protocol is not subversion-zero-knowledge and expect major changes to make it secure. In the following we show that by adding merely 4 group elements to the CRS (which we denote by *ck* for "checking key"), we can enable verification of well-formedness of (vk, pk) by using the pairing available in the bilinear group. We then show that under SKE (Definition 2.15), our modification of the scheme from [BCTV14] achieves subversion zero knowledge. The protocol is given in Figure 5, where we underlined our modifications. Below we define procedure CRS VERIFICATION, which a prover runs on a CRS before using it the first time.

Theorem 5.1 ([PHGR13, BCTV14]) *Let* Rg *be a relation generator that on input* 1^λ *returns a QAP of degree at most* $d(\lambda)$ *over an asymmetric group Gr. Define a group generator* aGen *that returns the first output Gr of* Rg *and let* $q := 4d + 4$*. If the q-PDH, the q-PKE and the* $2q$*-SDH assumptions hold for* aGen *then the scheme in Figure 5* without including *ck* in the CRS *is knowledge-sound. Moreover, it is statistical zero-knowledge.*

Standard security. Inspecting the proof of the theorem in [PHGR13], it is easily seen that the additional elements contained in *ck* can be produced by the reduction. Moreover, knowledge soundness is independent of the prove algorithm $\Pi.\mathsf{P}$, and a correctly generated CRS passes CRS verification. This yields the following.

Corollary 5.2 (to Theorem 5.1) *Let* Rg *and* aGen *be as in Theorem 5.1. If the q-PDH, the q-PKE and the* $2q$*-SDH assumptions hold for* aGen *for* $q := 4d + 4$ *then the scheme in Figure 5 is knowledge-sound. Moreover, it is statistical zero-knowledge.*

<u>CRS VERIFICATION.</u> On input (R, vk, pk, ck), let $\{a_{i,j}\}$, $\{b_{i,j}\}$, $\{c_{i,j}\}$, $\{z_k\}$ denote the coefficients of $A_i(X)$, $B_i(X)$, $C_i(X)$ and $Z(X)$, respectively, for $0 \le i \le m$ and $0 \le j \le d-1$ and $0 \le k \le d$.

1. Check $P_1 \ne 0_{\mathbb{G}_1}$ and $P_2 \ne 0_{\mathbb{G}_2}$.
2. Check choice of secret values: $ck_A \ne 0_{\mathbb{G}_2}$, $ck_B \ne 0_{\mathbb{G}_2}$, $vk_\gamma \ne 0_{\mathbb{G}_2}$, $vk_{\beta\gamma} \ne 0_{\mathbb{G}_1}$ and $vk_Z \ne 0_{\mathbb{G}_2}$.
3. Check consistency of pk_H: Check $pk_{H,0} = P_1$; and for $i = 1, \dots, d$:
$$\mathbf{e}(pk_{H,i}, P_2) = \mathbf{e}(pk_{H,i-1}, ck_H)$$
4. Check consistency of pk_A, pk'_A, pk_B, pk'_B: for $i = 0, \dots, m+3$:
$$\mathbf{e}(pk_{A,i}, P_2) = \mathbf{e}(\textstyle\sum_{j=0}^{d-1} a_{i,j} pk_{H,j}, ck_A) \qquad \mathbf{e}(pk'_{A,i}, P_2) = \mathbf{e}(pk_{A,i}, vk_A)$$
$$\mathbf{e}(P_1, pk_{B,i}) = \mathbf{e}(\textstyle\sum_{j=0}^{d-1} b_{i,j} pk_{H,j}, ck_B) \qquad \mathbf{e}(pk'_{B,i}, P_2) = \mathbf{e}(vk_B, pk_{B,i})$$
5. Check consistency of ck_C: $\mathbf{e}\big(pk_{A,m+1}, ck_B\big) = \mathbf{e}\big(\sum_{j=0}^{d} z_j pk_{H,j}, ck_C\big)$
(Note that for an honest CRS we have $pk_{A,m+1} = Z(\tau)\rho_A P_1 \ne 0$.)
6. Check consistency of vk: for $i = 0, \dots, n$: $vk_{IC,i} = pk_{A,i}$ and
$$\mathbf{e}(vk_{\beta\gamma}, P_2) = \mathbf{e}(P_1, \widehat{vk}_{\beta\gamma}) \qquad \mathbf{e}(P_1, vk_Z) = \mathbf{e}\big(\textstyle\sum_{j=0}^{d} z_j pk_{H,j}, ck_C\big)$$

Key generation. On input R as in Eq. (8) representing a QAP for an asymmetric group Gr do the following:

1. Sample $P_1 \leftarrow\!\!\$\, \mathbb{G}_1^*$ and $P_2 \leftarrow\!\!\$\, \mathbb{G}_2^*$
2. Set $\begin{bmatrix} A_{m+1} & B_{m+1} & C_{m+1} \\ A_{m+2} & B_{m+2} & C_{m+2} \\ A_{m+3} & B_{m+3} & C_{m+3} \end{bmatrix} := \begin{bmatrix} Z & 0 & 0 \\ 0 & Z & 0 \\ 0 & 0 & Z \end{bmatrix}$
3. Sample random $\rho_A, \rho_B, \beta, \gamma \leftarrow\!\!\$\, \mathbb{F}^*$ and $\tau, \alpha_A, \alpha_B, \alpha_C, \leftarrow\!\!\$\, \mathbb{F}$, conditioned on $Z(\tau) \neq 0$.
4. Set $vk = (P_1, P_2, vk_A, vk_B, vk_C, vk_\gamma, vk_{\beta\gamma}, \widehat{vk}_{\beta\gamma}, vk_Z, vk_{IC})$, where

$$vk_A := \alpha_A P_2 \qquad vk_B := \alpha_B P_1 \qquad vk_C := \alpha_C P_2$$
$$vk_\gamma := \gamma P_2 \qquad vk_{\beta\gamma} := \gamma\beta P_1 \qquad \widehat{vk}_{\beta\gamma} := \gamma\beta P_2$$
$$vk_Z := Z(\tau)\rho_A\rho_B P_2 \qquad \{vk_{IC,i}\}_{i=0}^n := \{A_i(\tau)\rho_A P_1\}_{i=0}^n$$

5. Set $pk = (pk_A, pk'_A, pk_B, pk'_B, pk_C, pk'_C, pk_K, pk_H)$ where

for $i = 0, \ldots, m+3$:
$$pk_{A,i} := A_i(\tau)\rho_A P_1 \qquad pk'_{A,i} := A_i(\tau)\alpha_A\rho_A P_1$$
$$pk_{B,i} := B_i(\tau)\rho_B P_2 \qquad pk'_{B,i} := B_i(\tau)\alpha_B\rho_B P_1$$
$$pk_{C,i} := C_i(\tau)\rho_A\rho_B P_1 \qquad pk'_{C,i} := C_i(\tau)\alpha_C\rho_A\rho_B P_1$$
$$pk_{K,i} := \beta(A_i(\tau)\rho_A + B_i(\tau)\rho_B + C_i(\tau)\rho_A\rho_B)P_1$$

for $i = 0, \ldots, d$:
$$pk_{H,i} := \tau^i P_1$$

6. **Set** $ck := (ck_A, ck_B, ck_C, ck_H)$ **where** $ck_A := \rho_A P_2$, $ck_B := \rho_B P_2$, $ck_C := \rho_A\rho_B P_2$, $ck_H := \tau P_2$**.**
7. Return $crs := (vk, pk, ck)$.

Prove. On input R, (vk, pk, ck) and $\vec{s} \in \mathbb{F}^m$ s.t. Eq. (6) is satisfied for some $H'(X) \in \mathbb{F}[X]$.

1. **If** (R, vk, pk, ck) **does not pass** CRS verification **then return** $\bot$**.**
2. Sample $\delta_A, \delta_B, \delta_C \leftarrow\!\!\$\, \mathbb{F}$ and define

$$A(X) := A_0(X) + \textstyle\sum_{i=1}^m s_i A_i(X) + \delta_A Z(X)$$
$$B(X) := B_0(X) + \textstyle\sum_{i=1}^m s_i B_i(X) + \delta_B Z(X)$$
$$C(X) := C_0(X) + \textstyle\sum_{i=1}^m s_i C_i(X) + \delta_C Z(X)$$

3. Compute $H(X)$ such that $A(X)B(X) - C(X) = H(X)Z(X)$; let $(h_0, \ldots, h_d) \in \mathbb{F}^{d+1}$ be its coefficients.
4. Define $\widetilde{pk}_{A,i} := \mathbb{1}_{i>n} pk_{A,i}$ (that is, $\widetilde{pk}_{A,i} = 0$ for $0 \le i \le n$ and $= pk_{A,i}$ otherwise)
Define $\widetilde{pk}'_{A,i} := \mathbb{1}_{i>n} pk'_{A,i}$
5. Let $\vec{c} := 1 \,\|\, \vec{s} \,\|\, \delta_A \,\|\, \delta_B \,\|\, \delta_C \in \mathbb{F}^{m+4}$ and compute ("$\langle\cdot,\cdot\rangle$" denotes the scalar product)

$$\pi_A := \langle \vec{c}, \widetilde{pk}_A \rangle \qquad \pi'_A := \langle \vec{c}, \widetilde{pk}'_A \rangle \qquad \pi_B := \langle \vec{c}, pk_B \rangle \qquad \pi'_B := \langle \vec{c}, pk'_B \rangle$$
$$\pi_C := \langle \vec{c}, pk_C \rangle \qquad \pi'_C := \langle \vec{c}, pk'_C \rangle \qquad \pi_K := \langle \vec{c}, pk_K \rangle \qquad \pi_H := \langle \vec{h}, pk_H \rangle$$

6. Return $\pi := (\pi_A, \pi'_A, \pi_B, \pi'_B, \pi_C, \pi'_C, \pi_K, \pi_H)$.

Verify. On input R, vk, $\vec{x} \in \mathbb{F}^n$ and proof $\pi \in \mathbb{G}_1^7 \times \mathbb{G}_2$.

1. Compute $vk_x := vk_{IC,0} + \sum_{i=1}^n x_i vk_{IC,i}$.
2. Check validity of π'_A, π'_B, and π'_C:

$$\mathbf{e}(\pi'_A, P_2) = \mathbf{e}(\pi_A, vk_A) \qquad \mathbf{e}(\pi'_B, P_2) = \mathbf{e}(vk_B, \pi_B) \qquad \mathbf{e}(\pi'_C, P_2) = \mathbf{e}(\pi_C, vk_C)$$

3. Check same coefficients were used via π_K: $\mathbf{e}(\pi_K, vk_\gamma) = \mathbf{e}(vk_x + \pi_A + \pi_C, \widehat{vk}_{\beta\gamma}) \cdot \mathbf{e}(vk_{\beta\gamma}, \pi_B)$
4. Check QAP is satisfied via π_H: $\mathbf{e}(vk_x + \pi_A, \pi_B) = \mathbf{e}(\pi_H, vk_Z) \cdot \mathbf{e}(\pi_C, P_2)$
5. If all checks in 2.–4. succeeded then return `true` and otherwise `false`.

Figure 5: S-ZK Asymmetric Pinocchio, adapted from [BCTV14].

7. Check consistency of pk_C, pk'_C, pk_K: for $i = 0, \ldots, m+3$:

$$\mathbf{e}(pk_{C,i}, P_2) = \mathbf{e}(\textstyle\sum_{j=0}^{d-1} c_{i,j} pk_{H,j}, ck_C) \qquad \mathbf{e}(pk'_{C,i}, P_2) = \mathbf{e}(pk_{C,i}, vk_C)$$
$$\mathbf{e}(pk_{K,i}, vk_\gamma) = \mathbf{e}(pk_{A,i} + pk_{C,i}, \widehat{vk}_{\beta\gamma}) \cdot \mathbf{e}(vk_{\beta\gamma}, pk_{B,i})$$

8. If all checks in 1.–7. succeeded then return true and otherwise false.

Remark 5.3 The condition that in Key generation $\rho_A, \rho_B, \beta, \gamma$ and $Z(\tau)$ must be non-zero is not made explicit in [BCTV14]. However if $\gamma = 0$ then any π_K satisfies the verification equation in 3; and if $\beta = 0$ and $\gamma \neq 0$ then no π_K satisfies it. If $Z(\tau) = 0$ or $\rho_A = 0$ or $\rho_B = 0$ then $vk_Z = 0_{\mathbb{G}_2}$ and setting π_B and π_C to zero always satisfies the equation in 4 in verification.

CRS verifiability. We show that a CRS (vk, pk, ck) that passes verification is constructed as in Key generation; in particular, there exist $\tau, \alpha_A, \alpha_B, \alpha_C \in \mathbb{F}$ and $\rho_A, \rho_B, \beta, \gamma, \in \mathbb{F}^*$ such that (vk, pk, ck) is computed as in Key generation. Let $\tau, \alpha_A, \alpha_B, \alpha_C, \rho_A, \rho_B, \gamma, \xi \in \mathbb{F}$ be the values defined by the logarithms of the elements ck_H, vk_A, vk_B, vk_C, ck_A, ck_B, vk_γ and $vk_{\beta\gamma}$, respectively. Check 2. ensures that ρ_A, ρ_B, γ, ξ and $Z(\tau)$ are all non-zero. Set $\beta := \xi\gamma^{-1} \neq 0$.

Check 3. ensures that pk_H is correctly computed w.r.t. τ. Check 4. ensures that pk_A, pk'_A, pk_B and pk'_B are correctly computed w.r.t. τ, ρ_A, ρ_B, α_A and α_B. Check 5. ensures that pk_C is correctly computed: since by 4., $pk_{A,m+1} = Z(\tau)\rho_A P_1$ and $Z(\tau) \neq 0$, we have $ck_C = \rho_A\rho_B P_2$. Check 6. ensures that $\widehat{vk}_{\beta\gamma}$ and vk_Z are correctly computed and Check 7. does the same for pk_C, pk'_C and pk_K. This shows that with respect to ck_H, vk_A, vk_B, vk_C, ck_A, ck_B, vk_γ and $vk_{\beta\gamma}$ (which implicitly define the randomness used in a CRS), all remaining elements $pk_A, pk'_A, pk_B, pk'_B, pk_C, pk'_C, pk_K$, pk_H, as well as $\widehat{vk}_{\beta\gamma}, vk_Z, vk_{IC}$ and ck_C are defined as prescribed by Key generation.

Trapdoor extraction. This is done exactly as for the scheme in Section 4. Let X be a CRS subvertor that outputs (vk, pk, ck). Define $\mathsf{X}'(1^\lambda; r)$ that runs $(vk, pk, ck) \leftarrow \mathsf{X}(1^\lambda; r)$, parses the output as above and returns $(pk_{H,0}, pk_{H,1}, pk_{H,2}, P_2, ck_H)$. By SKE for aGen (Definition 2.15) there exists a PT algorithm $\mathsf{E}_{\mathsf{X}'}$ such that if for some $\tau \in \mathbb{F}$: $pk_{H,1} = \tau\, pk_{H,0}$, $pk_{H,2} = \tau^2 pk_{H,0}$ and $ck_H = \tau P_2$ then with overwhelming probability $\mathsf{E}_{\mathsf{X}'}$ extracts τ. Using $\mathsf{E}_{\mathsf{X}'}$ we define the CRS simulator S.crs which computes (crs, r, td) as follows: On input 1^λ:

1. Sample randomness for X: $r \leftarrow\!\!\$\ \{0,1\}^{\mathsf{X.rl}(\lambda)}$.
2. Run $(vk, pk, ck) \leftarrow \mathsf{X}(1^\lambda; r)$.
3. If (R, vk, pk, ck) passes CRS verification then $\tau \leftarrow\!\!\$\ \mathsf{E}_{\mathsf{X}'}(1^\lambda, r)$; else $\tau \leftarrow \bot$.
4. Return $((vk, pk, ck), r, \tau)$.

Proof simulation. Given (vk, pk, ck), trapdoor τ and a statement $x \in \mathbb{F}^n$, the proof simulator S.pf is defined as follows:

1. If $\tau = \bot$ then return $\bot$.
2. Use τ to compute $Z(\tau)$ (which in a verified CRS is non-zero). Compute the following "simulation keys":

$$sk_A := Z(\tau)^{-1} pk_{A,m+1} = \rho_A P_1 \qquad sk'_A := Z(\tau)^{-1} pk'_{A,m+1} = \alpha_A \rho_A P_1$$
$$sk_B := Z(\tau)^{-1} pk_{B,m+2} = \rho_B P_2 \qquad sk'_B := Z(\tau)^{-1} pk'_{B,m+2} = \alpha_B \rho_B P_1$$
$$sk_C := Z(\tau)^{-1} pk_{C,m+3} = \rho_A \rho_B P_1 \qquad sk'_C := Z(\tau)^{-1} pk'_{C,m+3} = \alpha_C \rho_A \rho_B P_1$$
$$sk''_A = Z(\tau)^{-1} pk_{K,m+1} = \beta \rho_A P_1$$
$$sk''_B = Z(\tau)^{-1} pk_{K,m+2} = \beta \rho_B P_1 \qquad sk''_C = Z(\tau)^{-1} pk_{K,m+3} = \beta \rho_A \rho_B P_1$$

3. Compute $vk_x := pk_{A,0} + \sum_{i=1}^{n} x_i pk_{A,i}$ and $vk'_x := pk'_{A,0} + \sum_{i=1}^{n} x_i pk'_{A,i}$

4. Choose $a, b, c \leftarrow_{\$} \mathbb{F}$ and define the proof $\pi := (\pi_A, \pi'_A, \pi_B, \pi'_B, \pi_C, \pi'_C, \pi_K, \pi_H)$ with:

$$\begin{aligned} &\pi_A := a\, sk_A - vk_x = a\rho_A P_1 - vk_x && \pi'_A := a\, sk'_A - vk'_x = a\,\alpha_A\rho_A P_1 - \alpha_A vk_x \\ &\pi_B := b\, sk_B = b\,\rho_B P_2 && \pi'_B := b\, sk'_B = b\,\alpha_B\rho_B P_1 \\ &\pi_C := c\, sk_C = c\,\rho_A\rho_B P_1 && \pi'_C := c\, sk'_C = c\,\alpha_C\rho_A\rho_B P_1 \\ &\pi_K := a\, sk''_A + b\, sk''_B + c\, sk''_C && \pi_H := Z(\tau)^{-1}(ab - c)P_1 \end{aligned}$$

Theorem 5.4 *Let* Rg *be a QAP generator defining a bilinear-group generator* aGen *for which SKE holds. Then the scheme in Figure 5 for* Rg *satisfies subversion zero knowledge.*

Proof. Consider $(vk, pk, ck) \leftarrow \mathsf{X}(1^\lambda; r)$ and let E denote the event that (R, vk, pk, ck) passes CRS VERIFICATION but $\mathsf{E}_{\mathsf{X}'}$ fails to extract τ. From Check 3 in CRS VERIFICATION, we have $\mathbf{e}(pk_{H,1}, P_2) = \mathbf{e}(pk_{H,0}, ck_H)$ and $\mathbf{e}(pk_{H,2}, P_2) = \mathbf{e}(pk_{H,1}, ck_H)$; thus $(pk_{H,0}, pk_{H,1}, pk_{H,2}, P_2, ck_H)$ is a valid SKE tuple. By the SKE assumption the probability of E is thus negligible. It now suffices to show that, conditioned on E not happening, the probability that A outputs 1 in game S-ZK when $b = 0$ is the same as when $b = 1$.

If (vk, pk, ck) fails CRS VERIFICATION then $\tau = \perp$ and both prover and proof simulator return $\perp$. If (vk, pk, ck) verifies then (because of $\neg E$) $\mathsf{E}_{\mathsf{X}'}$ extracts τ.

We show that the outputs of the prover and the proof simulator are distributed equivalently. Above we showed that for a valid CRS there exist $\tau, \rho_A, \rho_B, \beta, \gamma, \alpha_A, \alpha_B, \alpha_C \in \mathbb{F}$ with $\rho_A \neq 0$, $\rho_B \neq 0$, $\beta \neq 0$, $\gamma \neq 0$ and $Z(\tau) \neq 0$ such that vk and pk are defined as in Items 4. and 5. in KEY GENERATION.

Because of this, $\delta_A Z(\tau)\rho_A P_1$, the $(m+2)$-th summand in π_A is uniformly random. And so are the $(m+3)$-th summand $\delta_B Z(\tau)\rho_B P_1$ of π_B and the $(m+4)$-th summand $\delta_C Z(\tau)\rho_A\rho_B P_1$ in π_C. But this means that π_A, π_B and π_C are uniformly random group elements. For fixed vk, π_A, π_B and π_C the equations in 2. of VERIFY uniquely determine π'_A, π'_B and π'_C, while the equations in 3. and 4. uniquely determine π_K and π_H (since $vk_\gamma \neq 0_{\mathbb{G}_2}$ and $vk_Z \neq 0_{\mathbb{G}_2}$).

Since for a valid CRS the values ρ_A and ρ_B are non-zero, the simulated proof elements π_A, π_B and π_C are also uniformly random. Thus, it suffices to show that the remaining proof elements satisfy the verification equations:

$$\begin{aligned} \mathbf{e}(\pi'_A, P_2) &= \mathbf{e}\big(a\,\alpha_A\rho_A P_1 - \alpha_A vk_x, P_2\big) = \mathbf{e}(\pi_A, vk_A) \\ \mathbf{e}(\pi'_B, P_2) &= \mathbf{e}\big(b\,\alpha_B\rho_B P_1, P_2\big) = \mathbf{e}(vk_B, \pi_B) \\ \mathbf{e}(\pi'_C, P_2) &= \mathbf{e}\big(c\,\alpha_C\rho_A\rho_B P_1, P_2\big) = \mathbf{e}(\pi_C, vk_C) \\ \mathbf{e}(\pi_K, vk_\gamma) &= \mathbf{e}\big(\beta(a\rho_A P_1 + b\rho_B P_1 + c\,\rho_A\rho_B P_1), \gamma P_2\big) = \mathbf{e}(vk_x + \pi_A + \pi_C, \widehat{vk}_{\beta\gamma}) \cdot \mathbf{e}(vk_{\beta\gamma}, \pi_B) \\ \mathbf{e}(\pi_H, vk_Z) &= \mathbf{e}\big(Z(\tau)^{-1}(ab - c)P_1, Z(\tau)\rho_A\rho_B P_2\big) = \\ &= \mathbf{e}\big(a\rho_A P_1, b\rho_B P_2\big) \cdot \mathbf{e}\big(c\rho_A\rho_B P_1, P_2\big)^{-1} = \mathbf{e}(vk_x + \pi_A, \pi_B) \cdot \mathbf{e}(\pi_C, P_2)^{-1} \end{aligned}$$

This concludes the proof. ▮

Corollary 5.5 *The scheme in Figure 5 for a QAP generator* Rg *satisfies perfect subversion witness indistinguishability.*

Proof. The corollary follows analogously to Corollary 4.3. ▮

DFGK's SSP-Based SNARK

Danezis, Fournet, Groth and Kohlweiss [DFGK14] define *square* span programs, which are described by only one set $\{A_i(X)\}_i$ of polynomials (cf. Definition 3.3). They show how to convert any boolean circuit into an SSP. They construct a zk-SNARK for SSPs with proofs only consisting of 4 elements of an asymmetric bilinear group. Analogously to the SNARK from [BCTV14], their scheme is shown to satisfy subversion zero knowledge by observing that (1) the structure of a CRS can be verified via the bilinear map; (2) the trapdoor τ (which is s in their notation) can be extracted analogously to the SNARK analyzed above; and (3) proofs can be simulated using s by simply following the simulation procedure described in [DFGK14]. (When s is known, the element G^{β} (in their multiplicative notation) can be obtained from the CRS element $G^{\beta t(s)}$ since $t(s) \neq 0$.)

6 Groth's Near-Optimal SNARK

Groth [Gro16] proposed the most efficient zk-SNARK system to date. He drastically reduced the proof size for QAP-based SNARKs to 3 group elements and verification to one equation using 3 pairings. He achieves this by proving soundness directly in the generic-group model. His system is given in Figure 6, to which we added a procedure CRS VERIFICATION defined below.

Theorem 6.1 ([Gro16]) *The scheme in Figure 6 is knowledge-sound against adversaries that only use a polynomial number of generic bilinear group operations. It also has perfect zero knowledge.*

<u>CRS VERIFICATION.</u> On input (R, vk, pk), let $\{a_{i,j}\}$, $\{b_{i,j}\}$, $\{c_{i,j}\}$, $\{z_k\}$ denote the coefficients of $A_i(X)$, $B_i(X)$, $C_i(X)$ and $Z(X)$, respectively, for $0 \leq i \leq m$ and $0 \leq j \leq d-1$ and $0 \leq k \leq d$. .

1. Check $P_1 \neq 0_{\mathbb{G}_1}$ and $P_2 \neq 0_{\mathbb{G}_2}$.
2. Check that $\alpha, \beta, \gamma, \delta$ and $Z(\tau)$ are non-zero: $pk_\alpha \neq 0_{\mathbb{G}_1}$, $pk_\beta \neq 0_{\mathbb{G}_1}$, $vk'_\gamma \neq 0_{\mathbb{G}_2}$, $pk_\delta \neq 0_{\mathbb{G}_1}$, $pk_{Z,0} \neq 0_{\mathbb{G}_1}$
3. Check consistency of pk_H and pk'_H: check $pk_{H,0} = P_1$ and $pk'_{H,0} = P_2$. For $i = 1, \ldots, d$:
$$\mathbf{e}(pk_{H,i}, P) = \mathbf{e}(pk_{H,i-1}, pk'_{H,1}) \qquad \mathbf{e}(P_1, pk'_{H,i}) = \mathbf{e}(pk_{H,i}, P_2)$$
4. Check consistency of the remaining pk elements:
$$\mathbf{e}(P_1, pk'_\beta) = \mathbf{e}(pk_\beta, P_2) \qquad \mathbf{e}(P_1, pk'_\delta) = \mathbf{e}(pk_\delta, P_2)$$
for $i = n+1, \ldots, m$:
$$\mathbf{e}(pk_{K,i}, pk'_\delta) = \mathbf{e}\big(\textstyle\sum_{j=0}^{d-1} a_{i,j} pk_{H,j},\ pk'_\beta\big) \cdot \mathbf{e}\big(pk_\alpha,\ \textstyle\sum_{j=0}^{d-1} b_{i,j} pk'_{H,j}\big) \cdot \mathbf{e}\big(\textstyle\sum_{j=0}^{d-1} c_{i,j} pk_{H,j}, P_2\big)$$
for $i = 0, \ldots, d-2$: $\quad \mathbf{e}(pk_{Z,i}, pk'_\delta) = \mathbf{e}\big(\sum_{j=0}^{d-1} z_j\, pk_{H,j}, pk'_{H,i}\big)$
5. Check consistency of the remaining vk elements: for $i = 0, \ldots, n$:
$$\mathbf{e}(pk_{L,i}, pk'_\gamma) = \mathbf{e}\big(\textstyle\sum_{j=0}^{d-1} a_{i,j} pk_{H,j},\ pk'_\beta\big) \cdot \mathbf{e}\big(pk_\alpha,\ \textstyle\sum_{j=0}^{d-1} b_{i,j} pk'_{H,j}\big) \cdot \mathbf{e}\big(\textstyle\sum_{j=0}^{d-1} c_{i,j} pk_{H,j}, P_2\big)$$
$$vk_T = \mathbf{e}(pk_\alpha, pk'_\beta) \qquad vk'_\delta = pk'_\delta$$
6. If all checks in 1.–5. succeeded then return true and otherwise false.

KEY GENERATION. On input R as in Eq. (8) representing a QAP for an asymmetric group Gr:

1. Sample random group generators $P_1 \leftarrow\!\$\; \mathbb{G}_1^*$ and $P_2 \leftarrow\!\$\; \mathbb{G}_2^*$.
2. Sample random $\alpha, \beta, \gamma, \delta \leftarrow\!\$\; \mathbb{F}^*$ and $\tau \leftarrow\!\$\; \mathbb{F}$ conditioned on $Z(\tau) \neq 0$.
3. Set $vk = (P_1, P_2, vk_T, vk'_\gamma, vk'_\delta, vk_L)$, where

$$vk_T := \mathbf{e}(P_1, P_2)^{\alpha\beta} \qquad vk'_\gamma := \gamma P_2 \qquad vk'_\delta := \delta P_2$$
$$\text{for } i = 0, \ldots, n: \qquad vk_{L,i} := \gamma^{-1}\big(\beta A_i(\tau) + \alpha B_i(\tau) + C_i(\tau)\big) P_1$$

4. Set $pk = (pk_\alpha, pk_\beta, pk'_\beta, pk_\delta, pk'_\delta, pk_H, pk'_H, pk_K, pk_Z)$, where

$$pk_\alpha := \alpha P_1 \qquad pk_\beta := \beta P_1 \qquad pk'_\beta := \beta P_2 \qquad pk_\delta := \delta P_1 \qquad pk'_\delta := \delta P_2$$
$$\text{for } i = 0, \ldots, d-1: \qquad pk_{H,i} := \tau^i P_1 \qquad pk'_{H,i} := \tau^i P_2$$
$$\text{for } i = n+1, \ldots, m: \qquad pk_{K,i} := \delta^{-1}\big(\beta A_i(\tau) + \alpha B_i(\tau) + C_i(\tau)\big) P_1$$
$$\text{for } i = 0, \ldots, d-2: \qquad pk_{Z,i} := \delta^{-1} \tau^i Z(\tau) P_1$$

5. Return $crs := (vk, pk)$.

PROVE. On input R, (vk, pk) and $\vec{s} \in \mathbb{F}^m$ s.t. Eq. (6) is satisfied for some $H(X) \in \mathbb{F}[X]$:

1. **If** (R, vk, pk) **does not pass** CRS VERIFICATION **then return** $\perp$.
2. Compute $H(X)$ such that Eq. (6) is satisfied and let $(h_0, \ldots, h_{d-2}) \in \mathbb{F}^{d-1}$ be its coefficients.
3. Sample $r, s \leftarrow\!\$\; \mathbb{F}$ and define

$$\pi_A := pk_\alpha + \textstyle\sum_{j=0}^{d-1}\big(a_{0,j} + s_i \sum_{i=1}^m a_{i,j}\big) pk_{H,j} + r\, pk_\delta$$
$$\pi'_B := pk'_\beta + \textstyle\sum_{j=0}^{d-1}\big(b_{0,j} + s_i \sum_{i=1}^m b_{i,j}\big) pk'_{H,j} + s\, pk'_\delta$$
$$\pi_{B,\text{aux}} := pk_\beta + \textstyle\sum_{j=0}^{d-1}\big(b_{0,j} + s_i \sum_{i=1}^m b_{i,j}\big) pk_{H,j} + s\, pk_\delta$$
$$\pi_C := \textstyle\sum_{i=n+1}^m s_i\, pk_{K,i} + \sum_{j=0}^{d-2} h_j\, pk_{Z,i} + s\, \pi_A + r\, \pi_{B,\text{aux}} - rs\, pk_\delta$$

4. Return $\pi := (\pi_A, \pi'_B, \pi_C)$.

VERIFY. On input R, vk, $\vec{x} \in \mathbb{F}^n$ and proof $\pi \in \mathbb{G}_1^2 \times \mathbb{G}_2$:

1. Compute $vk_x := vk_{L,0} + \sum_{i=1}^n x_i vk_{L,i}$.
2. Return true if and only if the following holds: $\mathbf{e}(\pi_A, \pi'_B) = vk_T + \mathbf{e}(vk_x, vk'_\gamma) + \mathbf{e}(\pi_C, vk'_\delta)$

Figure 6: Groth's SNARK [Gro16] with CRS verification (in bold)

CRS verifiability. Let $\tau, \alpha, \beta, \gamma, \delta$ denote the logarithms of $pk_{H,1}$, pk_α, pk_β, vk'_γ, pk_δ. By Check 2. in CRS VERIFICATION, $\alpha, \beta, \gamma, \delta, Z(\tau)$ are non-zero. It follows by inspection that if all checks in 3.–5. pass then the remaining elements of pk and vk are correctly computed.

Trapdoor extraction. Let X be a CRS subvertor that outputs (vk, pk). Define $\mathsf{X}'(1^\lambda; r)$ that runs $(vk, pk) \leftarrow \mathsf{X}(1^\lambda; r)$, parses the output as above and returns $(P_1, pk_{H,1}, pk_{H,2}, P_2, pk'_{H,1})$. For a valid CRS this corresponds to $(P_1, \tau P_1, \tau^2 P_1, P_2, \tau P_2)$ for some $P_1 \in \mathbb{G}_1$, $P_2 \in \mathbb{G}_2$ and $\tau \in \mathbb{F}$. By SKE there exists a PT algorithm $\mathsf{E}_{\mathsf{X}'}$ which from a valid tuple extracts τ with overwhelming probability.

Define another algorithm $\mathsf{X}''(1^\lambda; (r, r'))$ that runs $(vk, pk) \leftarrow \mathsf{X}(1^\lambda; r)$ and then extracts $\tau \leftarrow$

$\mathsf{E}_{\mathsf{X}'}(1^\lambda, r; r')$, computes $Z(\tau)$ (which is non-zero in a valid CRS) and sets $P_1' := Z(\tau)^{-1}\, pk_{Z,0}$ (which for a valid CRS yields $P_1' = \delta^{-1} P_1$). Finally, X'' returns $(P_1', P_1, pk_\delta, P_2, pk_\delta')$. For a valid CRS this corresponds to $\big(P_1', \delta P_1', (\delta^2 P_1', P_2, \delta P_2\big)$. By SKE there exist a PT algorithm $\mathsf{E}_{\mathsf{X}''}$ that on input $r'' = (r, r')$ returns δ with overwhelming probability.

Using $\mathsf{E}_{\mathsf{X}'}$ and $\mathsf{E}_{\mathsf{X}''}$, we define the CRS simulator $\mathsf{S.crs}$ as follows: On input 1^λ do the following:

- Sample randomness for X and $\mathsf{E}_{\mathsf{X}'}$: $r \leftarrow_\$ \{0,1\}^{\mathsf{X.rl}(\lambda)}$; $r' \leftarrow_\$ \{0,1\}^{\mathsf{E}_{\mathsf{X}'}.\mathsf{rl}(\lambda)}$
- Run $(vk, pk) \leftarrow \mathsf{X}(1^\lambda; r)$
- If (R, vk, pk) verifies then $\tau \leftarrow \mathsf{E}_{\mathsf{X}'}(1^\lambda, r; r')$ and $\delta \leftarrow_\$ \mathsf{E}_{\mathsf{X}''}(1^\lambda, (r, r'))$,
  else $(\tau, \delta) \leftarrow (\bot, \bot)$
- Return $((vk, pk), r, (\tau, \delta))$

Remark 6.2 Proof simulation is defined in [Gro16] using the full randomness of the CRS and does not work with the trapdoor (τ, δ), as the simulator requires α and β, which SKE does not allow to extract. Note that it is impossible to extract α, since a valid CRS can be computed without knowing α: obliviously sample a random generator $pk_\alpha \leftarrow_\$ \mathbb{G}_1^*$ and then compute vk_T and, for all i, $vk_{L,i}$ and $pk_{K,i}$ from pk_α. In the following we show how to simulate a proof without knowledge of α and β.

Proof simulation. Given (vk, pk), trapdoor (τ, δ) and a statement $x \in \mathbb{F}^n$, the proof simulator $\mathsf{S.pf}$ does the following:

1. If $(\tau, \delta) = (\bot, \bot)$ then return $\bot$.
2. Choose $a, b \leftarrow_\$ \mathbb{F}$ and define the proof $\pi := (\pi_A, \pi_B', \pi_C)$ as follows

$$\pi_A := aP_1 + pk_\alpha \qquad\qquad \pi_B' := bP_2 + pk_\beta'$$
$$\pi_C := \delta^{-1}\big(ab - C_0(\tau) - \textstyle\sum_{i=1}^n x_i C_i(\tau)\big)P_1 + \delta^{-1}\big(b - B_0(\tau) - \textstyle\sum_{i=1}^n x_i B_i(\tau)\big)pk_\alpha$$
$$+\, \delta^{-1}\big(a - A_0(\tau) - \textstyle\sum_{i=1}^n x_i A_i(\tau)\big)pk_\beta$$

Theorem 6.3 *Let* Rg *be a QAP generator defining a bilinear-group generator* aGen *for which SKE holds. Then Groth's SNARK [Gro16] with CRS verification (Figure 6) for* Rg *satisfies subversion zero knowledge.*

Proof. Let E denote the event that (R, vk, pk) passes CRS verification but either $\mathsf{E}_{\mathsf{X}'}$ or $\mathsf{E}_{\mathsf{X}''}$ fails to extract τ and δ. Since a correct (vk, pk) satisfies $\mathbf{e}(pk_{H,1}, P_2) = \mathbf{e}(P_1, pk_{H,1}')$ as well as $\mathbf{e}(pk_{H,2}, P_2) = \mathbf{e}(pk_{H,1}, pk_{H,1}')$, by SKE (Definition 2.15), the probability that $\mathsf{E}_{\mathsf{X}'}$ fails when X' outputs $(P_1, pk_{H,1}, pk_{H,2}, P_2, pk_{H,1}')$ is negligible. A correct CRS also satisfies both $\mathbf{e}(P_1, P_2) = \mathbf{e}(Z(\tau)^{-1} pk_{Z,0}, pk_\delta')$ and $\mathbf{e}(pk_\delta, P_2) = \mathbf{e}(P_1, pk_\delta')$, thus again by SKE, the probability that $\mathsf{E}_{\mathsf{X}''}$ fails when X'' outputs $\big(Z(\tau)^{-1}\, pk_{Z,0}, P_1, pk_\delta, P_2, pk_\delta'\big)$ is also negligible. By a union bound, the probability of E is thus negligible.

It now suffices to show that, conditioned on E not happening, game S-ZK when $b = 0$ is distributed as game S-ZK when $b = 1$. If (vk, pk) fails verification then $(\tau, \delta) = (\bot, \bot)$ and both the prover and the proof simulator return $\bot$.

If (vk, pk) verifies then we show that the outputs of the prover and the proof simulator are distributed equivalently. Above we argued that for some non-zero $\alpha, \beta, \gamma, \delta$ and τ with $Z(\tau) \neq 0$ we have that vk and pk are defined as in 3. and 4. in Key generation.

Since for a valid CRS both pk_δ and pk'_δ are non-zero, for honestly generated proofs the elements rpk_δ in π_A, and spk'_δ in π'_B, make π_A and π'_B uniformly random. For fixed vk, π_A and π'_B, the verification equation uniquely determines π_C, since $vk'_\delta \neq 0$.

In a simulated proof π_A and π'_B are also uniformly random, so it suffices to show that the simulated π_C satisfies the verification equation:

$$
\begin{aligned}
&\mathbf{e}(\pi_C, vk'_\delta) = \\
&\quad = \mathbf{e}\Big(\Big(ab - C_0(\tau) - \sum x_i C_i(\tau) + \alpha\big(b - B_0(\tau) - \sum x_i B_i(\tau)\big) + \beta\big(a - A_0(\tau) - \sum x_i A_i(\tau)\big)\Big) P_1,\ P_2\Big) \\
&\quad = \mathbf{e}(abP_1, P_2) + \mathbf{e}(a\beta P_1, P_2) + \mathbf{e}(\alpha b P_1, P_2) + \mathbf{e}(\alpha\beta P_1, P_2) - \mathbf{e}(\alpha\beta P_1, P_2) \\
&\qquad\quad - \mathbf{e}\big(\big(\beta A_0(\tau) + \sum x_i \beta A_i(\tau) + \alpha B_0(\tau) + \sum x_i \alpha B_i(\tau) + C_0(\tau) + \sum x_i C_i(\tau)\big) P_1,\ P_2\big) \\
&\quad = \mathbf{e}(\pi_A, \pi'_B) - vk_T - \mathbf{e}(vk_x, vk'_\gamma)
\end{aligned}
$$

This concludes the proof. ∎

Corollary 6.4 *Groth's SNARK [Gro16] with CRS verification for a QAP generator* Rg *(Figure 6) satisfies perfect subversion witness indistinguishability.*

Proof. The corollary follows analogously to Corollary 5.5. ∎

Acknowledgments

The author would like to thank Mihir Bellare and Rosario Gennaro for helpful discussions and the anonymous reviewers for PKC'18 for their valuable comments. The author is supported by the French ANR EfTrEC project (ANR-16-CE39-0002).

References

[ABLZ17] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, 2017.

[AF07] Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 118–136. Springer, 2007.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, 2012.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, 2013.

[BCG+14a] 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, 2014.

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

[BCG+15] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE Computer Society Press, 2015.

[BCI+10] Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 237–254. Springer, 2010.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, 2013.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, 2014.

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von Neumann architecture. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security Symposium*, pages 781–796. USENIX Association, 2014.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, 1988.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, 2016.

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420. Springer, 1993.

[BGG17] Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. Cryptology ePrint Archive, Report 2017/602, 2017. http://eprint.iacr.org/2017/602.

[BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 273–289. Springer, 2004.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, 1993.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, 2006.

[BSBC+17] Eli Ben-Sasson, Iddo Bentov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer, and Madars Virza. Computational integrity with a public random string from quasi-linear PCPs. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 551–579. Springer, 2017.

[CGGN17] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 17*, pages 229–243. ACM Press, 2017.

[CNE+14] Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the practical exploitability of Dual EC in TLS implementations. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security Symposium*, pages 319–335. USENIX Association, 2014.

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, 1992.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, 2014.

[FKL17] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. Cryptology ePrint Archive, Report 2017/620, 2017. http://eprint.iacr.org/2017/620.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, 1990.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.

[Gen04] Rosario Gennaro. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 220–236. Springer, 2004.

[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, 2013.

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

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, 2006.

[GOS06b] 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, 2006.

[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, 2006.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, 2010.

[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, 2016.

[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, 2008.

[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, 2011.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 408–423. Springer, 1998.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, 2012.

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

[Nak09] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. http://bitcoin.org/bitcoin.pdf.

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

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

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, 1997.

[SvdW06] Andrew Shallue and Christiaan van de Woestijne. Construction of rational points on elliptic curves over finite fields. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *ANTS-VII*, volume 4076 of *LNCS*, pages 510–524. Springer, 2006.

[Zca] Zcash. http://z.cash.