

# WI Is Not Enough: Zero-Knowledge Contingent (Service) Payments Revisited

GEORG FUCHSBAUER<sup>1</sup>

## Abstract

While fair exchange of goods is known to be impossible without assuming a trusted party, smart contracts in cryptocurrencies forgo such parties by assuming trust in the currency system. They allow a seller to sell a digital good, which the buyer will obtain if and only if she pays. *Zero-knowledge contingent payments* (zkCP) show that, despite the limited expressiveness of its scripting language, this is even possible in Bitcoin by using zero-knowledge proofs.

At CCS’17, Campanelli, Gennaro, Goldfeder and Nizzardo showed that the zkCP protocol was flawed, in that the buyer could obtain information about the good without paying. They proposed countermeasures to repair zkCP and moreover observed that zkCP cannot be used when a service is sold. They introduce the notion of ZK contingent payments *for services* and give an instantiation based on a witness-indistinguishable (WI) proof system.

We show that the main countermeasures they proposed are not sufficient and present an attack against their fixed zkCP scheme. We also show that their realization of zkCP for services is insecure, as the buyer could learn the desired information (i.e., whether the service was provided) without paying; in particular, we show that WI of the used proof system is not enough.

**Keywords:** ZK contingent payments, Bitcoin, zkSNARKs, subversion zero knowledge.

## 1 Introduction

FAIR EXCHANGE IN CRYPTOCURRENCIES. Contingent payments are best explained via an example. Consider Bob, a Sudoku buff, who, after failing to find a solution to a particularly hard Sudoku puzzle, is willing to pay anyone providing him with a solution and announces this on the Internet. Alice solves the Sudoku and would like to claim the promised reward. While she wants to be paid before sending the solution, Bob wants to see the solution before paying. Alice and Bob thus face the problem of *fair exchange*, which cannot be solved without relying on a third party that they both trust [Cle86].

*Smart contracts* [Sza97] in cryptocurrencies provide a way around this impossibility by placing the trust in the currency system. They promise to forgo the need for trusted third parties, such as judges, who can enforce classical contracts. *Ethereum* [But13, Woo14] is a cryptocurrency which supports such smart contracts: it lets a user make a payment and encode, in a Turing-complete language, any conditions that must be met in order for the payment to be executed. Bob could thus write a smart contract that pays anyone who presents a value  $s$ , such that  $V(s)$  holds, where  $V$  is the predicate that verifies a solution to his Sudoku puzzle.

---

<sup>1</sup> Inria, École normale supérieure, CNRS, PSL, France. [georg.fuchsbauer@ens.fr](mailto:georg.fuchsbauer@ens.fr), [www.di.ens.fr/~fuchsba](http://www.di.ens.fr/~fuchsba). The author is supported by ANR project *EfTrEC* (ANR-16-CE39-0002) and by the *MSR-Inria Joint Centre*.

In contrast, *Bitcoin* [Nak09], the first and by far most important cryptocurrency, has only restricted means of expressing payment conditions via its *scripting* language. One simple condition that is supported is expressed by a *hash-locked* transaction [Wik19] bound to a hash value  $y$ : such a payment can be redeemed by anyone who presents a SHA256 preimage of  $y$ , that is, a value  $x$  such that  $\text{SHA}(x) = y$ . It turns out that this elementary functionality (basically) suffices to solve Alice and Bob’s dilemma of who goes first within the Bitcoin system, as it enables *contingent payments* (CP), a simple protocol first suggested by Maxwell [Max11].

In a CP, Alice, the seller, first chooses a key  $k$  for a symmetric-encryption scheme and encrypts the solution  $s$  as ciphertext  $c$  under  $k$ ; she sends  $c$  together with  $y := \text{SHA}(k)$  to Bob, the buyer. Bob makes a *hash-locked* transaction to  $y$ . In order to redeem the payment, Alice must post the preimage  $k$  to the blockchain; but this allows Bob to decrypt  $c$  and obtain the purchased information.<sup>1</sup> To prevent Alice from cheating, Bob requires her to prove that  $c$  really encrypts the desired information under a preimage of  $y$ , for which she can use a cryptographic proof system. To prevent Bob from learning anything before he pays, Alice uses a zero-knowledge (ZK) proof [GMR89], which guarantees that her proof does not leak anything about the values  $s$  and  $k$  she used.

While the Sudoku example is instructive, *zero-knowledge contingent payments* (zkCP) can be used to sell any information that can be verified by an efficiently computable predicate  $V$  for a payment in Bitcoin; and this in a way so that neither the seller nor the buyer can cheat the other party. Someone could for example offer to pay for the secret key related to a public encryption key or a signature verification key, or to a Bitcoin address.

Security of the zkCP protocol relies on the *soundness* and *zero-knowledge* of the used proof system: the former requires that only true statements can be proved (and thus Alice must really send an encryption of a solution that can be decrypted using a preimage of  $y$ ), whereas ZK guarantees that Bob does not obtain any information before Alice has actually revealed the decryption key, which at the same time executes the payment to her. Thus, either Alice gets paid and Bob receives the digital good, or both do not obtain anything. This is precisely the definition of fairness.

zkSNARKs. While ZK proofs have been studied for a long time, until recently there were no practical schemes which could be used for zkCPs related to arbitrary predicates  $V$ . The development of *succinct non-interactive arguments* (SNARGs) and *SNARGs of knowledge* (SNARKs) [GGPR13, PHGR13, BCTV14, DFGK14, Gro16] has changed this. SNARKs are a very efficient form of *non-interactive* proof systems. Non-interactive ZK proofs [BFM88] require trusted parameters in the form of a so-called *common reference string* (CRS), to which the prover and the verifier have access [GO94]. Both soundness and ZK of the proof system rely on the fact that the CRS was computed by a trusted party, thus neither Alice nor Bob. It seems thus that for the practical realization of zkCP, we are back to square one, as the initial goal was to avoid trusted third parties.

Soundness of all zkSNARK systems from the literature relies on strong cryptographic assumptions (so-called *knowledge* assumptions), for which there is evidence that they might be unavoidable [GW11, BCCT12]. Moreover, soundness of these schemes completely breaks down when the CRS is not trusted, as the creator of the CRS can prove arbitrary false statements. Zero knowledge, on the other hand, is perfect, meaning that proofs do not reveal any information, even to an unbounded adversary.

---

<sup>1</sup>To simplify our exposition, we ignore the fact that in the actual implementation Bob’s transaction is slightly more complex: it also contains Alice’s address (so no-one—in particular the miner that adds the transaction to the blockchain—can claim the reward by using Alice’s hash preimage); and it lets Bob claim back his money if the transaction was not redeemed after some predetermined time.

As in zkCP the buyer is the party that values soundness of the proof, the zkCP protocol [Max11] specifies that the CRS for the zkSNARK is created by the buyer. (Presumably, this was considered OK, since the proofs satisfy perfect ZK.) The zkCP protocol was implemented [Bow16] using the libsnark [BCG<sup>+</sup>14b] implementation of the optimized version [BCTV14] of the Pinocchio SNARK [PHGR13], which is derived from the original SNARK [GGPR13].

THE CGGN ATTACK AND SUBVERSION RESISTANCE. At CCS 2017 Campanelli, Gennaro, Goldfeder and Nizzardo (CGGN) [CGGN17] showed that the implementation of zkCP [Bow16] is insecure. They show that the buyer, by sending a maliciously formed CRS, can learn one bit of information about the offered Sudoku solution without paying, which clearly violates the fairness of the protocol. In particular, they show how to set up the CRS in a way that leads to proofs that are still valid, but which reveal whether cell  $(x, y)$  of the Sudoku solution contains value  $z$ , where  $x, y$  and  $z$  can be chosen by the malicious buyer when setting up the CRS.

The reason this attack does not violate perfect ZK is that this notion assumes proofs are computed using an *honestly generated* CRS, which in the zkCP protocol is not necessarily the case. The security notion that the used SNARK must satisfy for the zkCP protocol to be secure is in some sense stronger than perfect ZK, and is called *subversion zero knowledge* [BFS16]. It requires that proofs do not reveal any information even when they were computed under a CRS that was maliciously set up in any arbitrary way.

Fuchsbaauer [Fuc18] investigated the subversion resistance of the most important SNARK systems in the literature, including the SNARK [BCTV14] used for the zkCP implementation [Bow16].<sup>2</sup> He shows that these schemes satisfy the notion of subversion ZK if before using a CRS, the prover checks that all its elements are well-formed. Subversion ZK can then be proved under a *knowledge assumption*, the type of assumption that already underlies the soundness of SNARKs (knowledge soundness).

He also shows that, when provers check CRS well-formedness, *subversion witness indistinguishability* holds unconditionally. Witness indistinguishability (WI) is a weaker notion than ZK: whereas ZK guarantees that nothing is revealed about the *witness* used in a proof (in the Sudoku example, the witness consists of the solution  $s$  and the secret key  $k$ ), WI guarantees that proofs using different witnesses are indistinguishable.

In the SNARK systems analyzed in [Fuc18] the CRS consists of elements from a *bilinear group* and the available bilinear map (*pairing*) can be used to perform these consistency checks (see Figure 3 for such checks). The authors [CGGN17] of the attack against zkCP have implemented these consistency checks and performing them for the Sudoku example implementation would take the prover one hour, which reduces the practicality of the zkCP protocol.

THE CGGN COUNTERMEASURES. While using a SNARK system that satisfies subversion ZK eliminates the security issues of zkCP, this fix is not cheap. The authors [CGGN17] thus propose alternative countermeasures which only rely on *subversion witness indistinguishability*. They observe that WI is not sufficient for the Sudoku example, since a solution  $s$  is unique and even a proof that reveals  $s$  would satisfy WI; however, there could be contexts for which WI is sufficient. In particular, they argue that this is the case for their protocol for *ZK contingent service payments* (see below), which they then adapt to give a new implementation of zkCP. Consequently, the used SNARK system would only have to be subversion-witness-indistinguishable (and would thus also not rely on strong computational hardness assumptions [Fuc18]).

---

<sup>2</sup>Abdolmaleki, Baghery, Lipmaa and Zajac [ABLZ17] also presented a subversion-ZK SNARK scheme by modifying Groth’s scheme [Gro16].

To make a SNARK scheme subversion-WI, Campanelli et al. [CGGN17] propose to have the prover perform some “*minimal checks*” on the CRS, an idea that goes back to the first SNARK scheme [GGPR13]. These checks are much less costly than the checks required for subversion ZK [Fuc18]. Implementing the latter without any optimizations would require the prover to evaluate  $18m - 2n + 2d + 76$  pairings, where the values  $m$ ,  $n$  and  $d$  come from the representation of the proved statement as an arithmetic circuit:  $m$  is the total number of wires,  $n$  is the number of input wires and  $d$  is the number of multiplication gates. (See Section 3.1 for details on arithmetic circuits.)

In stark contrast, the proposed *minimal checks* consist in verifying that  $d + 8$  group elements from the CRS are different from the neutral element of the bilinear group. For their Sudoku zkCP implementation, the prover’s running time is less than a minute even after adding these minimal checks.

Our first finding in revisiting [CGGN17] is that the proposed checks do not prevent all attacks; in particular (Section 4.2):

- We present a CRS-subversion attack against witness indistinguishability of the SNARK scheme implementing the *minimal checks* from [CGGN17]. The attack extends to any variant that does not perform (most of) the consistency checks from [Fuc18].

Despite their simplicity, the minimal checks provide a strong guarantee: even under an arbitrarily subverted CRS, proofs computed using different witnesses are indistinguishable *as long as they are valid* (in that they pass verification). But this still leaves room for attacks.

The CGGN attack against a non-verified CRS modifies the CRS elements in a clever way so that proofs computed under it are valid, but they reveal one bit  $s_j$  of the witness. In contrast, our attack also works against provers performing the minimal checks on the CRS. It only changes one CRS element, which then leads to proofs that are valid if  $s_j = 0$  and invalid if  $s_j = 1$ . From the validity of the proof, the subverter thus learns the value of  $s_j$ .

Subversion WI restricted to *valid* proofs does not yield a secure (fair) zkCP protocol: even if the seller does not send the proof if it detects that it is invalid (which is actually the case in the zkCP implementation [Bow16]), the buyer can from this behavior deduce the value of the bit  $s_j$ .

As we show that the modification of any (of the majority) of CRS elements enables this attack, verification of the CRS is necessary to defend against it; the expensive checks for subversion ZK [Fuc18] are thus even required for subversion WI.

Another countermeasure proposed in [CGGN17] is to create the CRS using secure two-party computation between the seller (prover) and the buyer (verifier). The SNARK parameters used in the cryptocurrency *Zcash* [BCG<sup>+</sup>14a, Zca] have been computed using secure multi-party computation [BCG<sup>+</sup>15, BGG18]; but the main motivation for this was to defend against the known vulnerability of soundness under CRS subversion. (The *Zcash* parameters guarantee soundness as long as there was at least one honest participant in the generation protocol.) In zkCP, soundness is only of concern to the buyer and it is him that computes the CRS.

Our subversion attacks against WI show that in any joint computation of the CRS, the prover must be convinced of the well-formedness of the CRS elements, so the efficiency gains might be small compared to the prover simply checking well-formedness on her own, as for subversion ZK.<sup>3</sup>

Campanelli et al.’s preferred countermeasure against their attack is to use ZK contingent payments *for services*, a notion they introduce and which is the basis for their new implementation of zkCP [CGGN17]. Before discussing this notion, let us mention that there is an alternative realization of zkCP over Bitcoin by Banasik et al. [BDM16], which uses an interactive protocol between

---

<sup>3</sup>A way to avoid computation of pairings in the consistency checks would be to have the CRS generator additionally provide Schnorr-type proofs [Sch91, FS87] of CRS consistency in the random-oracle model [BR93]. While this might speed up CRS verification, it would result in a blow-up of the CRS length.

the seller and the buyer (the protocol is claimed to be “*vulnerable to the so-called mauling problem*” in [CGGN17]). In Ethereum, a zkCP implementation was given by Tramèr et al. [TZL<sup>+</sup>17] and a fair exchange protocol avoiding zero-knowledge proofs was presented by Dziembowski, Eckey and Faust [DEF18].

ZK CONTINGENT SERVICE PAYMENTS. Campanelli et al. [CGGN17] observe that when the seller offers a *service* rather than information, then the zkCP protocol cannot be used for fair payments; even if the fact that the service has been provided can be expressed as the seller knowing some *proof*  $s$  that satisfies a predicate  $V$ . The authors illustrate this by considering a cloud-storage company that offers *proofs of retrievability* as an extra service. These proofs allow the company to convince the customers that their complete data is currently stored. Sticking to our running example, we could also imagine that Bob creates his own Sudoku puzzles and just wants to find out whether they are *solvable*.

The problem with using zkCP in such scenarios is that the buyer is not really interested in the value  $s$  (a proof or a solution); he only cares about the fact that the seller *knows* that value: if the company knows a proof, it must be that it still stores the data; if Alice knows a Sudoku solution then the Sudoku is solvable. Now if in the first step of the zkCP protocol the seller proves knowledge of  $s$ , then the buyer can abort without paying, since he now has the desired information. The problem is that (in the cloud-storage example) a proof of knowledge of a proof is a proof; and a proof of knowledge of a Sudoku solution proves that a solution exists.

To remedy this, the authors [CGGN17] suggest *ZK contingent service payments*, a protocol derived from the ideas of zkCP. In zkCSP, the seller sends a hash value  $y$  and makes a proof of the following statement:

*Either I know a value  $s$  that satisfies  $V$  and I know a SHA256 preimage of  $y$ ; or I don't know a satisfying value and (with overwhelming probability) I do not know a SHA256 preimage of  $y$  either.* Thus, either the seller knows a “proof”  $s$  and a preimage, which it can then use to redeem the hash-locked transaction the buyer will make; or the seller *cannot* redeem the payment. If the proof does not reveal which of the two is the case, the buyer will only find out if the seller knew  $s$  once the payment is redeemed, which yields fairness.

The concept of zkCSP lets a seller sell some assurance to the buyer and it applies to many settings, in particular when these are decentralized. For example [CGGN17], a cryptocurrency exchange could offer, for a fee, to prove its solvency using the *Provisions* protocol [DBB<sup>+</sup>15], and payment for this service could be fairly implemented using zkCSP.

In addition to the protocol, the authors also introduce a security model for zkCSP [CGGN17], which consists of two notions: *Extraction* means that if the seller claims the money, she must know a value  $s$  satisfying  $V$ . *Zero-knowledge* for zkCSP is defined as follows: for any malicious buyer, his view of the interaction with an honest seller can be simulated without knowledge of the seller's input (be it a valid  $s$  or not). While (as for the original zkCP) the zkCSP protocol achieves this notion if the used proof system is subversion-ZK, the authors claim that subversion WI is enough.<sup>4</sup> Our second result is that a subversion-WI proof does not protect the seller in the zkCSP scheme [CGGN17]; in particular (Theorem 5.1):

- We present a proof system satisfying subversion WI, which, when used in the zkCSP protocol, lets the buyer learn whether the seller knows  $s$  without paying.

---

<sup>4</sup>The intuitive argument is that a simulator (which does not have a satisfying  $s$ ) can compute an honest proof by using the second clause of the either/or statement. WI should then imply that this looks like a proof using a satisfying witness  $s$ . However, in their proof sketch the authors do not show how to simulate a redemption payment made by a seller that uses a satisfying  $s$ .

Moreover, even if in the protocol the CRS is created by a trusted party, (regular) WI is not sufficient, as the above result can be easily adapted. The attack against zkCSP is in some sense worse than the one against zkCP. Whereas in the latter the attacker only learns one bit of the sold information, in a flawed zkCSP protocol, the buyer learns the full information (which only consists of one bit).

ZK CONTINGENT PAYMENTS FROM ZKCSP. The suggested fix for the attacked zkCP protocol is to base zkCP on the ideas of their zkCSP protocol [CGGN17]. The seller sends a hash value  $y$  and a ciphertext  $c$  and makes a proof of the following statement:

*Either I know a SHA256 preimage  $k$  of  $y$  which decrypts  $c$  to a satisfying value  $s$ ; or I do not know a SHA256 preimage of  $y$ .*

We extend our result from above and show that subversion witness indistinguishability of the used proof system does not suffice in the zkCP-from-zkCSP protocol for predicates  $V$  with *unique* solutions, and thus for the Sudoku example; in particular (Theorem 5.4):

- We present a subversion-WI proof system, which, when used in zkCP-from-zkCSP for unique solutions, lets the buyer learn the value  $s$  (entirely) without paying.

RELEVANCE. We believe that it is crucial that cryptographic schemes undergo a thorough security analysis, and the best methodology available are rigorous proofs in the provable-security framework. This is particularly important in a decentralized context where there might be no possibility of recourse to courts of justice if something goes wrong (recall the “The DAO” incident that led to a fork of the Ethereum blockchain), which is an inherent feature of the “code is law” [Les00] approach of decentralized applications.

Another striking example of the consequences of a missing security proof for a scheme described in a research paper is the SNARK system underlying Zcash. The parameter-generation protocol [BGG18] for Zcash was implemented based on the description in [BCTV14, eprint May 2015], against which Gabizon [Gab19] later found an attack. He discovered that the public transcript of the generation protocol revealed values that can be used to create proofs of false statements, which in turn enables unlimited creation of Zcash. Since Zcash is an anonymous cryptocurrency that hides the payment details, it is not even clear whether such an attack was mounted before the parameters were updated prior to the disclosure of the bug.

The blockchain/cryptocurrency space today sees a dizzyingly fast transfer of cutting-edge research results to deployed technology, which can rapidly find itself safeguarding millions of dollars. This makes rigorous analysis of the protocols that potentially underly tomorrow’s technology all the more indispensable.

## 2 Definitions

We use the notation from [BFS16, Fuc18] and start with recalling their formal definitions for subversion-secure proof systems.

### 2.1 Notation

We denote by  $\lambda \in \mathbb{N}$  the security parameter and by  $1^\lambda$  its unary representation. Algorithms are randomized unless otherwise indicated. “PT” stands for “polynomial time”, whether for randomized or deterministic algorithms. By  $y \leftarrow A(x_1, \dots, x_n; r)$  we denote the operation of running algorithm  $A$  on inputs  $x_1, \dots, x_n$  and coins  $r$  and letting  $y$  denote the output. By  $y \leftarrow^s A(x_1, \dots, x_n)$ , we denote letting  $y \leftarrow A(x_1, \dots, x_n; r)$  for random  $r$ . If  $S$  is a finite set then  $s \leftarrow^s S$  denotes picking

an element uniformly from  $S$  and assigning it to  $s$ . We denote by  $[A(x_1, \dots, x_n)]$  the set of points that have positive probability of being output by  $A$  on inputs  $x_1, \dots, x_n$ .

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

## 2.2 NP Relations and NI Systems

NP RELATIONS. Consider  $R: \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{\text{true}, \text{false}\}$ . For  $x \in \{0, 1\}^*$ , the *witness set* of  $x$  is  $R(x) := \{w \mid R(x, w) = \text{true}\}$ . The *language* associated to  $R$  is  $L(R) := \{x \mid R(x) \neq \emptyset\}$ . Relation  $R$  is an NP relation if it is PT and there is a polynomial  $P_R$  s.t. for all  $x$  and all  $w \in R(x)$  we have  $|w| \leq P_R(|x|)$ , where  $|w|$  denotes the length of  $w$ . As customary for SNARKs we will consider relations output by a PT *relation generator*  $\text{Rg}$ .<sup>5</sup>

NI SYSTEMS. A non-interactive (NI) system  $\Pi$  for relation generator  $\text{Rg}$  consists of PT algorithms  $\Pi.\text{Pg}$ ,  $\Pi.\text{P}$  and  $\Pi.\text{V}$ . A common reference string  $\text{crs}$  for relation  $R$  is generated via  $\text{crs} \leftarrow_s \Pi.\text{Pg}(R)$ . Given  $x$  and  $w \in R(x)$ , the prover generates  $\pi \leftarrow_s \Pi.\text{P}(R, \text{crs}, x, w)$ , where  $\pi$  proves that  $x \in L(R)$ . The verifier runs  $\Pi.\text{V}(R, \text{crs}, x, \pi)$ , which yields an output in  $\{\text{true}, \text{false}\}$  indicating whether  $\pi$  is a valid proof that  $x \in L(R)$ . We require perfect completeness, that is, for all  $\lambda \in \mathbb{N}$ , all  $R \in [\text{Rg}(1^\lambda)]$ , all  $\text{crs} \in [\Pi.\text{Pg}(R)]$ , all  $x \in L(R)$ , all  $w \in R(x)$  and all  $\pi \in [\Pi.\text{P}(R, \text{crs}, x, w)]$  we have that  $\Pi.\text{V}(R, \text{crs}, x, \pi)$  returns `true`. We assume that  $\Pi.\text{V}$  returns `false` if any of its arguments is  $\perp$ .

## 2.3 Security Notions

SOUNDNESS. The most fundamental notion for a proof system is *soundness*, which in its computational variant requires that it is hard to create a valid proof for any  $x \notin L(R)$ .

*Knowledge soundness* [BG93] is stronger than soundness and 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. (This implies soundness, since for a wrong statement there are no witnesses.) This is the notion satisfied by SNARK (as opposed to SNARG) systems and for the applications considered in this paper standard soundness is not sufficient.

**Definition 2.1 (KSND)** *An NI system  $\Pi$  for relation generator  $\text{Rg}$  is knowledge-sound if for all PT adversaries  $A$  there exists a PT extractor  $E$  such that  $\text{Adv}_{\Pi, \text{Rg}, A, E}^{\text{ksnd}}(\cdot)$  is negligible, where  $\text{Adv}_{\Pi, \text{Rg}, A, E}^{\text{ksnd}}(\lambda) = \Pr[\text{KSND}_{\Pi, \text{Rg}, A, E}(\lambda)]$  and game KSND is specified in Figure 1.*

WI. Witness indistinguishability [FLS90] requires that proofs for the same statement using different witnesses are indistinguishable. This definition [BFS16] lets the adversary adaptively request multiple proofs for statements  $x$  under one of two witnesses  $w_0, w_1$ .

**Definition 2.2 (WI)** *An NI system  $\Pi$  for  $\text{Rg}$  is witness-indistinguishable if  $\text{Adv}_{\Pi, \text{Rg}, A}^{\text{wi}}(\cdot)$  is negligible for all PT adversaries  $A$ , where  $\text{Adv}_{\Pi, \text{Rg}, A}^{\text{wi}}(\lambda) = 2 \Pr[\text{WI}_{\Pi, \text{Rg}, A}(\lambda)] - 1$  and game WI is specified in Figure 1.*

<sup>5</sup>in contrast to generic proof systems, the language proved by SNARKs is not independent of the system setup when it is expressed as an arithmetic circuit, as arithmetic is done modulo the group order of the bilinear group over which the SNARK is defined.

<p><u>GAME KSND<math>_{\Pi, \text{Rg}, A, E}(\lambda)</math></u>  <math>R \leftarrow_{\\$} \text{Rg}(1^\lambda)</math>  <math>\text{crs} \leftarrow_{\\$} \Pi.\text{Pg}(R) ; r \leftarrow_{\\$} \{0, 1\}^{\text{A.t}(\lambda)}</math>  <math>(x, \pi) \leftarrow \text{A}(R, \text{crs}; r)</math>  <math>w \leftarrow_{\\$} \text{E}(R, \text{crs}, r)</math>  Return <math>(\neg R(x, w) \text{ and } \Pi.\text{V}(R, \text{crs}, x, \pi))</math></p>	<p><u>GAME WI<math>_{\Pi, \text{Rg}, A}(\lambda)</math></u> <span style="border: 1px solid black; padding: 2px;"><u>S-WI<math>_{\Pi, \text{Rg}, A}(\lambda)</math></u></span>  <math>b \leftarrow_{\\$} \{0, 1\} ; R \leftarrow_{\\$} \text{Rg}(1^\lambda)</math>  <math>\text{crs} \leftarrow_{\\$} \Pi.\text{Pg}(R) ; st \leftarrow \perp</math>  <span style="border: 1px solid black; padding: 2px;"><math>(\text{crs}, st) \leftarrow_{\\$} \text{A}(R)</math></span>  <math>b' \leftarrow_{\\$} \text{A}^{\text{PROVE}}(R, \text{crs}, st)</math>  Return <math>(b = b')</math>  <u>PROVE<math>(x, w_0, w_1)</math></u>  If <math>\neg R(x, w_0)</math> or <math>\neg R(x, w_1)</math>  then return <math>\perp</math>  <math>\pi \leftarrow_{\\$} \Pi.\text{P}(R, \text{crs}, x, w_b)</math>  Return <math>\pi</math></p>
<p><u>GAME ZK<math>_{\Pi, \text{Rg}, A}(\lambda)</math></u>  <math>b \leftarrow_{\\$} \{0, 1\} ; R \leftarrow_{\\$} \text{Rg}(1^\lambda)</math>  <math>\text{crs}_1 \leftarrow_{\\$} \Pi.\text{Pg}(R)</math>  <math>(\text{crs}_0, td) \leftarrow_{\\$} \Pi.\text{Sim.crs}(R)</math>  <math>b' \leftarrow_{\\$} \text{A}^{\text{PROVE}}(R, \text{crs}_b)</math>  Return <math>(b = b')</math>  <u>PROVE<math>(x, w)</math></u>  If <math>\neg R(x, w)</math> then return <math>\perp</math>  If <math>b = 1</math> then <math>\pi \leftarrow_{\\$} \Pi.\text{P}(R, \text{crs}_1, x, w)</math>  Else <math>\pi \leftarrow_{\\$} \Pi.\text{Sim.pf}(R, \text{crs}_0, td, x)</math>  Return <math>\pi</math></p>	<p><u>GAME S-ZK<math>_{\Pi, \text{Rg}, X, S, A}(\lambda)</math></u>  <math>b \leftarrow_{\\$} \{0, 1\} ; R \leftarrow_{\\$} \text{Rg}(1^\lambda)</math>  <math>r_1 \leftarrow_{\\$} \{0, 1\}^{X.\text{t}(\lambda)} ; \text{crs}_1 \leftarrow X(R; r_1)</math>  <math>(\text{crs}_0, r_0, td) \leftarrow_{\\$} \text{S.crs}(R)</math>  <math>b' \leftarrow_{\\$} \text{A}^{\text{PROVE}}(R, \text{crs}_b, r_b)</math>  Return <math>(b = b')</math>  <u>PROVE<math>(x, w)</math></u>  If <math>\neg R(x, w)</math> then return <math>\perp</math>  If <math>b = 1</math> then <math>\pi \leftarrow_{\\$} \Pi.\text{P}(R, \text{crs}_1, x, w)</math>  Else <math>\pi \leftarrow_{\\$} \text{S.pf}(R, \text{crs}_0, td, x)</math>  Return <math>\pi</math></p>

Figure 1: Games defining knowledge-soundness, witness-indistinguishability (ignoring the boxes), subversion-WI (including the boxes), zero knowledge (ZK) and subversion-ZK for an NI system  $\Pi$ .

*Subversion WI* [BFS16] demands that even when the adversary creates a CRS in any way, it can still not decide which of two witnesses of its choice were used to create a proof. The adversary first outputs a CRS  $\text{crs}$  and some state information. Its second stage receives this state and is otherwise defined like for the standard WI game.

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

*An NI system  $\Pi$  is perfect S-WI if  $\text{Adv}_{\Pi, \text{Rg}, A}^{\text{s-wi}}(\cdot) \equiv 0$  for all  $A$ .*

ZERO KNOWLEDGE. This notion [GMR89] means that no information about the witness is leaked by the proof. Formally, there must exist a simulator, which can create a CRS for which it can then compute proofs without using a witness. This CRS and proofs are indistinguishable from real ones. In the security game, the distinguisher  $A$  can adaptively request proofs by submitting an instance and a witness for it. It receives a proof that is produced either by the honest prover using the witness or by the simulator.

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

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

Subversion ZK [BFS16] considers a CRS subverter  $\mathbf{X}$  that returns an arbitrarily formed CRS. It requires that for any such  $\mathbf{X}$  there exists a simulator that is able to simulate (1) the full view of the CRS subverter, including its coins, and (2) proofs for adaptively chosen instances without knowing the witnesses. The simulator consists of two parts:  $\text{S.crs}$  returns a CRS, coins for  $\mathbf{X}$  and a simulation trapdoor; and  $\text{S.pf}$  simulates proofs using this trapdoor. The adversary’s task is to decide whether it is given a real CRS and the coins used to produce it, as well as real proofs (case  $b = 1$ ); or whether everything has been simulated (case  $b = 0$ ).

**Definition 2.5 (S-ZK)** An NI system  $\Pi$  for generator  $\text{Rg}$  is subversion-zero-knowledge if for all PT CRS subvertors  $\mathbf{X}$  there exists a PT simulator  $\text{S} = (\text{S.crs}, \text{S.pf})$  such that  $\text{Adv}_{\Pi, \text{Rg}, \mathbf{X}, \mathbf{S}, \mathbf{A}}^{\text{s-zk}}(\cdot)$  is negligible for all PT adversaries  $\mathbf{A}$ , where  $\text{Adv}_{\Pi, \text{Rg}, \mathbf{X}, \mathbf{S}, \mathbf{A}}^{\text{s-zk}}(\lambda) = 2 \Pr[\text{S-ZK}_{\Pi, \text{Rg}, \mathbf{X}, \mathbf{S}, \mathbf{A}}(\lambda)] - 1$  and game S-ZK is specified in Figure 1.

## 2.4 Bilinear Groups

The SNARK scheme [BCTV14, Gab19] we consider is based on asymmetric bilinear groups.

**Definition 2.6** A bilinear-group generator  $\text{Gen}$  is a PT algorithm that takes input a security parameter  $1^\lambda$  and outputs a description of a bilinear group  $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e})$  with the following properties:

- $p$  is a prime of length  $\lambda$ .
- $(\mathbb{G}_1, +)$ ,  $(\mathbb{G}_2, +)$  and  $(\mathbb{G}_T, \cdot)$  are groups of order  $p$ .
- $\mathbf{e}: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  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, group operations and the bilinear map must be efficiently computable, membership of the groups and equality of group elements must be efficiently decidable, and group generators efficiently samplable. We denote the neutral element of  $\mathbb{G}_i$  by  $0_{\mathbb{G}_i}$ .

Fuchsbaauer [Fuc18] proves subversion zero knowledge of the SNARK schemes he analyzed under the “square knowledge of exponent” (SKE) assumption. It states that an adversary  $\mathbf{A}$  which outputs a triple  $(P, sP, s^2P) \in \mathbb{G}_1^3$  must know the values  $s$ . To make such triples verifiable via the pairing, the adversary is required to also output an element  $Q \in \mathbb{G}_2^*$  as well as  $sQ$ . (Elements  $(P, P', P'', Q, Q')$  are of the required form if and only if  $\mathbf{e}(P', Q) = \mathbf{e}(P, Q')$  and  $\mathbf{e}(P', Q') = \mathbf{e}(P'', Q)$ .)<sup>6</sup>

<sup>6</sup>We refer to [Fuc18] for more on SKE and to [BFS16] for why subversion ZK plausibly requires strong assumptions of this type.

## 2.5 Claw-Free Functions

The standard security notion for a cryptographic hash function  $H$  is *collision-resistance*, which requires that it is hard to find two different values  $x_0, x_1$  with  $H(x_0) = H(x_1)$ . Claw-freeness is a similar notion for pairs of functions, and is used in the instantiation of zkCSP [CGGN17] given in Definition 5. Two functions  $H_0$  and  $H_1$  are claw-free if it is hard to find (not necessarily different) values  $x_0, x_1$  such that  $H_0(x_0) = H_1(x_1)$ . A formal asymptotic definition is as follows:

**Definition 2.7** *A function generator  $FGen$  is a PT algorithm that on input a security parameter  $1^\lambda$  returns two descriptions of functions  $H_{0,\lambda}, H_{1,\lambda}$  with image  $\{0, 1\}^\lambda$  that can both be efficiently computed.*

*Generator  $FGen$  is claw-free (in which case  $(H_{0,\lambda}, H_{1,\lambda})$  is called a claw-free pair) if  $\mathbf{Adv}_{FGen,A}^{cf}(\cdot)$  is negligible for all PT adversaries  $A$ , where  $\mathbf{Adv}_{FGen,A}^{cf}(\lambda) = \Pr[CF_{FGen,A}(\lambda)]$  and game  $CF$  is defined as follows:*

```

GAME  $CF_{FGen,A}(\lambda)$ 
 $(H_{0,\lambda}, H_{1,\lambda}) \leftarrow \$ FGen(1^\lambda)$ 
 $(x_0, x_1) \leftarrow \$ A(H_{0,\lambda}, H_{1,\lambda})$ 
Return  $(H_{0,\lambda}(x_1) = H_{1,\lambda}(x_0))$ 

```

## 3 SNARKs

### 3.1 Definitions

We start with a formal definition of SNARKs.

**Definition 3.1 (SNARK)** *An NI system  $\Pi = (\Pi.Pg, \Pi.P, \Pi.V)$  is a succinct non-interactive argument of knowledge for relation generator  $Rg$  if it is complete (Section 2.2) and knowledge-sound (Definition 2.1), and moreover succinct, meaning that for all  $\lambda \in \mathbb{N}$ , all  $R \in [Rg(1^\lambda)]$ , all  $crs \in [\Pi.Pg(R)]$ ,  $x \in L(R)$ ,  $w \in R(x)$  and all  $\pi \in [\Pi.P(1^\lambda, crs, x, w)]$  we have  $|\pi| = \text{poly}(\lambda)$  and  $\Pi.V(1^\lambda, crs, x, \pi)$  runs in time  $\text{poly}(\lambda + |x|)$ .*

Gennaro, Gentry, Parno and Raykova [GGPR13] introduce the notions of *quadratic span programs* and *quadratic arithmetic programs* (QAP), and show how to convert any arithmetic circuit into a QAP (as we recall below).

**Definition 3.2 (QAP)** *A quadratic arithmetic program over a field  $\mathbb{F}$  is a tuple of the form*

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

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

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

*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  $d-1$ ).*

ARITHMETIC CIRCUITS. An arithmetic circuit consists of wires, which can take values from a field  $\mathbb{F}$ , and addition and multiplication gates between several wires, as well as addition-with-constant and multiplication-by-scalar gates. It is transformed into a QAP as follows. First the circuit is transformed into an equivalent circuit with only one type of gate, namely *weighted-sum-multiplication* gates: given left inputs  $v_1, \dots, v_\ell \in \mathbb{F}$  and right inputs  $v'_1, \dots, v'_r$ , such a gate computes  $(\alpha_0 + \sum_{i=1}^\ell \alpha_i v_i)(\beta_0 + \sum_{i=1}^r \beta_i v'_i) \in \mathbb{F}$ . A circuit with  $d$  multiplication gates and any number of gates of other types can easily be transformed into a circuit with  $d + 1$  weighted-sum-multiplication gates.

Letting  $w_1, \dots, w_m$  denote all wires in the (transformed) circuit, we can define  $\alpha_{i,j}$ ,  $\beta_{i,j}$  and  $\gamma_{i,j}$ , so that the  $j$ -th gate can be written as:

$$(\alpha_{0,j} + \sum_{i=1}^m \alpha_{i,j} w_i)(\beta_{0,j} + \sum_{i=1}^m \beta_{i,j} w_i) = \gamma_{0,j} + \sum_{i=1}^m \gamma_{i,j} w_i ,$$

where  $\alpha_{i,j} = 0$  ( $\beta_{i,j} = 0$ ) if  $w_i$  is not among the left (right) inputs of gate  $j$  and  $\gamma_{i,j} = 1$  if  $w_i$  is the output wire of gate  $j$  and  $\gamma_{i,j} = 0$  otherwise.

The polynomial  $Z$  of the QAP is defined by picking distinct values  $r_1, \dots, r_d \in \mathbb{F}$  and setting  $Z(X) = \prod_{j=1}^d (X - r_j)$ . Value  $r_j$  “represents” the  $j$ -th gate of the circuit and the remaining polynomials represent the role of the  $i$ -th wire  $w_i$  among the left ( $A_i$ ) or right ( $B_i$ ) inputs or output wires ( $C_i$ ) of every gate. In particular, the polynomials are computed by interpolation so they satisfy the following: for every  $i = 0, \dots, m$  and  $j = 1, \dots, d$ :

$$A_i(r_j) = \alpha_{i,j} \qquad B_i(r_j) = \beta_{i,j} \qquad C_i(r_j) = \gamma_{i,j}$$

An assignment  $(s_1, \dots, s_m)$  to the wires of the circuit is satisfying if for every gate  $j$ , we have:

$$\begin{aligned} 0 &= (\alpha_{0,j} + \sum_{i=1}^m s_i \alpha_{i,j})(\beta_{0,j} + \sum_{i=1}^m s_i \beta_{i,j}) - (\gamma_{0,j} + \sum_{i=1}^m s_i \gamma_{i,j}) \\ &= (A_0(r_j) + \sum_{i=1}^m s_i A_i(r_j))(B_0(r_j) + \sum_{i=1}^m s_i B_i(r_j)) - (C_0(r_j) + \sum_{i=1}^m s_i C_i(r_j)) =: P(r_j) , \end{aligned}$$

thus the polynomial  $P(X)$  must vanish at  $r_1, \dots, r_d$ , which is implied by the fact that it is divisible by  $Z$  (which is precisely the condition that  $(s_1, \dots, s_m)$  satisfies the QAP).

To capture SNARK constructions for QAPs in the framework for non-interactive proof system from Section 2.2, we consider relation generators  $\text{Rg}$  of the following form:

**Definition 3.3 (QAP generator)** *A QAP relation generator  $\text{Rg}$  is a PT algorithm that on input  $1^\lambda$  returns a relation description of the following form:*

$$\begin{aligned} R = (Gr, n, \vec{A}, \vec{B}, \vec{C}, Z) \quad \text{where } Gr \text{ is a bilinear group whose order } p \text{ defines } \mathbb{F} := \mathbb{Z}_p \text{ and} \\ \vec{A}, \vec{B}, \vec{C} \in (\mathbb{F}^{(d-1)}[X])^{(m+1)}, Z \in \mathbb{F}^{(d)}[X], n \leq m . \end{aligned} \quad (2)$$

For  $x \in \mathbb{F}^n$  and  $w \in \mathbb{F}^{m-n}$  we define  $R(x, w) = \text{true}$  iff there exists  $H(X) \in \mathbb{F}[X]$  so that Eq. (1) holds for  $s := x \parallel w$  (where “ $\parallel$ ” denotes concatenation).

### 3.2 Asymmetric Pinocchio

Pinocchio [PHGR13] is a more efficient variant of the original zkSNARK [GGPR13]. Ben-Sasson et al. [BCG<sup>+</sup>13, BCTV14] further optimized Pinocchio by moving from symmetric to asymmetric bilinear groups, shortening the verification key and reducing the verifier’s work.<sup>7</sup> This is the zkSNARK system that underlies the first version of Zcash.

<sup>7</sup>In [PHGR13],  $\pi_B$  is of the form  $\pi_B := (B_0(\tau) + \sum_{i=n+1}^m s_i B_i(\tau) + \delta_B Z(\tau))\rho_B P_2$  and analogously for  $\pi_C$ , and the verifier has to compute  $\hat{\pi}_B := (B_0(\tau) + \sum_{i=1}^n s_i B_i(\tau))\rho_B P_2$  from the instance  $(s_1, \dots, s_n)$ . In [BCTV14] (see Figure 2),  $\pi_B$  is directly set as  $\pi_B + \hat{\pi}_B$  (and similarly for  $\pi_C$ ).

The description in [BCTV14] has recently been showed not to be sound by Gabizon [Gab19], as the CRS erroneously included elements  $pk'_{A,i} := A_i(\tau)\alpha_A\rho_AP_1$  for  $i = 0, \dots, n$ . The error did not affect the (subversion) zero-knowledge properties of the system or its variant in [Fuc18]. It does also not affect the results in [CGGN17].

Campanelli et al. [CGGN17] show that the protocol from [BCTV14], which underlies the ZK contingent payments implementation, is not subversion-zero-knowledge. Fuchsbauer [Fuc18] showed that adding 4 group elements to the CRS (denoted by  $ck$  in Figure 2) and requiring the prover to perform consistency checks on all CRS elements ( $vk, pk, ck$ ) makes the scheme subversion-zero-knowledge under SKE (see Section 2.4), a knowledge assumption (the type of assumption under which soundness of the scheme was proved).

The protocol given in Figure 2 is from [Fuc18, full version], which includes the fixes required for soundness from [Gab19].

**Theorem 3.4** ([Fuc18]) *The scheme in Figure 2 for a QAP generator  $Rg$  satisfies subversion zero knowledge if SKE holds for the underlying bilinear-group generator  $Gen$ .*

**Corollary 3.5** ([Fuc18]) *The scheme in Figure 2 for a QAP generator  $Rg$  satisfies perfect subversion witness indistinguishability.*

### 3.3 CGGN’s Subversion Attack

Campanelli et al. [CGGN17] show a CRS-subversion attack against the SNARK from [BCTV14] (which is the scheme in Figure 2 except that the prover does *not* perform step 1). Under their subverted CRS, honestly computed proofs verify, but for one particular  $j \in \{1, \dots, m\}$ , the proof reveals the value  $s_j$  used by the prover (which corresponds to one of the wire assignments in the arithmetic circuit expressing the statement).

In particular, *Key generation* in Figure 2 is subverted as follows: the values constituting  $vk$  remain unchanged, except for

$$vk_{IC,i} := 0_{\mathbb{G}_1} \text{ for } i = 0, \dots, n.$$

All values in  $pk$  are set to the zero element of the respective group, except for the following:

$$pk_{B,j} := \rho_B P_2 \qquad pk'_{B,j} := \alpha_B \rho_B P_1 \qquad pk_{K,j} := \beta \rho_B P_1$$

An honestly computed proof for  $\vec{s} \in \mathbb{F}^m$  under this CRS is thus of the following form:

$$\begin{array}{llll} \pi_A := 0_{\mathbb{G}_1} & \pi'_A := 0_{\mathbb{G}_1} & \pi_B := s_j \rho_B P_2 & \pi'_B := s_j \alpha_B \rho_B P_1 \\ \pi_C := 0_{\mathbb{G}_1} & \pi'_C := 0_{\mathbb{G}_1} & \pi_K := s_j \beta \rho_B P_1 & \pi_H := 0_{\mathbb{G}_1} \end{array}$$

This is easily seen to pass verification. The value of  $s_j$  is obtained by checking whether  $\pi_B = 0_{\mathbb{G}_2}$  or  $\pi_B = pk_{B,j}$ .

Since the zkCP instantiation uses this SNARK [BCTV14], the buyer, who creates the CRS, can subvert it as described above and thus, without paying, learn one bit of information about the solution used by the seller.

## 4 No Shortcuts to Subversion WI

### 4.1 Suggested Countermeasures by CGGN

The CGGN attack [CGGN17] leverages the faulty assumption that the used proof system is subversion-resistant, which has been implicitly made in the zkCP protocol. The authors [CGGN17, Section 3.3] propose four possible countermeasures against their attack:

**KEY GENERATION.** On input  $R$  as in Eq. (2) 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
 
$$\begin{aligned} vk_A &:= \alpha_A P_2 & vk_B &:= \alpha_B P_1 & vk_C &:= \alpha_C P_2 \\ vk_\gamma &:= \gamma P_2 & vk_{\beta\gamma} &:= \gamma\beta P_1 & \widehat{vk}_{\beta\gamma} &:= \gamma\beta P_2 & vk_Z &:= Z(\tau)\rho_A\rho_B P_2 \\ \text{for } i = 0, \dots, n : & & vk_{IC,i} &:= A_i(\tau)\rho_A P_1 \end{aligned}$$
5. Set  $pk = (pk_A, pk'_A, pk_B, pk'_B, pk_C, pk'_C, pk_K, pk_H)$  where
 
$$\begin{aligned} \text{for } i = n+1, \dots, m+3 : & & pk_{A,i} &:= A_i(\tau)\rho_A P_1 & pk'_{A,i} &:= A_i(\tau)\alpha_A\rho_A P_1 \\ \text{for } i = 0, \dots, m+3 : & & pk_{B,i} &:= B_i(\tau)\rho_B P_2 & pk'_{B,i} &:= B_i(\tau)\alpha_B\rho_B P_1 \\ & & pk_{C,i} &:= C_i(\tau)\rho_A\rho_B P_1 & pk'_{C,i} &:= C_i(\tau)\alpha_C\rho_A\rho_B P_1 \\ & & pk_{K,i} &:= \beta(A_i(\tau)\rho_A + B_i(\tau)\rho_B + C_i(\tau)\rho_A\rho_B) P_1 \\ \text{for } i = 0, \dots, d : & & pk_{H,i} &:= \tau^i P_1 \end{aligned}$$
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. (1) is satisfied for some  $H'(X) \in \mathbb{F}[X]$ .

1. **If**  $(R, vk, pk, ck)$  **does not pass CRS VERIFICATION as defined in Figure 3 then return**  $\perp$ .
2. Sample  $\delta_A, \delta_B, \delta_C \leftarrow \mathbb{F}$  and define
 
$$\begin{aligned} A(X) &:= A_0(X) + \sum_{i=1}^m s_i A_i(X) + \delta_A Z(X) \\ B(X) &:= B_0(X) + \sum_{i=1}^m s_i B_i(X) + \delta_B Z(X) \\ C(X) &:= C_0(X) + \sum_{i=1}^m s_i C_i(X) + \delta_C Z(X) \end{aligned}$$
3. Compute  $H(X)$  such that  $A(X)B(X) - C(X) = H(X)Z(X)$ ; let  $(h_0, \dots, h_d) \in \mathbb{F}^{d+1}$  be its coefficients.
4. For  $i = 0, \dots, n$  let  $pk_{A,i} := 0$  and  $pk'_{A,i} := 0$
5. Let  $\vec{c} := 1 \parallel \vec{s} \parallel \delta_A \parallel \delta_B \parallel \delta_C \in \mathbb{F}^{m+4}$  and compute (“ $\langle \cdot, \cdot \rangle$ ” denotes the scalar product)
 
$$\begin{aligned} \pi_A &:= \langle \vec{c}, pk_A \rangle & \pi'_A &:= \langle \vec{c}, pk'_A \rangle & \pi_B &:= \langle \vec{c}, pk_B \rangle & \pi'_B &:= \langle \vec{c}, pk'_B \rangle \\ \pi_C &:= \langle \vec{c}, pk_C \rangle & \pi'_C &:= \langle \vec{c}, pk'_C \rangle & \pi_K &:= \langle \vec{c}, pk_K \rangle & \pi_H &:= \langle \vec{h}, pk_H \rangle \end{aligned}$$
6. Return  $\pi := (\pi_A, \pi'_A, \pi_B, \pi'_B, \pi_C, \pi'_C, \pi_K, \pi_H)$ .

**VERIFY.** On input  $R$ ,  $vk$ ,  $\vec{x} \in \mathbb{F}^m$  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  $\pi'_A$ ,  $\pi'_B$ , and  $\pi'_C$ :
 
$$\mathbf{e}(\pi'_A, P_2) = \mathbf{e}(\pi_A, vk_A) \quad \mathbf{e}(\pi'_B, P_2) = \mathbf{e}(vk_B, \pi_B) \quad \mathbf{e}(\pi'_C, P_2) = \mathbf{e}(\pi_C, vk_C)$$
3. Check same coefficients were used via  $\pi_K$ :  $\mathbf{e}(\pi_K, vk_\gamma) = \mathbf{e}(vk_x + \pi_A + \pi_C, \widehat{vk}_{\beta\gamma}) \cdot \mathbf{e}(vk_{\beta\gamma}, \pi_B)$
4. Check QAP is satisfied via  $\pi_H$ :  $\mathbf{e}(vk_x + \pi_A, \pi_B) = \mathbf{e}(\pi_H, vk_Z) \cdot \mathbf{e}(\pi_C, P_2)$
5. If all checks in 2.-4. succeeded then return true and otherwise false.

Figure 2: Asymmetric Pinocchio from [BCTV14] with soundness fixes from [Gab19] and made **sub-version-zero-knowledge** as per [Fuc18].

**CRS VERIFICATION.** 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  $0 \leq i \leq m$  and  $0 \leq j \leq d-1$  and  $0 \leq k \leq d$ .

Perform the following consistency checks:

1. [generators]  $P_1 \neq 0_{\mathbb{G}_1}$  and  $P_2 \neq 0_{\mathbb{G}_2}$
2. [choice of secret values]  $ck_A \neq 0_{\mathbb{G}_2}$ ,  $ck_B \neq 0_{\mathbb{G}_2}$ ,  $vk_\gamma \neq 0_{\mathbb{G}_2}$ ,  $vk_{\beta\gamma} \neq 0_{\mathbb{G}_1}$  and  $vk_Z \neq 0_{\mathbb{G}_2}$
3. [consistency of  $pk_H$ ]  $pk_{H,0} = P_1$ ; For  $i = 1, \dots, d$ :  $\mathbf{e}(pk_{H,i}, P_2) = \mathbf{e}(pk_{H,i-1}, ck_H)$
4. [consistency of  $pk_A, pk'_A, pk_B, pk'_B$ ] For  $i = n+1, \dots, m+3$ :  $\mathbf{e}(pk_{A,i}, P_2) = \mathbf{e}(\sum_{j=0}^{d-1} a_{i,j} pk_{H,j}, ck_A)$   
 $\mathbf{e}(pk'_{A,i}, P_2) = \mathbf{e}(pk_{A,i}, vk_A)$   
and for  $i = 0, \dots, m+3$ :  $\mathbf{e}(P_1, pk_{B,i}) = \mathbf{e}(\sum_{j=0}^{d-1} b_{i,j} pk_{H,j}, ck_B)$   
 $\mathbf{e}(pk'_{B,i}, P_2) = \mathbf{e}(vk_B, pk_{B,i})$
5. [consistency of  $ck_C$ ]  $\mathbf{e}(pk_{A,m+1}, ck_B) = \mathbf{e}(\sum_{j=0}^d z_j pk_{H,j}, ck_C)$
6. [consistency of  $vk$ ]  $\mathbf{e}(vk_{\beta\gamma}, P_2) = \mathbf{e}(P_1, \widehat{vk}_{\beta\gamma})$ ;  $\mathbf{e}(P_1, vk_Z) = \mathbf{e}(\sum_{j=0}^d z_j pk_{H,j}, ck_C)$   
and for  $i = 0, \dots, n$ :  $\mathbf{e}(vk_{IC,i}, P_2) = \mathbf{e}(\sum_{j=0}^{d-1} a_{i,j} pk_{H,j}, ck_A)$
7. [consistency of  $pk_C, pk'_C, pk_K$ ] For  $i = 0, \dots, m+3$ :  $\mathbf{e}(pk_{C,i}, P_2) = \mathbf{e}(\sum_{j=0}^{d-1} c_{i,j} pk_{H,j}, ck_C)$   
 $\mathbf{e}(pk'_{C,i}, P_2) = \mathbf{e}(pk_{C,i}, vk_C)$   
and for  $i = 0, \dots, n$ :  $\mathbf{e}(pk_{K,i}, vk_\gamma) = \mathbf{e}(vk_{IC,i} + pk_{C,i}, \widehat{vk}_{\beta\gamma}) \cdot \mathbf{e}(vk_{\beta\gamma}, pk_{B,i})$   
and for  $i = n+1, \dots, m+3$ :  $\mathbf{e}(pk_{K,i}, vk_\gamma) = \mathbf{e}(pk_{A,i} + pk_{C,i}, \widehat{vk}_{\beta\gamma}) \cdot \mathbf{e}(vk_{\beta\gamma}, pk_{B,i})$
8. If all checks in 1.–7. succeeded then return true and otherwise false.

Figure 3: CRS-consistency checks for [BCTV14] from [Fuc18].

1. performing some “minimal checks” on the CRS that were described in [GGPR13] for applications where WI is sufficient;
2. using a subversion-ZK scheme [BFS16];
3. generating the CRS in a two-party secure computation between the prover and the verifier;
4. using their concept of contingent payments for services (see Section 5), which they claim to only require a subversion-WI proof system:

“The best solution in [the authors’] opinion is to use the protocol for ZK Contingent Service Payments [...] In contrast to the protocol of [Fuc18] that would take an hour to run, this protocol adds less than a minute to the prover’s runtime.” [CGGN17]

**COUNTERMEASURE 1.** In Section 4.2 we show that the proposed *minimal checks* on the CRS do not address the issue, in that they do not prevent all attacks. We show that most of the CRS elements need to be checked for consistency in order to exclude attacks *even against WI*.

**COUNTERMEASURE 2.** Subversion-ZK fixes the problem, since it assures the seller that the proof she produces, using a CRS which could have been generated maliciously, will not reveal any information about its witness.

One could thus simply replace the SNARK scheme in zkCP by the subversion-ZK version of Pinocchio from [Fuc18] given in Figure 2; but this entails the expensive checks of consistency of

every CRS element given in Figure 3. CGGN write: “*Note that this check requires the computation of  $m$  bilinear maps, a much more expensive task than the simple checks required for WI. [...] Our experimental results suggests that running the subversion-resistant checks of [Fuc18] for the pay-to-sudoku example would take more than an hour on our benchmark machine.*” [CGGN17]

They also write: “*Indeed, subversion-ZK can be obtained as long as the above ‘WI checks’ are performed and the value  $\tau$  can be extracted by the simulator from the Verifier when it produces a CRS.*” However, as per our attack (Section 4.2), the minimal checks are not enough to ensure subversion WI, thus a fortiori, they do not suffice to obtain subversion ZK.

**COUNTERMEASURE 3.** Using a secure computation protocol will still require techniques to ensure that every element of the CRS was generated correctly and will therefore not alleviate the burden of checking consistency of the entire CRS, as done by the prover in the subversion-ZK variant (although it could avoid the computation of numerous pairing evaluations, see Footnote 3).

**COUNTERMEASURE 4.** Whereas WI proofs cannot be used in situations where solution are unique, as for Sudoku, the authors suggest to use their protocol for *zero-knowledge contingent service payments* (see Section 5), as it only requires subversion WI. In Section 5.2 we show that even for zkCSP, subversion zero knowledge is required: their protocol could be insecure if the used proof system only satisfies subversion WI.

## 4.2 The “Minimal Checks” are not Enough

Campanelli et al. [CGGN17, Section 3.3] write: “*the prover can check that the CRS is ‘correctly formed’ and in this case the protocol is witness indistinguishable (WI). In the QAP-based SNARK described in [Figure 2], it is sufficient that the prover/seller checks that*

- *The polynomials  $A, B, C, Z$  are well formed with respect to the circuit  $C$ .*
- *The elements  $pk_{A,m+1}, pk'_{A,m+1}, pk'_{B,m+2}, pk_{C,m+3}, pk'_{C,m+3}$  are not equal to  $0 \in \mathbb{G}_1$  and the element  $pk_{B,m+2}$  is not equal to  $0 \in \mathbb{G}_2$*
- *All the elements  $pk_{H,i}$  are not  $0 \in \mathbb{G}_1$ .*
- *The element  $vk_Z$  is such that  $vk_Z \neq 0 \in \mathbb{G}_2$ .*

*since this will guarantee that the proof is a uniformly distributed random value no matter what witness is used.*”

We show that these checks are not sufficient to guarantee subversion witness indistinguishability. While under a CRS that passes the above checks, proofs that *verify* are distributed uniformly among all valid proofs, this does not suffice. Consider an application for which witness indistinguishability is a sufficient guarantee for the prover. As in the attack on the zkCP implementation [CGGN17], we show how a malicious verifier can learn the value of wire  $s_j$ . The verifier sets up the CRS as follows:

- Compute the CRS as in Figure 2, except that  $pk'_{B,j}$  is *not* set to  $B_j(\tau)\alpha_B\rho_B P_1$ , but to any other value; that is for an arbitrary  $\xi \not\equiv B_j(\tau) \pmod{p}$ , define

$$pk'_{B,j} := \xi\alpha_B\rho_B P_1 .$$

Consider a proof  $\pi = (\pi_A, \pi'_A, \pi_B, \pi'_B, \pi_C, \pi'_C, \pi_K, \pi_H)$  created under this CRS for  $\vec{s} \in \mathbb{F}^m$ . Inspection of the scheme in Figure 2 yields:

$$\begin{aligned}\pi_B &= \langle \vec{c}, pk_B \rangle = \underbrace{\left( B_0(\tau) + \sum_{i=1}^m s_i B_i(\tau) + \delta_B Z(\tau) \right)}_{:=\varphi} \rho_B P_2 \\ \pi'_B &= \langle \vec{c}, pk'_B \rangle = \underbrace{\left( B_0(\tau) + \sum_{\substack{i=1 \\ i \neq j}}^m s_i B_i(\tau) + s_j \xi + \delta_B Z(\tau) \right)}_{:=\varphi'} \alpha_B \rho_B P_1\end{aligned}$$

If  $s_j = 1$  then, by the choice of  $\xi$ , we have  $\varphi \not\equiv \varphi' \pmod{p}$  and thus the second verification equation,

$$\mathbf{e}(\pi'_B, P_2) = \mathbf{e}(vk_B, \pi_B), \quad (3)$$

does not hold since

$$\mathbf{e}(\pi'_B, P_2) = \mathbf{e}(\varphi' \alpha_B \rho_B P_1, P_2) \neq \mathbf{e}(\alpha_B P_1, \varphi \rho_B P_2) = \mathbf{e}(vk_B, \pi_B).$$

The proof  $\pi$  produced by the prover does thus not verify when  $s_j = 1$ .

On the other hand, if  $s_j = 0$  then  $\varphi = \varphi'$  and Eq. (3) does hold. Since  $pk'_B$  only affects the computation of  $\pi'_B$  and Eq. (3) is the only equation in which  $\pi'_B$  occurs, the resulting proof is valid.

The verifier can thus determine the value  $s_j$  by simply checking if the proof verifies. This kind of attack was not considered in the argument from [CGGN17] given above, which only applies to valid proofs. An apparent countermeasure could be to have the prover check its own proof before sending it (and in fact, libsnark [BCG<sup>+</sup>14b] implements this security measure), but this does not help either: if the verifier receives a valid proof, she knows that  $s_j = 0$ , whereas if she does not receive a proof, she can deduce that  $s_j = 1$ .

The same attack works by changing one of the values contained in  $pk'_A$  or  $pk'_C$ ; moreover, an analogous attack works by changing any of the values  $pk_{A,j}$ ,  $pk_{B,j}$ ,  $pk_{C,j}$  or  $pk_{K,i}$  for some  $j \in \{n+1, \dots, m\}$ . If the corresponding value  $s_j = 0$ , then, again, the change in  $pk$  has no influence on the corresponding value  $\pi_A$ ,  $\pi_B$ ,  $\pi_C$  or  $\pi_K$ , respectively, and the proof remains valid, whereas it will be invalid if  $s_j = 1$ .

These attacks show that the scheme from [BCTV14], with provers only performing the *minimal checks* proposed by [CGGN17], does not satisfy S-WI as formalized in Definition 2.3, no matter whether the prover simply outputs the (possibly wrong) proof, or whether it returns  $\perp$  if the proof does not verify. The adversary in game S-WI simply changes one of the key elements with index  $j$  and queries its PROVE oracle for two witnesses that differ in position  $j$ .

This shows that the majority of elements of  $pk$ , the part of the CRS used by the prover, must be checked for consistency. Only then can the prover be sure that *every* possible witness will lead to a valid proof, which will be distributed independently of the witness, which will in turn ensures subversion-resistant WI. In particular, the prover key  $pk$  consists of  $7m - 2n + d + 19$  group elements and changing any of  $7(m - n)$  values enables the attack.

### 4.3 Other zkSNARK Schemes

THE ORIGINAL GGPR SNARK. Our subversion-WI attacks against the Pinocchio instantiation from [BCTV14] with the additional minimal checks from [GGPR13, CGGN17] equally apply to the original zkSNARK [GGPR13] (of which Pinocchio is an optimized version).

THE GROTH16 SNARK. The most efficient zkSNARK scheme to date is due to Groth [Gro16] and it replaces asymmetric Pinocchio in the next iteration of Zcash.

Proofs in this scheme consist of merely 3 bilinear-group elements and are verified by computing only 3 pairings. There are no CRS elements  $pk_{A,i}, pk_{B,i}, pk_{C,i}$  as in Figure 2; instead the prover computes  $\pi_A$  and  $\pi_B$  by evaluating polynomials  $A_i$  and  $B_i$  “in the exponent” by itself, that is,  $\pi_A$  contains  $\sum_{j=0}^{d-1} (a_{0,j} + \sum_{i=1}^m s_i a_{i,j}) pk_{H,j}$  (where  $pk_{H,i}$  are defined as in Figure 2 and  $a_{i,0}, \dots, a_{i,d-1}$  are the coefficients of polynomial  $A_i$ ). Our initial attack from Section 4.2 does thus not carry over immediately.

The bulk of CRS elements are of the form

$$pk_{K,i} := \delta^{-1}(\beta A_i(\tau) + \alpha B_i(\tau) + C_i(\tau)) P_1,$$

(where  $\alpha, \beta$  and  $\delta$  are part of the CRS randomness) and the third proof element  $\pi_C$  contains the term  $\sum_{i=n+1}^m s_i pk_{K,i}$ . It is thus easy to see that by modifying  $pk_{K,j}$  for some  $j$ , proofs under such a CRS will be invalid if and only if  $s_j = 1$ .

Groth’s scheme [Gro16] was shown to be subversion-ZK [Fuc18] if all CRS elements are verified using the bilinear map. The above attack thus shows that there is no way around this expensive check, as already a single inconsistent element  $pk_{K,j}$  suffices to break even subversion WI.

## 5 ZK Contingent Service Payments

### 5.1 Definition and Construction from CGGN

INTUITION. Zero-knowledge contingent *service* payments (zkCSP) allow a service provider to convince the buyer that a service has been provided; and whether the buyer can really be convinced is contingent on him making the required payment. The underlying assumption is that there exists a predicate  $V$  so that provision of the service can be proved by exhibiting a value  $s$  that satisfies  $V$ . Campanelli et al. [CGGN17] introduce zkCSP after observing that the zkCP protocol cannot be used, as the buyer is not interested in actually acquiring the proof  $s$ , but only in the fact that the provider knows it.

Recall that in the ZK contingent payment protocol the seller of information  $s$ , which satisfies a predicate  $V$ , does the following: she chooses a random key  $r$ , encrypts  $s$  under  $r$ , which yields a ciphertext  $c$ , computes  $y = \text{SHA}(r)$  and sends the buyer the pair  $(y, c)$  together with a zkSNARK proof of the following statement:

*I know a value  $r$  such that the decryption  $s$  of  $c$  under key  $r$  satisfies  $V$ , and  $y = \text{SHA}(r)$ .*

In a ZK contingent *service* payment the service provider also selects a random value  $r$ , sends only one value  $y$ , and proves the following:

*I know two values  $s$  and  $r$  such that **either**  $s$  satisfies  $V$  and  $y$  is the SHA image of  $r$ ; **or**  $y$  is the image of  $r$  under a different hash function  $H$ .*

Now there are two possibilities: either the seller used values  $s, r$  so that  $\text{SHA}(r) = y$  and  $V(s)$ ; then she actually knows a “proof”  $s$  and can later use  $r$  to redeem the hash-locked transaction to  $y$  that the buyer will make. However, if  $V(s)$  does *not* hold then, by the soundness of the proof, the seller must have used a value  $r$  with  $H(r) = y$ . In this case, the seller *cannot* redeem the payment, assuming that  $\text{SHA}$  and  $H$  satisfy *claw-freeness* (Section 2.5), meaning it is hard to find values  $r_0, r_1$  with  $\text{SHA}(r_0) = H(r_1)$ . Indeed, if the seller does not know a satisfying value  $s$ , she must use a witness  $(s', r)$  with  $H(r) = y$  to prove the statement; if she later somehow redeems the payment by presenting  $r'$  with  $y = \text{SHA}(r')$ , then she has found a claw  $(r', r)$  for  $\text{SHA}$  and  $H$ .

DEFINITION. We give the definition from [CGGN17], adapted to the Bitcoin setting and, as before, denote by  $\text{SHA}$  the SHA256 hash function used by Bitcoin. (Note that here  $f$  plays the role of the predicate  $V$ .)

In a zkCSP, a server  $A$  proves to a client  $B$  knowledge of  $s$  such that  $f(s) = 1$  for an efficiently computable function  $f: \{0, 1\}^* \rightarrow \{0, 1\}$  and wants to be paid for this information. The protocol is required to have the following properties:

- If (a possibly malicious)  $A$  is paid then  $A$  really knows a value  $s$  satisfying  $f(s) = 1$ .
- If (a possibly malicious)  $B$  does not pay then  $B$  learns no information.
- Even when (a possibly malicious)  $B$  pays, he learns nothing except the fact  $A$  that knows a satisfying  $s$ .

In addition to  $A$  and  $B$ , the protocol involves a trusted party  $T$  representing the blockchain, which maintains in a ledger the coin balance of every party.  $T$  accepts two types of messages and executes the instructions honestly:

**Contingent payments** from party  $B$  of the form: *Transfer  $m$  bitcoins to a party that presents a SHA preimage of  $y$ .* If the party that issued the instruction has at least  $m$  coins then  $T$  publishes the message on the blockchain.

**Redemption payments** from  $A$  of the form: *Here is a SHA preimage  $x$  of  $y$ ; transfer  $m$  bitcoins to my account.* If there has been a corresponding contingent-payment message for  $m$  and  $y$ , and  $\text{SHA}(x) = y$ , then  $T$  posts the message on the blockchain, reduces  $B$ 's balance by  $m$  and credits  $A$ 's balance by  $m$ .

A zkCSP protocol is a 3-party protocol between  $A$ , who has private input  $s$ , and parties  $B$  and  $T$ , all of which have public input  $f$ . The view of (possibly malicious)  $\hat{B}$  is defined as its randomness and all exchanged messages:

$$\text{View}_{\hat{B}}(s, f) := \text{Coins}_{\hat{B}} \parallel \text{Messages}[A(s, f), \hat{B}(f), T(f)] \parallel \text{Out}[A(s, f), \hat{B}(f), T(f)] .$$

A zkCSP protocol must satisfy the following two security notions [CGGN17]:

**Extraction** For any possibly malicious efficient  $\hat{A}$ , if at the end of the protocol  $\hat{A}$ 's balance increases with non-negligible probability, then there exists an efficient extractor  $\text{Ext}_{\hat{A}}$ , which outputs a string  $\hat{s}$  such that  $f(\hat{s}) = 1$ ;

**Zero-Knowledge** For any possibly malicious efficient  $\hat{B}$ , there exists an efficient simulator  $\text{Sim}_{\hat{B}}$  which on input  $f$  outputs a distribution which is computationally indistinguishable from  $\text{View}_{\hat{B}}(s, f)$ .

INSTANTIATION. The instantiation [CGGN17] makes use of a hash function  $\text{H}: \{0, 1\}^* \rightarrow \{0, 1\}^{256}$ , which is claw-free with  $\text{SHA}$ , and whose outputs on a random value from  $\{0, 1\}^{256}$  are indistinguishable from those of  $\text{SHA}$ . It moreover uses a witness-indistinguishable proof-of-knowledge system  $\Pi = (\Pi.\text{Pg}, \Pi.\text{P}, \Pi.\text{V})$  for the following NP-relation:

$$R_{f, \text{H}}(y, (s, r)) \Leftrightarrow (f(s) = 1 \wedge y = \text{SHA}(r)) \vee (f(s) = 0 \wedge y = \text{H}(r)) , \quad (4)$$

In other words,  $\Pi$  lets users prove knowledge of a preimage of some  $y$  under the following function:

$$\mathcal{F}_{f, \text{H}}(s, r) = \begin{cases} \text{SHA}(r) & \text{if } f(s) = 1 \\ \text{H}(r) & \text{otherwise} \end{cases} \quad (5)$$

The protocol is given as Protocol 1 on the next page, and CGGN state the following theorem:

Protocol 1

1.  $A$ , on input  $s$ , chooses  $r$  randomly from  $\{0, 1\}^{256}$  and computes  $y = \mathcal{F}_{f,H}(s, r)$ .
2.  $A$  sends  $y$  to  $B$  and makes a WI proof of knowledge of  $(s, r)$  with  $y = \mathcal{F}_{f,H}(s, r)$ . If the proof fails, the buyer stops.
3.  $B$  makes a *contingent payment* of  $m$  bitcoins for the hash value  $y$ .
4.  $A$  sends a *redemption payment* message for value  $z$ . If  $\text{SHA}(z) = y$  then  $T$  posts the message and transfers  $m$  bitcoins from  $B$  to  $A$ .

\* THEOREM 4.1 [CGGN17]. *Assume that SHA and H are claw-free and the distributions  $\text{SHA}(r)$  and  $H(r)$  for  $r$  chosen at random from  $\{0, 1\}^{256}$  are computationally indistinguishable [and the proof system is a WI proof of knowledge] then Protocol 1 is a secure zkCSP protocol.*

**Extraction** of the protocol is shown by a reduction to claw-freeness of SHA and H. The extractor Ext is defined as follows: suppose  $A$  sends  $y$  and makes a convincing proof of knowledge; then using the extractor for the proof system, Ext extracts a witness  $(\hat{s}, \hat{r})$  and returns  $\hat{s}$ .

Assume that  $A$  is paid but extraction fails, that is,  $f(\hat{s}) \neq 1$ . By extractability of the proof system,  $(\hat{s}, \hat{r})$  is a witness for  $y$  under  $R_{f,H}$ , which implies  $y = H(\hat{r})$ . On the other hand,  $A$  is only paid if it presented a value  $z$  with  $\text{SHA}(z) = y$ . The pair  $(z, \hat{r})$  is thus a claw for SHA and H.

**Zero knowledge** is argued by constructing a simulator  $\text{Sim}_{\hat{B}}$  for a possibly malicious  $\hat{B}$  as follows:

*For step 1,  $\text{Sim}_{\hat{B}}$  will choose  $r, s$  at random and compute  $y = \mathcal{F}_{f,H}(s, r)$ . Note that the message in step 1 is computationally indistinguishable from the message sent by the real  $A$  due to the computational indistinguishability of the output distributions of SHA and H. For step 2,  $\text{Sim}_{\hat{B}}$  will just run a “real” proof that  $y = \mathcal{F}_{f,H}(s, r)$ : note that due to witness indistinguishability, this proof is indistinguishable from a proof of a “correct” proof when the witness is such that  $f(s) = 1$ .*

While  $\text{Sim}_{\hat{B}}$ , as described above, simulates steps 1 and 2 of Protocol 1, it is not clear how step 4 is simulated, which is part of  $\text{View}_{\hat{B}}$ . In particular, if the simulator chooses values  $s, r$  with  $f(s) = 0$  (which will be typically the case) then  $y = H(r)$ . However, if the value  $s'$  that party  $A$  holds satisfies  $f(s') = 1$  then, after sending  $y'$  in step 2,  $A$  would send  $z$  with  $\text{SHA}(z) = y'$  in step 4 (so  $\text{Sim}_{\hat{B}}$  would have to break claw-freeness to simulate this).

In fact, Protocol 1 can be proved to satisfy *zero knowledge*, when using a non-interactive proof system  $\Pi$  for which, as for zkCP, the buyer  $B$  generates the CRS—provided that  $\Pi$  satisfies *subversion zero knowledge* (Definition 2.5). This notion guarantees that for any  $\hat{B}$  that produces a CRS, there exists a simulator  $S = (S.\text{crs}, S.\text{pf})$ , where  $S.\text{crs}$  simulates a CRS and coins  $\rho$  for  $\hat{B}$  (as required for  $\text{View}_{\hat{B}}$ ) and outputs a trapdoor  $td$ .

Using this simulator  $S$ , the simulator  $\text{Sim}_{\hat{B}}$  for Protocol 1 first chooses  $r \leftarrow_{\$} \{0, 1\}^{256}$  and computes  $y := \text{SHA}(r)$ . It runs  $(\text{crs}, \rho, td) \leftarrow_{\$} S.\text{crs}(R_{f,H})$  and  $\pi \leftarrow_{\$} S.\text{pf}(R_{f,H}, \text{crs}, td, y)$  and sends  $(y, \pi)$  to  $\hat{B}$ . If  $\hat{B}$  makes a contingent payment for hash value  $y$  and  $A$  makes a redemption payment step 4, then  $\text{Sim}_{\hat{B}}$  can simulate this using value  $r$ .  $\text{View}_{\hat{B}}$  now consists of the coins  $\rho$ , as well as the messages and outputs of all parties.

We now show that subversion witness indistinguishability is actually not sufficient to guarantee security of the protocol, by defining a subversion-WI proof system which lets party  $B$  learn whether  $A$  holds a satisfying  $s$ , without paying.

## 5.2 WI is not Enough

In Section 4.2 we showed that for the zkSNARK systems from the literature [GGPR13, PHGR13, BCTV14, Gro16], there appears to be little computational gain in only aiming for subversion WI (sWI) as opposed to subversion ZK (sZK). There is however a huge difference in terms of the cryptographic hardness assumptions on which they rely: while sWI is *unconditional*, sZK requires a strong non-falsifiable *knowledge-type* assumption [Fuc18]. Moreover, since (s)WI is a weaker notion than (s)ZK, even from an efficiency perspective, it is preferable to construct protocols that only require (s)WI, as it may be achievable by more efficient protocols.

We now show that even if Protocol 1 can rely on a CRS generated by a trusted party, that is, when subversion resistance is not required, security is not guaranteed when the used proof system is only WI, as assumed in [CGGN17]. Consider a WI proof-of-knowledge (PoK) system  $\Pi = (\Pi.\text{Pg}, \Pi.\text{P}, \Pi.\text{V})$  for  $R_{f,\mathbb{H}}$  from Eq. (4) and consider the following variant  $\Pi'$  of  $\Pi$ :

- $\Pi'.\text{Pg}$  is defined as  $\Pi.\text{Pg}$ .
- $\Pi'.\text{P}(crs, y, (s, r))$ : run  $\pi \leftarrow \Pi.\text{P}(crs, y, (s, r))$ ;  
if  $f(s) = 1$ , let  $\beta := 1$ , else let  $\beta := 0$ ;  
return  $\pi' := (\pi, \beta)$ .
- $\Pi'.\text{V}(crs, y, (\pi, \beta))$ : return  $\Pi.\text{V}(crs, y, \pi)$ .

It is immediate that proof system  $\Pi'$  for  $R_{f,\mathbb{H}}$  again satisfies completeness and knowledge soundness (Definition 2.1), as this notion is independent of  $\Pi'.\text{P}$ . Moreover, if  $\text{SHA}$  and  $\mathbb{H}$  are claw-free then  $\Pi'$  is witness-indistinguishable (the same argument shows that subversion WI of  $\Pi'$  is inherited from subversion WI of  $\Pi$ ):

In game  $\text{WI}_{\Pi', R_{f,\mathbb{H}}, \mathbb{A}}$  (cf. Figure 1) the adversary must decide a bit  $b$ . It is given access to a  $\text{PROVE}$  oracle, which, given a statement  $y$  and two valid witnesses  $w_0 = (r_0, s_0)$  and  $w_1 = (r_1, s_1)$  for it, returns a proof  $(\pi_b, \beta_b)$  for  $y$  computed with  $w_b$ .

If  $\beta_0 = \beta_1$  then indistinguishability follows from indistinguishability of  $\pi_0$  and  $\pi_1$ , that is, from WI of  $\Pi$ . On the other hand,  $\beta_0$  and  $\beta_1$  are different (say  $\beta_0 = 0$  and  $\beta_1 = 1$ ) only if  $f(s_0) = 0$  and  $f(s_1) = 1$ . But together with validity of  $w_0$  and  $w_1$  for  $y$ , this implies  $y = \mathbb{H}(r_0)$  and  $y = \text{SHA}(r_1)$ , which means that the adversary must find a claw  $(r_1, r_0)$  for the pair  $(\text{SHA}, \mathbb{H})$  in order to make such a query.

As proofs computed by  $\Pi'.\text{P}$  obviously leak whether the used witness  $(s, r)$  satisfies  $f(s) = 1$ , this yields the following theorem (whose formal proof is analogous to the proof of Theorem 5.2).

**Theorem 5.1** *If  $\text{SHA}$  and  $\mathbb{H}$  are claw-free and there exists a WI proof-of-knowledge (PoK) scheme  $\Pi$  for relation  $R_{f,\mathbb{H}}$ , then there exists a WI PoK scheme  $\Pi'$  so that Protocol 1, when instantiated with  $\Pi'$ , reveals whether the seller proved knowledge of a valid value  $s$  or not.*

A slight variation of the above argument yields the following theorem, which we formally prove.

**Theorem 5.2** *If  $\text{SHA}$  and  $\mathbb{H}$  are claw-free, if there exists a unique value  $s$  with  $f(s) = 1$  and if there exists a subversion-WI PoK scheme  $\Pi$  for relation  $R_{f,\mathbb{H}}$ , then there exists a subversion-WI PoK scheme  $\Pi''$  so that Protocol 1, when instantiated with  $\Pi''$ , reveals the value  $s$  if the seller used it in the protocol.*

**Proof.** Let  $\Pi$  be a subversion-WI PoK system for the language defined by relation  $R_{f,\mathbb{H}}$  from Eq. (4). Then the proofs of the following modified system  $\Pi''$  reveal the value  $s$  whenever  $f(s) = 1$ :

<p style="margin: 0;"><u>GAME S-WI<math>_{\Pi'', A}</math></u> <span style="border: 1px solid black; padding: 2px;">GAME<sub>1</sub></span></p> <p style="margin: 0;"><i>// Ignore the boxes for game S-WI</i></p> <p style="margin: 0;"><math>b \leftarrow_s \{0, 1\}</math>; <math>(crs, st) \leftarrow_s A</math></p> <p style="margin: 0;"><math>b' \leftarrow_s A^{\text{PROVE}}(crs, st)</math></p> <p style="margin: 0;">Return <math>(b = b')</math></p>	<p style="margin: 0;"><u>PROVE<math>(y, (s_0, r_0), (s_1, r_1))</math></u></p> <p style="margin: 0;"><i>// If <math>R(y, (s_0, r_0))</math> is false, return <math>\perp</math>:</i></p> <p style="margin: 0;">    If <math>f(s_0) = 1 \wedge y \neq \text{SHA}(r_0)</math> then return <math>\perp</math></p> <p style="margin: 0;">    If <math>f(s_0) = 0 \wedge y \neq H(r_0)</math> then return <math>\perp</math></p> <p style="margin: 0;"><i>// If <math>R(y, (s_1, r_1))</math> is false, return <math>\perp</math>:</i></p> <p style="margin: 0;">    If <math>f(s_1) = 1 \wedge y \neq \text{SHA}(r_1)</math> then return <math>\perp</math></p> <p style="margin: 0;">    If <math>f(s_1) = 0 \wedge y \neq H(r_1)</math> then return <math>\perp</math></p> <p style="margin: 0;">    <span style="border: 1px solid black; padding: 2px;">If <math>f(s_0) \neq f(s_1)</math> then return <math>\perp</math> (I)</span></p> <p style="margin: 0;"><i>// Compute <math>\pi \leftarrow_s \Pi''.P(crs, y, (s_b, r_b))</math></i></p> <p style="margin: 0;">    <math>\pi \leftarrow_s \Pi.P(crs, y, (s_b, r_b))</math></p> <p style="margin: 0;">    If <math>f(s_b) = 1</math> then let <math>\pi'' := s_b</math>; else <math>\pi'' := 0</math></p> <p style="margin: 0;">Return <math>(\pi, \pi'')</math></p>
--	---

Figure 4: Subversion-WI game for adversary  $A$  against scheme  $\Pi''$  for relation  $R_{f,H}(y, (s, r))$  in the proof of Lemma 5.3.

- $\Pi''.Pg$  is defined as  $\Pi.Pg$ .
- $\Pi''.P(crs, y, (s, r))$ : run  $\pi \leftarrow \Pi.P(crs, y, (r, s))$ ;  
if  $f(s) = 1$ , return  $\pi'' := (\pi, s)$ ;  
else return  $\pi'' := (\pi, 0)$ .
- $\Pi''.V(crs, y, (\pi, \hat{s}))$ : return  $\Pi.V(crs, y, \pi)$ .

As for  $\Pi'$  above, completeness and knowledge soundness are both directly inherited from  $\Pi$ . We prove that  $\Pi''$  is sWI in the lemma below, which concludes the proof. ■

**Lemma 5.3** *Let  $\Pi$  be a subversion-WI PoK system for  $R_{f,H}$ . If  $\text{SHA}$  and  $H$  are claw-free and if there exists only one value  $s$  with  $f(s) = 1$  then the system  $\Pi''$  defined in the proof of Theorem 5.2 is sWI. In particular, for every adversary  $A$  against sWI of  $\Pi''$ , there exists an sWI adversary  $B$  against  $\Pi$  and an adversary  $C$  against the claw-freeness of  $\text{SHA}$  and  $H$ , such that*

$$\mathbf{Adv}_{\Pi'', A}^{\text{s-wi}} \leq \mathbf{Adv}_{\Pi, B}^{\text{s-wi}} + 2 \cdot \mathbf{Adv}_{(\text{SHA}, H), C}^{\text{cf}} .$$

**Proof.** We start with writing out the game S-WI $_{\Pi'', A}$  in Figure 4, where we also define GAME<sub>1</sub>, a variant of the game, which includes the box. Let  $E$  denote the event that in some call to the PROVE oracle, GAME<sub>1</sub> returns  $\perp$  in the boxed line (I). Since games  $G_0 := \text{S-WI}_{\Pi'', A}$  and  $G_1 := \text{GAME}_1$  behave identically if  $E$  does not occur, we have:

$$\begin{aligned} \Pr[G_0] - \Pr[G_1] &= (\Pr[G_0|E] - \Pr[G_1|E]) \Pr[E] + (\Pr[G_0|\neg E] - \Pr[G_1|\neg E]) \Pr[\neg E] \\ &\leq \Pr[E] . \end{aligned} \tag{6}$$

Consider an adversary  $C$  that simulates GAME<sub>1</sub> for  $A$  and if at some point  $A$  makes an oracle call  $\text{PROVE}(y, (s_0, r_0), (s_1, r_1))$  that returns  $\perp$  in line (I), then  $C$  stops the simulation. If  $f(s_0) = 0$  and  $f(s_1) = 1$  then  $C$  returns  $(r_1, r_0)$ , else it returns  $(r_0, r_1)$ .

We show that whenever  $E$  occurs then (and only then)  $C$  breaks claw-freeness of  $\text{SHA}$  and  $H$ : Suppose, w.l.o.g.  $f(s_0) = 0$  and  $f(s_1) = 1$ . Since the simulation of the oracle proceeded until

zkCP-from-zkCSP

1.  $A$ , on input  $s$ , selects a key  $r$ , encrypts  $c := \text{Enc}_r(s)$  and computes  $y = \text{SHA}(r)$ .
2.  $A$  sends  $(y, c)$  to  $B$  together with a WI proof of knowledge  $\pi$  that

$$y = \mathcal{G}_{V,c,H}(r) := \begin{cases} \text{SHA}(r) & \text{if } V(\text{Dec}_r(c)) = 1 \\ H(r) & \text{otherwise} \end{cases}$$

3.  $B$  makes a *contingent payment* of  $m$  bitcoins for the hash value  $y$ .
4.  $A$  sends a *redemption payment* message for value  $z$ . If  $\text{SHA}(z) = y$  then  $T$  posts the message and transfers  $m$  bitcoins from  $B$  to  $A$ .
5.  $B$  computes  $s' = \text{Dec}_z(c)$ .

line (I), we have  $y = H(r_0)$  and  $y = \text{SHA}(r_1)$  (otherwise it would have returned  $\perp$  before). Adversary  $C$ 's output  $(r_1, r_0)$  is thus a claw for  $\text{SHA}$  and  $H$  and we therefore have

$$\mathbf{Adv}_{(\text{SHA},H),C}^{\text{cf}} = \Pr[\text{CF}_{(\text{SHA},H),C}] = \Pr[E] . \quad (7)$$

Finally, consider the following adversary  $B$  against sWI of proof system  $\Pi$ .  $B$  forwards the CRS from  $A$  to its challenger and simulates  $A$ 's PROVE queries  $(y, (s_0, r_0), (s_1, r_1))$  by forwarding them to its own prove oracle. If it obtains  $\perp$  or if  $f(s_0) \neq f(s_1)$ , it returns  $\perp$  to  $A$ ; otherwise, if it obtained  $\pi$ , it does the following: if  $f(s_0) = 1$ , it returns  $(\pi, s_0)$ , else it returns  $(\pi, 0)$ .

This perfectly simulates  $\text{GAME}_1$ : Case  $b = 0$  it immediate; if  $b = 1$  and if  $B$  did not return  $\perp$ , we have  $f(s_0) = f(s_1)$ . If  $f(s_0) = 1$  then by uniqueness,  $s_0 = s_1$  and  $B$  returned  $(\pi, s_b)$  as required. If  $f(s_0) = 0$  then  $(\pi, 0)$  is the correct proof for both  $b = 0$  and  $b = 1$ . We thus have

$$\Pr[\text{S-WI}_{\Pi,B}] = \Pr[G_1] . \quad (8)$$

Together, we have

$$\mathbf{Adv}_{\Pi',A}^{\text{s-wi}} = 2 \Pr[G_0] - 1 \stackrel{(6)}{\leq} 2(\Pr[G_1] + \Pr[E]) - 1 \stackrel{(7),(8)}{=} \mathbf{Adv}_{\Pi,B}^{\text{s-wi}} + 2 \cdot \mathbf{Adv}_{(\text{SHA},H),C}^{\text{cf}} ,$$

which proves the lemma.  $\blacksquare$

### 5.3 ZK Contingent Payments from zkCSP

We conclude by showing that the issues found with zkCSP carry over to the authors' preferred fix of the problem they discovered with ZK contingent payments, namely to follow their zkCSP approach. Recall that in zkCP the seller  $A$  wants to sell a secret (or solution)  $s$  to the buyer  $B$ , and there is some predicate  $V$  that decides validity of values  $s$ .

The authors [CGGN17, Section 4.2] (we slightly changed notation for consistency with the above) propose a protocol that uses a symmetric-key encryption scheme  $(\text{Enc}, \text{Dec})$  and subversion-WI PoK system for the following relation:

$$R'_{V,H}((y, c), r) \Leftrightarrow (V(\text{Dec}_r(c)) = 1 \wedge y = \text{SHA}(r)) \vee (V(\text{Dec}_r(c)) = 0 \wedge y = H(r)) . \quad (9)$$

The protocol is specified as zkCP-from-zkCSP above. Note that  $V$  in Eq. (9) corresponds to  $f$  in Eq. (4); the main difference to zkCSP is that instead of  $\pi$  proving knowledge of  $s$ , this is done

explicitly via  $c$ , which makes  $s$  extractable by the buyer once  $r$  is revealed. Note also that by collision-resistance of SHA, in this case we must have  $z = r$  and thus  $s' = s$  with overwhelming probability.

We show that when solutions are unique, as is the case for Sudoku, using a WI proof system (rather than a ZK-secure one) makes the protocol insecure, as the buyer might learn  $s$  entirely before making any payment.

**Theorem 5.4** *If SHA and H are claw-free, if there is only one value  $s$  with  $V(s) = 1$  and if there exists a subversion-WI PoK scheme  $\Pi$  for relation  $R'_{V,H}$ , then there exists a subversion-WI PoK scheme  $\Pi'$ , so that zkCP-from-zkCSP, when instantiated with  $\Pi'$ , reveals the value  $s$  if the seller used it in the protocol.*

The theorem is shown analogously to Theorem 5.2; we give a proof sketch: Again, assume a subversion-WI PoK system  $\Pi$  for relation  $R'_{V,H}$ . Then the following proof system fully reveals the value  $s$  if  $f(s) = 1$ :

- $\Pi'.Pg$  is defined as  $\Pi.Pg$ .
- $\Pi'.P(crs, (y, c), r)$ : run  $\pi \leftarrow \Pi.P(crs, (y, c), r)$ ; compute  $s := \text{Dec}_r(c)$ ;  
if  $V(s) = 1$ , return  $\pi' := (\pi, s)$ ; else return  $\pi' := (\pi, 0)$ .
- $\Pi'.V(crs, y, (\pi, \hat{s}))$ : return  $\Pi.V(crs, y, \pi)$ .

Completeness and knowledge soundness of  $\Pi'$  are again immediate. Subversion WI is proved as in Lemma 5.3; we give a sketch:

Suppose in game S-WI, the adversary asks for a proof of statement  $(y, c)$  under one of two (different) valid witnesses  $r_0$  or  $r_1$ . Define  $s_b = \text{Dec}_{r_b}(c)$ . We distinguish 3 different cases:

- $V(s_0) = 1 = V(s_1)$ . By uniqueness of  $V$ , we have  $s_0 = s = s_1$  and the adversary receives a proof  $(\pi, s)$ , where  $\pi$  is WI.
- $V(s_0) = 0 \neq 1 = V(s_1)$  (or the other way round): validity of the witnesses w.r.t.  $R'_{V,H}$  implies  $y = H(r_0)$  and  $y = \text{SHA}(r_1)$ , which means that  $(r_1, r_0)$  is a claw for SHA and H.
- $V(s_0) = 0 = V(s_1)$ : Then for both values of  $b$ , the adversary receives a WI proof  $(\pi, 0)$ .

Thus to break sWI of the modified scheme  $\Pi'$ , either the adversary must break claw-freeness, or it breaks sWI of the original scheme  $\Pi$ .

CONCLUSION. We have showed that in the fixed zkCP protocol from [CGGN17], the *minimal checks* do not prevent an attack that lets the buyer learn one bit of information about the digital good. The reason is that the proposed variant of the SNARK proof is not subversion-witness-indistinguishable (sWI). Moreover, even if the used proof system was sWI, this would not prevent the buyer from even learning the good completely before making any payment.

To conclude, we gave evidence that expensive verification of the CRS of the common SNARK schemes cannot be avoided in order to secure zkCP and zkCSP protocols.

ACKNOWLEDGEMENTS. We are grateful to the anonymous reviewers of ACM CCS 2019 for their helpful remarks.

## 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, Heidelberg, December 2017.
- [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, January 2012.
- [BCG<sup>+</sup>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, Heidelberg, August 2013.
- [BCG<sup>+</sup>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, May 2014.
- [BCG<sup>+</sup>14b] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. libsnark, 2014. Available at <https://github.com/scipr-lab/libsnark>.
- [BCG<sup>+</sup>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, May 2015.
- [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 2014*, pages 781–796. USENIX Association, August 2014.
- [BDM16] Wacław Banasik, Stefan Dziembowski, and Daniel Malinowski. Efficient zero-knowledge contingent payments in cryptocurrencies without scripts. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016, Part II*, volume 9879 of *LNCS*, pages 261–280. Springer, Heidelberg, September 2016.
- [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, May 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, Heidelberg, December 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, Heidelberg, August 1993.
- [BGG18] Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *Financial Cryptography and Data Security - FC 2018*, volume 10958 of *LNCS*, pages 64–77. Springer, 2018.
- [Bow16] Sean Bowe. pay-to-sudoku, 2016. <https://github.com/zcash-hackworks/pay-to-sudoku>.
- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 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, Heidelberg, May / June 2006.

- [But13] Vitalik Buterin. A next-generation smart contract and decentralized application platform, 2013. <http://www.ethereum.org/pdfs/EthereumWhitePaper.pdf>.
- [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 2017*, pages 229–243. ACM Press, October / November 2017.
- [Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th ACM STOC*, pages 364–369. ACM Press, May 1986.
- [DBB<sup>+</sup>15] Gaby G. Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 720–731. ACM Press, October 2015.
- [DEF18] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. FairSwap: How to fairly exchange digital goods. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 967–984. ACM Press, October 2018.
- [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, Heidelberg, December 2014.
- [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, October 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, Heidelberg, August 1987.
- [Fuc18] Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018.
- [Gab19] Ariel Gabizon. On the security of the bctv pinocchio zk-snark variant. Cryptology ePrint Archive, Report 2019/119, 2019. [ia.cr/2019/119](https://ia.cr/2019/119).
- [GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.
- [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, December 1994.
- [Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.
- [GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.
- [Les00] Lawrence Lessig. Code is law. Harvard Magazine, 2000. <https://harvardmagazine.com/2000/01/code-is-law-html>.
- [Max11] Gregory Maxwell. Zero knowledge contingent payments, 2011. [https://en.bitcoin.it/wiki/Zero\\_Knowledge\\_Contingent\\_Payment](https://en.bitcoin.it/wiki/Zero_Knowledge_Contingent_Payment).

- [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, January 1991.
- [Sza97] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday* 9, 1997.
- [TZL<sup>+</sup>17] Florian Tramèr, Fan Zhang, Huang Lin, Jean-Pierre Hubaux, Ari Juels, and Elaine Shi. Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge. In *EuroS&P 2017*, pages 19–34. IEEE, 2017.
- [Wik19] Bitcoin Wiki. Hash-locked transaction, 2019. <https://en.bitcoin.it/wiki/Hashlock>.
- [Woo14] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger, 2014. <https://gavwood.com/paper.pdf>.
- [Zca] Zcash. <http://z.cash>.