WI is Almost Enough: Contingent Payment All Over Again

Ky Nguyen
ky.nguyen@ens.fr
École Normale Supérieure, CNRS, PSL University

Miguel Ambrona
miguel.ambrona.fu@hco.ntt.co.jp
NTT Secure Platform Laboratories

Masayuki Abe
masayuki.abe.cp@hco.ntt.co.jp
NTT Secure Platform Laboratories

ABSTRACT

The problem of fair exchange consists of interchanging goods between two parties that do not trust each other. Despite known impossibility results, recent works leverage the block-chain and zero-knowledge proofs to implement zero-knowledge contingent payment (zkCP) systems that make fair exchange of digital goods possible. Implementing these systems in a secure and efficient way is a big challenge, as evidenced by several unsuccessful attempts from the literature.

Campanelli et al. (ACM CCS 2017) discovered a vulnerability on an existing zkCP proposal based on SNARKs (succinct non-interactive arguments of knowledge) and suggested several repairs. Fuchsbauer (ACM CCS 2019) found a flaw in the mentioned countermeasures. In particular, he showed that witness-indistinguishability (WI) is not sufficient for the zkCP schemes proposed by Campanelli et al. to be secure.

In this work, we observe that a slightly stronger notion of WI, that we coin trapdoor subversion WI (tS-WI), rules out Fuchsbauer's attack. We formally define security properties for CP systems and show that, under tS-WI, Campanelli et al.'s proposal indeed satisfies these properties. Additionally, we explore alternative approaches to implement ZK (other than SNARKs) and develop a prototype, using it to demonstrate their potential. Our new ideas result in a protocol to sell ECDSA signatures with contingent payment that can be executed in less than 150 milliseconds over a LAN network.

CCS CONCEPTS

• **Security and privacy → Cryptography**; *Public key (asymmetric) techniques*; Privacy-preserving protocols; Security protocols.

KEYWORDS

fair exchange; contingent payment; zero-knowledge; witness indistinguishability; garbled circuits

ACM Reference Format:
Ky Nguyen, Miguel Ambrona, and Masayuki Abe. 2020. WI is Almost Enough: Contingent Payment All Over Again. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3372297.3417888

CCS '20, November 9–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7089-9/20/11.
https://doi.org/10.1145/3372297.3417888

1 INTRODUCTION

Fair exchange and smart contracts. With the development of online commercial activities, the relationship between a buyer and a seller plays a more and more important role in the design of modern payment systems, even in decentralized systems such as cryptocurrencies. The *fair exchange problem* relates to a situation where the buyer wants some piece of valuable information from the seller, but the buyer is willing to pay only after receiving the desired information. Being even more vigilant, the buyer will consider doing business with the seller only after being convinced about the fact that *the seller has the desired information.* On the other hand, in a natural way, the seller wants to be paid as soon as the information is revealed. The forgoing problem can be applied in many practical contexts, ranging from payments for solutions to hard problem to payments for some authorized signature on a contract. Unfortunately, it is well-known that fair-exchange cannot be enforced without a trusted third party [21].

One way around this requirement is using *smart contracts* [41] in cryptocurrencies. The whole cryptocurrency system acts as a *trusted third-party* whose reliability stems from the public policies agreed by the community's members. This trusted entity is not bound to a single member anymore, but rather understood in a decentralized sense. *Ethereum* [16, 45] and *Bitcoin* [37] are typical cryptocurrencies that offer the functionality of smart contracts. For instance, Ethereum allows a member to write a smart contract which encodes, in a Turing-complete language, the predicate that needs to be satisfied for a payment to be made. The buyer can then post the smart contract making a payment to any seller who presents a piece of information s that meets the condition $f(s)$ embedded in the contract. Such information s is presumably not publicly known and valuable for the buyer. A description of contingent payment system in Ethereum was given in [42]. On the other hand, Bitcoin, which is by far the most important and popular cryptocurrency, also supports smart contracts but in a very limited way. The problem lies in its *scripting language* that facilitates only restricted ways to describe conditions of payments. One type of smart contract that can be made possible by Bitcoin is the payment made for unlocking a hash value, the so-called *hash-locked transaction* [44]: the buyer posts a smart contract on the Bitcoin blockchain stating a payment of n bitcoins to anyone who presents a preimage k of a public value y under SHA. Consequently, a natural question arises: how to make possible a fair exchange payment system on Bitcoin, given only its primitive support for hash-locked transactions? Here comes the idea of *contingent payment (CP)*, which was first studied and made possible with a protocol proposed in [36].

Zero-knowledge contingent payment systems. In a CP system, the seller will select a key k for a symmetric encryption scheme and encrypt the secret information with k to obtain a ciphertext c,

as well as a hash value $y = \mathsf{SHA}(k)$. The seller then sends both the ciphertext and hash value to the buyer, along with a proof that the secret information used to compute c, y indeed satisfies the payment condition. The buyer, upon receiving c and y, will post a hash-locked transaction for y on the Bitcoin blockchain. In order to get paid, the seller just needs to present k, which also enables the buyer to obtain the secret information decrypted from c. The contingency of the payment, i.e. the payment is made only when the seller reveals the secret information, is ensured by the hash-locked smart contract. Reciprocally, the system prevents the seller from cheating by requiring a proof of correctness for the information, which is sent together with its ciphertext and the hash value. The cryptographic proof system used by the seller must satisfy *soundness* for the buyer's own interest. Furthermore, it must be a *zero-knowledge* proof system [32] to forbid the buyer to learn any information from the proof itself. Altogether, by CP systems we mean *zero-knowledge contingent payment (zkCP)* systems.

When the notion of zkCP was first introduced in [36], there was still no generic zero-knowledge proof that was practical enough for an arbitrary predicate in zkCP. Nonetheless, with a great deal of research in ZK proof systems, especially the development of *succinct non-interactive arguments (SNARGs)* and *SNARGs of knowledge (SNARKs)* [12, 23, 28, 29, 33, 38], the situation has changed. SNARKs are an efficient form of *non-interactive ZK proof system* [13], which is set up with a *common reference string (crs)* accessible to both prover and verifier [30]. The soundness of SNARKs is totally violated unless the crs is generated by a trusted third party, as a maliciously generated crs can be used to prove arbitrary false statements. Indeed, the proof of soundness in [29] uses the fact that the crs is honestly computed. As a result, SNARKs require some kind of trusted set up, but "trust" is precisely what zkCP tries to get rid of. However, another observation comes to light: it is the buyer, who verifies the proof of the seller, for whom the soundness is important. Thus, in the initial attempts to make zkCP practical, *the buyer himself is the* crs *generator*. For instance, the implementation in [14] made use of this idea and gave the first realization of zkCP, where the payment condition is a solution for a 9×9 Sudoku puzzle.

Zero-knowledge contingent service payment systems. Before long, an attack against the implementation in [14] was presented in [18]. Furthermore, the authors proposed what they coined *zero-knowledge contingent service payment (zkCSP)*. At a high-level, zkCSP serves as a system for payment of *service*, instead of *information* as it is the case for zkCP.

As an example, consider the setting of a contingent payment system where Alice is a buyer and also a customer of a cloud storage company, represented by Bob, who is a seller. Alice stores her files on Bob's servers and pays for his service each month. On the one hand, the relation for a usual zkCP system cannot be applied in this case, because it is *the proof that Bob still keeps her files* that Alice is interested in. At the moment Alice receives a valid proof from Bob, she learns that her files are still safe and may not pay. The problem is what is being traded is not the files themselves but instead *the storage service* that Bob is offering. Therefore, the idea of zkCSP by Campanelli et al. [18] was to propose a different statement to be proved in the system so as to solve this problem. More specifically, Bob will prove to Alice that *"Either your files are kept by me and k is a* SHA *preimage of y or I* ***neither*** *keep them* ***nor*** *have a preimage of y"*.

More formally, the authors propose the following relation that will be proved by the proof system of zkCSP [18]:

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

where H is a cryptographic hash function different from SHA.

A secure zkCSP, where a Server proves to a Client knowledge of s such that $f(s) = 1$ for certain efficiently computable predicate f and will be paid for it, should satisfy the following properties [18].

- P1 If the (malicious) Server is paid, then the Server does know a satisfying value for f.
- P2 A (malicious) Client can learn no information without paying.
- P3 A (malicious) Client who pays will learn that the Server knows a satisfying value for f, but nothing more.

Fuchsbauer showed that using a *subversion witness indistinguishability*[1] *(S-WI)* argument system for the relation (1) to implement contingent service payment is not sufficient [27]. He proposed a method to modify any S-WI scheme preserving the S-WI property but making it unsuitable for contingent service payment. In a nutshell, his idea is to modify the prover algorithm so that not only does it output the proof, but it also leaks the value $f(s)$ (and thus, it cannot be applied for contingent service payment[2]). This modification does not compromise S-WI if SHA and H are claw-free, because an adversary that wins the S-WI game by submitting witnesses (s_0, r_0) and (s_1, r_1) such that $f(s_0) \neq f(s_1)$ can be transformed into an adversary that breaks the claw-freeness of SHA and H. (Note that if both witnesses are valid, $\mathsf{SHA}(r_b) = \mathsf{H}(r_{1-b})$ for either $b = 0$ or $b = 1$.)

The proof in [27] claimed that an S-WI argument system for zkCSP is not enough, but it still opens up opportunities for other directions. One main challenge is now determining what property is sufficient for an argument system to successfully realize zkCSP.

A countermeasure against attacks on pay-to-sudoku. Back to the attack in [18], it in fact demonstrated that allowing the buyer to generate the crs can lead to a breach of the ZK property. More specifically, a malicious buyer can generate a crs that still produces a valid proof for a sudoku solution, but will leak whether a cell (i, j) contains a number m or not in that solution, where i, j, m are freely chosen bythe buyer. This whole situation raises the problem of *subversion security*, in particular the property of *subversion ZK (S-ZK)*, as coined in [7, 26].

In [26], Fuschbauer studied extensively the subversion resilience of most important zkSNARKs in the literature, including the one from [12] that was used in the implementation of [14]. He showed that all these schemes will satisfy S-ZK if before using the crs, the prover does a *consistency check* w.r.t its elements. Moreover, he also showed that once the crs well-formedness is ensured, S-WI will hold unconditionally. Regarding the notion of WI, first introduced in [25], it ensures that a proof will not leak the information about which witness was used to craft it.

[1] The notion of S-WI ensures that the system is WI even if the verifier chooses the crs.
[2] Observe that all the verifier (the buyer/client) wants to learn is $f(s)$ and in these conditions they can derive it from the proof, before the payment is done.

The authors of [18] proposed checking the consistency of the crs as a countermeasure against their own attack (the seller performs those checks before producing the proof). They first propose a full check of consistency that results in S-ZK, but requires an expensive computation. The prover's runtime so far is estimated to exceed one hour in [18]. Alternatively, the authors suggested to perform a *minimal check*, which in fact does not guarantee that the zkSNARK in use is S-ZK, but only S-WI. However, also in [27], another attack against the pay-to-sudoku system was presented, successful even when the minimal checks by [18] are performed. This result shows that zkSNARK with these checks is not S-WI and it even exacerbates the situation by implying that there is no other way around to avoid the full expensive checks over the crs' elements. With more than one hour of estimated running time, the zkCP system using S-ZK zkSNARK still leaves a lot of room for improvement.

1.1 Our contributions

Formal security definitions for contingent payment. We provide formal security definitions for the properties that a contingent payment system must satisfy (see Section 3). More concretely, we define a security game that models the *soundness* of the protocol: no efficiently computable seller should be able to receive the payment without providing the buyer with the desired information (Figure 3). Furthermore, we design a security game (that resembles the ZK game) to model the fact that the buyer cannot learn any information until the payment takes place (Figure 2).

To the best of our knowledge, these properties have not been formalized in the literature before. (Although we acknowledge the work by Campanelli et al. [18] for having provided informal definitions for zero-knowledge contingent service payment.) As a consequence, previous works contained incorrect statements or proposed invalid repairs to existing security problems. Our formal treatment of these notions allows us to give precise proofs of security that make the constructions robust and reliable.

New WI security notion. We define a new security notion for proof systems that we coin *"trapdoor subversion witness indistinguishablility"* (tS-WI). Roughly, it states that the scheme is witness indistinguishable even if the verifier chooses the crs and samples the relation (possibly getting a trapdoor associated to it). For example, consider the following relation:

$$R_{f,\mathsf{H}}((c,y),r) \Leftrightarrow \{f(\mathsf{Dec}_r(c)) \wedge y = \mathsf{SHA}(r)\} \vee y = \mathsf{H}_\kappa(k,r) \ .$$

Trapdoor subversion witness indistinguishability for the above relation (formally, it should be relative to a *relation generator*) would state that the system is subversion witness indistinguishable even if the distinguisher is given a trapdoor that allows them to efficiently find preimages for H.

Formal security proof from tS-WI. In order to soften and amend the criticism to [18] about S-WI not being enough for contingent payment, we show that our new notion of *tS-WI is enough* to achieve secure contingent payment according to our formal definition (Theorem 3.5). In particular, observe that the transformations of Fuchsbauer's attacks (both the attack against CP and the one against CSP) do not preserve our tS-WI property.[3]

Interactive approach for zkCP. We consider our previous contribution of theoretical interest. However, even if tS-WI suffices for the contingent payment protocol of Campanelli et al. [18] to be secure, to the best of our knowledge, no S-WI constructions are known that are significantly more efficient than full S-ZK schemes. Implementing zkCP in this direction with SNARKs is still risky and hard to be made efficient. We take on this issue and approach it with an interactive protocol. Particularly, we replace SNARKs with an interactive ZK proof system using *garbled circuits* and *oblivious transfer*, as proposed in [35]. In addition, we take into account not only arithmetic statements but also those of *hybrid* nature. More specifically, let G be a group with generator P and let $A \in G$. Given a boolean circuit $f : \{0,1\}^m \rightarrow \{0,1\}^n$ and given $com \in G$, in section 4, we propose a zero-knowledge proof system for:

$$\mathrm{PoK}\{(s,x,t) \ : \ com = s\,A + tP \ \wedge \ f(x) = s\} \ . \tag{2}$$

We also prove that our construction is indeed a zero-knowledge proof of knowledge.

Implementation. Finally, we implement our construction of interactive proof systems for zkCP from Section 4 and present the results in Section 5. The experiments concern selling ECDSA and RSA signatures on a contract. To do so, we design a method for proving the knowledge of an ECDSA signature or an RSA signature by just proving knowledge of a discrete logarithm (see Sections 5.1, 5.2).

1.2 Other related works

In [35], Jawurek et al. proposed an efficient, interactive ZK proof system for statements that are expressed as boolean circuits. Their work was motivated by the idea of Yao's garbled circuits and it provides active security while having complexity proportional to the size of the circuit representing the relation verification function. Moreover, this protocol does not require the common *cut-and-choose* technique to provide active security, because the verifier does not have any secret information to hide. The verifier just needs to transmit one single garbled circuit, which implies a significant efficiency gain for communication time. Taking into account the recent advances in garbling schemes, e.g. the half-gate technique in [46], and in oblivious transfer, e.g. an elegant idea of a simple OT in [20] or the OT extension technique [4, 5], the protocol of [35] could result in an even more optimistic performance.

When there are algebraic operations included in the relation, e.g. group exponentiations, the idea of having a proof system for statements that involves both arithmetic and algebraic computation has inspired other works in this line, in particular the work by Chase et al. [19]. Their work originated from the question of verifying signatures on committed messages, which usually concerns a hash value of messages and an (EC)DSA or RSA signature on it. The authors proposed a method that took advantage from the approach

[3] An adversary (against WI) with the ability to invert H can choose values s_0 and s_1 such that $f(s_0) = 0$ and $f(s_1) = 1$, then sample $r_1 \xleftarrow{\$} \{0,1\}^{256}$, compute $y := \mathsf{SHA}(r_1)$ and $r_0 := \mathsf{H}^{-1}(y)$, calling its oracle on instance y and witnesses (s_0, r_0), (s_1, r_1). If the received proof leaks $f(s)$, the adversary will realize which witness was used.

in [35] to prove the non-algebraic part, and link it with sigma protocols [22] for proving the algebraic one. Additionally, their results showed the possibility to obtain an efficient proof of knowledge of signatures on a committed message using DSA, ECDSA, and RSA signature schemes. We observe that our protocols for proving knowledge of ECDSA and RSA signatures are simpler, because we are assuming that the message is public (a natural assumption in the application of signing a contract), whereas their model can provide privacy on the message. Furthermore, in the case of ECDSA signature, we are assuming that the seller is the owner of the secret key or, in other words, the signature (r, s) is freshly generated (and so the first half, r, can be sent in the clear).

Prior to our work, there was already an attempt by Banasik et al. [6] that replaces the generic non-interactive ZK building block by an interactive ZK protocol in the construction of a ZKCP system. Their protocol does not use hash-locked transactions but only standard transactions, which are preferably shorter and less error-prone. At the same time, it employs the cut-and-choose technique together with timed commitment to avoid expensive generic ZK schemes. However, it is vulnerable to the so-called *mauling attacks*: an adversary can "maul" a valid transaction in the ledger and output an equivalent but invalid transaction, e.g., one with the same input and output and of the same amount, but whose hash identifier is different from the original one. The authors of [6] were aware of this attack and pointed out that in some scenarios this attack may be an issue because many Bitcoin clients cannot handle transactions whose hash identifiers are different from what was posted in the ledger. In an attempt to resolve this issue, the authors propose a fix which prevents the mauling attack that exploits the malleability of ECDSA signatures. However, their improved construction is still vulnerable to mauling by changing the script, as mentioned in [18]. Hence, for CP systems that are implemented using the method in [6], a mauling attack can still be mounted to make the transactions unredeemable and lead to problems when creating Bitcoin contracts, see [1, 2].

Speaking of subversion resistance for SNARKs, Ben-Sasson et al. [11] provided a generic multi-party protocol for constructing the crs, where only if *all* parties collude will they be able to reproduce the trapdoor, or in other words, "cheat". Motivated by this work, Bowe et al. [15] proposed an efficient and simpler way to achieve the same goal. More interestingly, the protocol in [15] provides statistical zero-knowledge even in the case where *all players are malicious*. These interactive methods indeed cast a light on possible solutions for the challenge of subverted crs and the systems susceptible to it.

2 PRELIMINARIES

For a finite set S, we write $a \xleftarrow{\$} S$ to denote that a is uniformly sampled from S. We denote the security parameter by $\kappa \in \mathbb{N}$. Given two functions $f, g : \mathbb{N} \to [0, 1]$, we write $f \approx g$ if the difference $|f(\kappa) - g(\kappa)|$ is asymptotically smaller than the inverse of any polynomial in κ. A function f is said to be *negligible* if $f \approx 0$, whereas it is said to be *overwhelming* when $f \approx 1$. For integers $m, n \in \mathbb{N}$, we denote by $[m, n]$ the range $\{m, \ldots, n\}$ and by $[n]$ the range $[1, n]$. We denote by $\mathbb{Z}_n$ the set of integers modulo n and denote by $=_n$ equality modulo n. Given a string or an integer s, we denote by $|s|$ the length of its binary representation. For a bit-string $s \in \{0, 1\}^n$, when it is clear from the context, we abuse notation and refer to s as the integer that it represents in binary (in big-endian). We use additive notation for groups. Given a cyclic group G of order p with generator P, and given $a \in \mathbb{Z}_p$, we use $[\![a]\!]$ to denote the implicit representation aP of a in G, following [24].

For a cryptographic scheme Π with security notion sec and advantage function (of an adversary $\mathcal{A}$), $\mathsf{Adv}^{\mathsf{sec}}_{\Pi,\mathcal{A}}(\kappa)$, we define:

$$\mathsf{Adv}^{\mathsf{sec}}_{\Pi}(\kappa) \coloneqq \max_{\text{ppt } \mathcal{A}} \left\{ \mathsf{Adv}^{\mathsf{sec}}_{\Pi,\mathcal{A}}(\kappa) \right\} .$$

We explicitly write the state associated to stateful adversaries, usually denoted by σ. Given two distributions D_1 and D_2, we write $D_1 \equiv D_2$ if they are identical and $D_1 \approx D_2$ if they are computationally indistinguishable.

2.1 Zero-knowledge proof systems

A zero-knowledge (ZK) proof system [31] is a two party protocol executed between a prover and a verifier that allows the prover to convince the verifier about the validity of certain statement, without revealing any other information, e.g., why the statement is true. More formally, given a binary relation $R : \mathcal{X} \times \mathcal{W} \to \{0, 1\}$, defined over a set of statements $\mathcal{X}$ and a set of witnesses $\mathcal{W}$, let L_R be the language defined as $L_R \coloneqq \{x \in \mathcal{X} \mid \exists w \in \mathcal{W} : R(x, w) = 1\}$. A zero-knowledge proof system allows a prover in possession of $(x, w) \in R$ to convince a verifier of the fact that $x \in L_R$ without revealing any information about w. Non-interactive ZK proof systems [13] are a version of ZK proof systems where the prover sends one single message to the verifier.

Definition 2.1 (Non-Interactive Argument System). A *non-interactive argument system* (NIZK) for relation R is a triple of polynomial-time algorithms $\Pi = (\mathsf{CrsGen}, \mathsf{Prove}, \mathsf{Verify})$ where:

- $\mathsf{CrsGen}(1^\kappa) \to \mathsf{crs}$ is a probabilistic algorithm that takes a security parameter and generates a common reference string.
- $\mathsf{Prove}(\mathsf{crs}, x, w) \to \pi$ is a probabilistic algorithm that on input a crs, an instance x and a witness w, outputs a proof π.
- $\mathsf{Verify}(\mathsf{crs}, x, \pi) \to 0/1$ is a deterministic algorithm that on input a crs, an instance x and a proof π, outputs a bit representing acceptance (1) or rejection (0).

A NIZK system is *complete* if for all $(x, w) \in R$ and all $\kappa \in \mathbb{N}$, $\mathsf{crs} \leftarrow \mathsf{CrsGen}(1^\kappa)$, it holds $\mathsf{Verify}(\mathsf{crs}, x, \mathsf{Prove}(\mathsf{crs}, x, w)) = 1$. A NIZK argument system is said to have the *zero-knowledge property* if there exists a *simulator* that, without any witness (but possibly some useful piece of information like a trapdoor, or the ability of rewinding), can produce proofs that look indistinguishable from proofs produced by an honest prover in possession of a valid witness. A NIZK argument system is *knowledge-sound* if there exists a negligible function ϵ_{ks} and there exists a polynomial-time *extractor* $\mathcal{E}$ such that for any ppt adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CrsGen}(1^\kappa) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ w \leftarrow \mathcal{E}^{\mathcal{A}}(\mathsf{crs}, x, \pi) \end{array} : \begin{array}{c} \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \\ \wedge\ R(x, w) = 0 \end{array}\right] \leq \epsilon_{\mathsf{ks}}(\kappa)$$

Following the notation introduced by Camenisch and Stadler in [17], we denote a *ZK proof of knowledge of secret w satisfying $R(x, w)$ for some public R and x* as $\mathrm{PoK}\{(w) : R(x, w) = 1\}$.

A Σ-protocol is a simple 3-move interactive protocol used for building very efficient ZK proof systems. We refer to [34] for a formal definition of Σ-protocols.

2.2 Trapdoor permutations

Definition 2.2. A collection of efficiently computable functions $\mathcal{H} := \{\mathsf{H}_\kappa : \mathcal{K}_\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa\}$ is said to be a *family of trapdoor permutations* if for every $\kappa \in \mathbb{N}$ and every $k \in \mathcal{K}_\kappa$, we have that $\mathsf{H}_\kappa(k,\cdot)$ is a permutation over $\{0,1\}^\kappa$. Furthermore, there exists a pair of ppt algorithms $(\mathcal{H}.\mathsf{Gen}, \mathcal{H}.\mathsf{Inv})$ where (for certain polynomial p):

- $\mathcal{H}.\mathsf{Gen}(1^\kappa) \to (k, \tau_k)$ takes as input the security parameter κ and outputs a key $k \in \mathcal{K}_\kappa$ and a trapdoor $\tau_k \in \{0,1\}^{p(\kappa)}$.
- $\mathcal{H}.\mathsf{Inv}(\tau_k, y) \to x$ takes as input a trapdoor and a value and outputs another value x.

And such that, for every $\kappa \in \mathbb{N}$, every $y \in \{0,1\}^\kappa$ and every $(k,\tau_k) \leftarrow \mathcal{H}.\mathsf{Gen}(1^\kappa)$, it holds $\mathsf{H}_\kappa(k, \mathcal{H}.\mathsf{Inv}(\tau_k, y)) = y$. Furthermore, for every ppt adversary $\mathcal{A}$, the following advantage is negligible in κ,

$$\mathsf{Adv}^{\mathsf{TP}}_{\mathcal{H},\mathcal{A}}(\kappa) := \Pr\left[\begin{array}{l}(y,\sigma) \leftarrow \mathcal{A}(1^\kappa)\\ (k,\tau_k) \leftarrow \mathcal{H}.\mathsf{Gen}(1^\kappa)\\ x \leftarrow \mathcal{A}(k,\sigma)\end{array} : \mathsf{H}_\kappa(k,x) = y\right] .$$

For a fixed $\kappa \in \mathbb{N}$, we often write H_κ^{-1} instead of $\mathcal{H}.\mathsf{Inv}$.

2.3 Claw-freeness and collision-resistance

The notion of claw-freeness (between two functions H_1 and H_2) captures the hardness of finding two (not necessarily different) values x_1 and x_2 such that $H_1(x_1) = H_2(x_2)$.

Definition 2.3. A *function generator* FGen is a ppt algorithm which, on input the security parameter, outputs the description of two efficiently computable functions H_1 and H_2 with range $\{0,1\}^\kappa$. A function generator FGen is said to be *claw-free* if for every ppt adversary $\mathcal{A}$, the following advantage is negligible in κ,

$$\mathsf{Adv}^{\mathsf{CF}}_{\mathsf{FGen},\mathcal{A}}(\kappa) := \Pr\left[\begin{array}{l}(H_1,H_2) \leftarrow \mathsf{FGen}(1^\kappa)\\ (x_1,x_2) \leftarrow \mathcal{A}(H_1,H_2)\end{array} : H_1(x_1) = H_2(x_2)\right].$$

On the other hand, collision-resistance refers to the hardness of finding two different inputs with the same image.

Definition 2.4. A family of efficiently computable functions $\mathcal{F} = \{F_\kappa : \{0,1\}^* \to \{0,1\}^\kappa\}_{\kappa\in\mathbb{N}}$ is said to be *collision-resistant* if for every ppt $\mathcal{A}$ the following probability is negligible in κ:

$$\mathsf{Adv}^{\mathsf{CR}}_{\mathcal{F},\mathcal{A}}(\kappa) := \Pr\left[(x_1,x_2) \leftarrow \mathcal{A}(1^\kappa) : F_\kappa(x_1) = F_\kappa(x_2)\right] .$$

2.4 Encryption

Definition 2.5. A *private-key encryption* scheme $\mathcal{E}$ is a triple of ppt algorithms $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ over a key space $\mathcal{K}$, a message space $\mathcal{M}$ and a ciphertext space C, where:

- $\mathsf{Gen}(1^\kappa) \to k$ is a ppt algorithm that on input the security parameter outputs a key $k \in \mathcal{K}$.
- $\mathsf{Enc}(k,m) \to c$ is a ppt algorithm that on input a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$, outputs a ciphertext $c \in C$.
- $\mathsf{Dec}(k,c) \to m$ is a deterministic algorithm that on input a key $k \in \mathcal{K}$ and a ciphertext $c \in C$, outputs a message $m \in \mathcal{M}$ or $\perp$.

We often write $\mathsf{Enc}_k(m)$ and $\mathsf{Dec}_k(c)$ instead of $\mathsf{Enc}(k,m)$ and $\mathsf{Dec}(k,c)$ respectively. We require an encryption scheme be *correct*, that is, for every $m \in \mathcal{M}$, all $\kappa \in \mathbb{N}$ and all $k \leftarrow \mathsf{Gen}(1^\kappa)$, it holds $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$.

We say an encryption scheme $\mathcal{E}$ has *indistinguishable encryptions in the presence of an eavesdropper* if for every ppt adversary $\mathcal{A}$, the following advantage is negligible in κ,

$$\mathsf{Adv}^{\mathsf{eav}}_{\mathcal{E},\mathcal{A}}(\kappa) := \left|\Pr\left[\begin{array}{l}(m_0,m_1,\sigma) \leftarrow \mathcal{A}(1^\kappa)\\ k \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa)\\ b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathcal{E}.\mathsf{Enc}_k(m_b)\\ b' \leftarrow \mathcal{A}(c,\sigma)\end{array} : b' = b\right] - \frac{1}{2}\right| ,$$

where the adversary is required to follow the constraint $|m_0| = |m_1|$.

We say an encryption scheme $\mathcal{E}$ is secure against one-time *key recovery* if for every ppt adversary $\mathcal{A}$, the following advantage is negligible in κ,

$$\mathsf{Adv}^{\mathsf{key\text{-}rec}}_{\mathcal{E},\mathcal{A}}(\kappa) := \Pr\left[\begin{array}{ll}m \leftarrow \mathcal{A}(1^\kappa); & c \leftarrow \mathcal{E}.\mathsf{Enc}_k(m)\\ k \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa); & k' \leftarrow \mathcal{A}(c)\end{array} : k' = k\right] .$$

Observe that these notions of security are quite weak and both are implied by the standard IND-CPA security (which is similar to eav security where the adversary also interacts with an encrypting oracle). However, both notions are sufficient for our security proofs.

2.5 Garbled Circuits

As in [35], we follow the definitions of garbled circuits from [8], simplifying the terminology to capture our necessities. We also focus on projective garbling schemes and consider a verification algorithm.

Definition 2.6 (Garbled circuit). A *garbled circuit scheme* is a triple $\mathcal{GC}$ of ppt algorithms $\mathcal{GC} = (\mathsf{Garble}, \mathsf{Eval}, \mathsf{Verify})$ where:

- $\mathsf{Garble}(1^\kappa, f) \to (\mathsf{GC}, \{L_0^{(i)}, L_1^{(i)}\}_{i\in[m]}, \{Z_0^{(i)}, Z_1^{(i)}\}_{i\in[n]})$ on input κ and the description of a function $f : \{0,1\}^m \to \{0,1\}^n$, outputs a garbled circuit, a set of m input label pairs and a set of n output label pairs.
- $\mathsf{Eval}(\mathsf{GC}, \{L^{(i)}\}_{i\in[m]}) \to \{Z^{(i)}\}_{i\in[n]}$ on input a garbled circuit and a set of input labels, outptus a set of output labels.
- $\mathsf{Verify}(\mathsf{GC}, f, \{L_0^{(i)}, L_1^{(i)}\}_{i\in[m]}, \{Z_0^{(i)}, Z_1^{(i)}\}_{i\in[n]}) \to \{0,1\}$ on input a garbled circuit, a function and a set of input label pairs and a set of output label pairs, outputs a bit (1 for *acceptance* and 0 for *rejection*).

We say a garbled circuit scheme is *correct* if for all polynomial-size (in κ) functions $f : \{0,1\}^m \to \{0,1\}^n$ and all $x \in \{0,1\}^m$, for $y = f(x)$, the following probability is overwhelming in κ,

$$\Pr\left[\begin{array}{l}(\mathsf{GC}, \{L_0^{(i)}, L_1^{(i)}\}_{i\in[m]}, \{Z_0^{(i)}, Z_1^{(i)}\}_{i\in[n]}) \leftarrow \mathsf{Garble}(1^\kappa, f)\\ \{\hat{Z}^{(i)}\}_{i\in[n]} \leftarrow \mathsf{Eval}(\mathsf{GC}, \{L_{x_i}^{(i)}\}_{i\in[m]})\ :\ \hat{Z}^{(i)} = Z_{y_i}^{(i)}\ \forall i \in [n]\end{array}\right].$$

We say a garbled circuit scheme has *authenticity* if for all polynomial-size (in κ) functions $f : \{0,1\}^m \to \{0,1\}^n$ and all $x \in \{0,1\}^m$, for $y = f(x)$, and for all ppt adversaries $\mathcal{A}$, the following probability, denoted by $\mathsf{Adv}^{\mathsf{auth}}_{\mathcal{GC},\mathcal{A}}(\kappa)$, is negligible in κ,

$$\Pr\left[\begin{array}{l}(\mathsf{GC}, \{L_0^{(i)}, L_1^{(i)}\}_{i\in[m]}, \{Z_0^{(i)}, Z_1^{(i)}\}_{i\in[n]}) \leftarrow \mathsf{Garble}(1^\kappa, f)\\ (Z^*, i^*) \leftarrow \mathcal{A}(f, x, \mathsf{GC}, \{L_{x_i}^{(i)}\}_{i\in[m]})\ :\ Z^* = Z_{\neg y_{i^*}}^{(i^*)}\end{array}\right].$$

Game $\mathrm{WI}^{\Pi,\mathcal{RG}}_{\mathcal{A}}(\kappa)$ / $\mathrm{S\text{-}WI}^{\Pi,\mathcal{RG}}_{\mathcal{A}}(\kappa)$ / $\mathrm{tS\text{-}WI}^{\Pi,\mathcal{RG}}_{\mathcal{A}}(\kappa)$:

$b \xleftarrow{\$} \{0,1\}$
$(R,\tau) \leftarrow \mathcal{RG}(1^\kappa)$
$\mathsf{crs} \leftarrow \Pi.\mathsf{CrsGen}(R)$ / $(\mathsf{crs},\sigma) \leftarrow \mathcal{A}(R)$ / $(\mathsf{crs},\sigma) \leftarrow \mathcal{A}(R,\tau)$
$b' \leftarrow \mathcal{A}^{O(R,\mathsf{crs},b,\cdot,\cdot,\cdot)}(\mathsf{crs},\sigma)$
return $(b' = b)$

$O(R,\mathsf{crs},b,x,w_0,w_1)$:
$\pi \leftarrow \Pi.\mathsf{Prove}(R,\mathsf{crs},x,w_b)$
if $R(x,w_0) = 0$ or $R(x,w_1) = 0$, return $\perp$
otherwise, return π

Figure 1: Subversion WI games.

We say a garbled circuit scheme is *verifiable* if for all polynomial-size (in κ) functions $f : \{0,1\}^m \to \{0,1\}^n$ and all $x \in \{0,1\}^m$, for $y = f(x)$ and for all ppt adversaries $\mathcal{A}$, the following probability is negligible in κ,

$$\Pr\left[\begin{array}{l}(\mathsf{GC},\{L_0^{(i)},L_1^{(i)}\}_{i\in[m]},\{Z_0^{(i)},Z_1^{(i)}\}_{i\in[n]}) \leftarrow \mathcal{A}(1^\kappa,f)\\ \{\hat{Z}^{(i)}\}_{i\in[n]} \leftarrow \mathsf{Eval}(\mathsf{GC},\{L_{x_i}^{(i)}\}_{i\in[m]}) \ :\\ \quad \mathsf{Verify}(\mathsf{GC},f,\{L_0^{(i)},L_1^{(i)}\}_{i\in[m]},\{Z_0^{(i)},Z_1^{(i)}\}_{i\in[n]}) = 1\\ \wedge\ \exists i \in [n],\ \hat{Z}^{(i)} \neq Z_{y_i}^{(i)}\end{array}\right].$$

2.6 Committing oblivious transfer

For simplicity in the presentation of constructions depending on this primitive, we describe the ideal functionality associated to committing oblivious transfer, $\mathcal{F}_{COT}$. We assume a fixed $n \in \mathbb{N}$:

(1) *Choose:* on input (choose, $\{b_i\}_{i\in[n]}$) from the receiver, where $b_i \in \{0,1\}, \forall i \in [n]$, inform receiver that a choice was received.
(2) *Transfer:* on input (transfer, $\{m_0^{(i)}, m_1^{(i)}\}_{i\in[n]}$) from the sender, send messages $\{m_{b_i}^{(i)}\}_{i\in[n]}$ to the receiver.
(3) *Open:* on input (open-all) from the sender, reveal all pairs $\{m_0^{(i)}, m_1^{(i)}\}_{i\in[n]}$ to the receiver and halt.

A valid committing oblivious transfer construction is such that for every (possibly malicious) receiver, there is a simulator that can extract input bits b_i, invoke the ideal functionality obtaining m_{b_i} and use these values to simulate the receiver's real world view. We will use such a simulator in the proof of Theorem 4.1.

2.7 Witness indistinguishability

The notion of *witness indistinguishability* [25] tries to model the fact that proofs do not leak any information about the witness they have been created with. (Observe that proofs may leak information about the witness, as long as such information is common for all valid witnesses.) In Figure 1 we detail the security game of three different variants of this notion: the standard *witness indistinguishability*, *subversion witness indistinguishability* and *trapdoor subversion witness indistinguishability*. The latter is a new notion that we formally define below. We refer to Appendix A for details about the former two.

Trapdoor subversion witness indistinguishability. Our new notion of WI[4] requires that the system remain WI even when the adversary generates the crs and is given a trapdoor associated with the relation to be proven.

Definition 2.7 (tS-WI). A NI proof system $\Pi = (\mathsf{CrsGen}, \mathsf{Prove}, \mathsf{Verify})$ with respect to a relation generator $\mathcal{RG}$ is *trapdoor subversion witness indistinguishable* if for all ppt $\mathcal{A}$, the following advantage is negligible in κ,

$$\mathsf{Adv}^{\text{tS-WI}}_{\Pi,\mathcal{A}}(\kappa) := \left|\Pr\left[\mathrm{tS\text{-}WI}^{\Pi,\mathcal{RG}}_{\mathcal{A}}(\kappa)\right] - \frac{1}{2}\right| .$$

3 CONTINGENT PAYMENT FROM CSP

In the work by Campanelli et al. [18], the authors explored the idea of using the relation designed for contingent service payment for *information* contingent payment. The main motivation is that the witness indistinguishability of the underlying NI system *could* be enough to completely hide the information about the secret (thanks to the disjunctive structure on the property). Unfortunately, Fuchsbauer showed that WI is not enough for this application [27]. More concretely, he proposed NI schemes for the *CP-from-CSP relation* that are WI and can lead to completely insecure contingent payment systems, where the buyer can learn all the information before making the payment. In this section we show that our notion of tS-WI is indeed enough to achieve a secure CP based on the following relation (used in [18]):

Definition 3.1 (CP-from-CSP relation generator). Let $\mathcal{E} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme and let $f : \{0,1\}^* \to \{0,1\}$ be a predicate. Let $\mathcal{H} := \{\mathsf{H}_\kappa : \mathcal{K}_\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa\}$ be a trapdoor permutation, and let $\mathcal{F} := \{\mathsf{F}_\kappa : \{0,1\}^* \to \{0,1\}^\kappa\}$ be a hash function family. We define the *CP relation generator* $\mathcal{RG}_{\mathsf{CP}}$ as:

$\mathcal{RG}_{\mathsf{CP}}(1^\kappa)$:
$(k,\tau_k) \leftarrow \mathcal{H}.\mathsf{Gen}(1^\kappa)$;
$R((c,y),r) := \{f(\mathsf{Dec}_r(c)) \wedge y = \mathsf{F}_\kappa(r)\} \vee y = \mathsf{H}_\kappa(k,r)$;
return (R,τ_k)

The only difference with the relation in [18] is that we require H be a trapdoor permutation, a technical requirement in our security proof.

Remark 1. We assume that R implicitly includes all the necessary information to be evaluated efficiently. In particular, it includes k and the description of f, F_κ and H_κ.

Remark 2. In practice (and especially if this protocol is implemented over the Bitcoin blockchain) F_κ will be replaced by the SHA function. However, for formality, security notions need to be defined asymptotically.

3.1 Contingent payment security from tS-WI

We want to model the fact that the Client cannot learn anything about the value s that the Server holds until the payment is performed. Our security game from Figure 2 captures this fact, where the Client plays the role of the adversary $\mathcal{A}$.

[4] We abbreviate both *witness indistinguishable* and *witness indistinguishability* to WI.

Game $\text{InfoCP}^{\Pi}_{\mathcal{A}}(\kappa)$:

$b \xleftarrow{\$} \{0,1\}$
$(R, \tau_k) \leftarrow \mathcal{RG}_{\text{CP}}(1^\kappa)$
$(\text{crs}, \sigma) \leftarrow \mathcal{A}(R)$
$b' \leftarrow \mathcal{A}^{O(R,\text{crs},b,\cdot)}(\sigma)$
return $(b' = b)$

$O(R, \text{crs}, b, s)$:

if $f(s) = 0$, return $\perp$
$r \leftarrow \mathcal{E}.\text{Gen}(1^\kappa)$
if $b = 0$: $c \coloneqq \text{Enc}_r(s)$; $y \coloneqq \text{F}_\kappa(r)$; $w \coloneqq r$
if $b = 1$: $c \coloneqq \text{Enc}_r(0^{|s|})$; $w \xleftarrow{\$} \{0,1\}^\kappa$; $y \coloneqq \text{H}_\kappa(k, w)$
$\pi \leftarrow \Pi.\text{Prove}(R, \text{crs}, (c, y), w)$
return (c, y, π)

Figure 2: Information contingent payment security game.

Definition 3.2. A NI argument system Π is secure for contingent payment with respect to relation generator $\mathcal{RG}_{\text{CP}}$ if for every ppt adversary $\mathcal{A}$, the following advantage is negligible in κ:

$$\text{Adv}^{\text{InfoCP}}_{\Pi,\mathcal{A}}(\kappa) \coloneqq \left| \Pr\left[\text{InfoCP}^{\Pi}_{\mathcal{A}}(\kappa)\right] - \frac{1}{2} \right| .$$

Our next theorem establishes that tS-WI is enough to achieve InfoCP security. Its detailed proof is given in Appendix B.1.

Theorem 3.3. *Let Π be an NI argument system for the CP relation generator $\mathcal{RG}_{\text{CP}}$. Let F_κ be a random oracle for every $\kappa \geq 1$. For every ppt adversary $\mathcal{A}$ against the* InfoCP *game making at most q_O queries to its oracle and at most q_F queries to the random oracle, there exists a ppt adversary $\mathcal{B}$ such that for every $\kappa \in \mathbb{N}$,*

$$\text{Adv}^{\text{InfoCP}}_{\Pi,\mathcal{A}}(\kappa) \leq \text{Adv}^{\text{tS-WI}}_{\Pi,\mathcal{B}}(\kappa) + q_O\left(q_\text{F}\text{Adv}^{\text{key-rec}}_{\mathcal{E}}(\kappa) + \text{Adv}^{\text{eav}}_{\mathcal{E}}(\kappa)\right) .$$

One can think of the InfoCP security game as the *zero-knowledge* security game for relation f. That is, the adversary has access to an oracle that, on input a valid witness s (i.e., such that $f(s) = 1$), receives a normal proof (the branch in O with $b = 0$) or a "simulated" proof (the branch in O with $b = 1$), but the adversary cannot distinguish between the two. However, in this scenario everybody can compute simulated proofs (no trapdoor is needed, just the ability to run H forwards). Observe that this is not a problem since, for the application of contingent payment, the *soundness* requirement states that without knowing secret s, it must be hard to compute a valid proof and at the same time to be able to withdraw the money. More precisely, it should be hard to compute a valid proof (c, y, π) together with a preimage r of y under F_κ such that $f(\text{Dec}_r(c)) = 0$. We formalize this notion in the game presented in Figure 3.

Definition 3.4. The contingent payment protocol based on NI argument system Π with respect to relation generator $\mathcal{RG}_{\text{CP}}$ is *sound* if for every ppt adversary $\mathcal{A}$, the following advantage is negligible in κ:

$$\text{Adv}^{\text{sndss}}_{\Pi,\mathcal{A}}(\kappa) \coloneqq \Pr\left[\text{InfoCP-Soundness}^{\Pi}_{\mathcal{A}}(\kappa) = 1\right] .$$

We now show that the protocol is sound as long as $\mathcal{H}$ and $\mathcal{F}$ are claw-free and $\mathcal{F}$ is collision-resistant.

Game $\text{InfoCP-Soundness}^{\Pi}_{\mathcal{A}}(\kappa)$:

$\text{crs} \leftarrow \text{CrsGen}(1^\kappa)$
$(R, \tau_k) \leftarrow \mathcal{RG}_{\text{CP}}(1^\kappa)$
$((c, y, \pi), r) \leftarrow \mathcal{A}(\text{crs}, R)$
$s \coloneqq \text{Dec}_r(c)$
return $\Pi.\text{Verify}(\text{crs}, (c, y), \pi) = 1 \ \wedge \ y = F_\kappa(r) \ \wedge \ f(s) \neq 1$

Figure 3: Information contingent payment soundness game.

Theorem 3.5 (Soundness). *For every ppt adversary $\mathcal{A}$, it holds:*

$$\text{Adv}^{\text{sndss}}_{\Pi,\mathcal{A}}(\kappa) \leq \epsilon_{\text{ks}}(\kappa) + \text{Adv}^{\text{CF}}_{\mathcal{F},\mathcal{H}}(\kappa) + \text{Adv}^{\text{CR}}_{\mathcal{F}}(\kappa)$$

where the probability is taken over the coins of $\mathcal{A}$ and ϵ_{ks} represents the knowledge-soundness error of Π.

Proof. Considering the knowledge-soundness extractor, $\mathcal{E}^{\mathcal{A}}$, of Π, the above probability can be upper-bounded by $\epsilon_{\text{ks}}(\kappa)$ plus

$$\Pr\left[\begin{array}{rc} \text{crs} \leftarrow \text{CrsGen}(1^\kappa) & \\ (R, \tau_k) \leftarrow \mathcal{RG}_{\text{CP}}(1^\kappa) & \\ ((c, y, \pi), r) \leftarrow \mathcal{A}(\text{crs}, R) & : \\ s \coloneqq \text{Dec}_r(c) & \\ r' \leftarrow \mathcal{E}^{\mathcal{A}}(\text{crs}, (c, y), \pi) & \end{array} \begin{array}{c} \Pi.\text{Verify}(\text{crs}, (c, y), \pi) = 1 \\ \wedge \ y = F_\kappa(r) \ \wedge \ f(s) \neq 1 \ \wedge \\ \Big((y = F_\kappa(r') \wedge f(\text{Dec}_{r'}(c)) = 1) \\ \vee \ y = H_\kappa(r')\Big) \end{array}\right]$$

Now, $\mathcal{A}$ combined with $\mathcal{E}^{\mathcal{A}}$ can be seen as an adversary against the claw-freeness of function generators $\mathcal{F}, \mathcal{H}$ or against the collision-resistance of $\mathcal{F}$. (Observe that $f(s) \neq 1 \wedge f(\text{Dec}_{r'}(c)) = 1$ implies that $r \neq r'$.) Therefore, the above probability can be bounded by $\text{Adv}^{\text{CF}}_{\mathcal{F},\mathcal{H}}(\kappa)+\text{Adv}^{\text{CR}}_{\mathcal{F}}(\kappa)$. This analysis gives us the final bound stated in the theorem. □

Remark 3. The statement from Theorem 3.3 is asymmetric in the sense that we provide a reduction from tS-WI through adversary $\mathcal{B}$, but we do not consider adversaries against the encryption scheme $\mathcal{E}$. This is to highlight the importance of the tS-WI notion in the proof. Furthermore, some encryption schemes (see the next remark) can be proven to have negligible key-rec and eav advantages independently of the (possibly unbounded) adversary.

Remark 4. Observe that, in Theorem 3.3 the properties we require on the encryption scheme are one-time *key recovery* security and one-time *indistinguishability* (see Definition 2.5), which are weaker than the standard IND-CPA. This opens the possibility of using a modified version of one-time-pad (OTP) as the encryption algorithm. Namely,

$\text{Gen}(1^\kappa)$:	$\text{Enc}(k_1 \| k_2, m)$:	$\text{Dec}(k_1 \| k_2, c)$:
$k_1, k_2 \xleftarrow{\$} \{0,1\}^\kappa$	$c \coloneqq m \oplus k_1$	$m \coloneqq k_1 \oplus c$
return $k_1 \| k_2$	return c	return m

where the second part of the key is only used to achieve protection against *key recovery*. The main advantage of this encryption scheme is its simplicity, so it has no overhead on the relation for the proof system. However, this would only be a good option for functions f that take short inputs[5], because a long message implies a long encryption key, which would require several iterations of the SHA

[5]For instance, Sudoku, the classical proof of concept example, where a bit length of 256 is enough for encoding a solution, since $81 \log_2(9) < 257$, but not all cells need to be given (some are part of the puzzle).

computation, probably the bottleneck of relation R. If f takes long inputs, a block cipher (that can handle arbitrarily long messages with a constant-size key), e.g. AES, would be a better choice.

Remark 5. Observe that the *random oracle* assumption on F_κ is not necessary, as long as $\mathcal{E}$ and $\mathcal{F}$ are such that for every s,

$$\left(\mathcal{E}.\mathsf{Enc}_r(s),\ F_\kappa(r)\right) \approx \left(\mathcal{E}.\mathsf{Enc}_r(0^{|s|}),\ u\right)$$

for $r \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa)$ and u sampled uniformly from the range of F_κ. This becomes clear in the proof of Theorem 3.3 (see Appendix B.1). On the other hand, the standard approach to reason about SHA is to assume it is a random oracle and it is not clear how to prove that the above condition holds for the SHA function otherwise.

4 ZK FROM GARBLED CIRCUITS

While the result in the previous section establishes a theoretical ground for CP, its instantiation will suffer from a considerable overhead due to the inevitable use of algebraic structure for the trapdoor permutation. Optimization techniques from [19] allow to integrate algebraic relations into arithmetic ones and help to shorten the proof. Nevertheless, having a trapdoor permutation whose output is as large as 4096 bits of RSA modulus, for example, in the on-chain part of the statement remains problematic.

In this section we explore another direction to make contingent payment more practical. We observe that the proof is useful only for the buyer's sake. Therefore, there seems to be no strong reason to neither keep the proof succinct nor keep the proof system itself non-interactive, as both parties can engage in it in an off-chain way.

Our goal is to replace SNARKs by another interactive proof system, which will be used to prove a statement of the form *"I have a preimage k of y under* SHA *and the decryption (under k) of the ciphertext c will satisfy predicate f"*. This statement involves an arithmetic function: SHA. Constructions for such type of statements have been studied thoroughly in the literature. For instance, in [35], the authors proposed a construction based on *garbled circuit (GC)* and *oblivious transfer (OT)*. Roughly, their protocol works as follows:

- The verifier, given as input x, will garble a circuit implementing the function $f(x, \cdot)$. The prover, given as input (x, w), gets the input labels corresponding to the bits of w via oblivious transfer and evaluates the garbled circuit, obtaining an output label. This label is sent to the verifier in committed form.
- Now, the verifier reveals their coins used during the garbling process and the OT, to convince the prover about the verifier's honest behaviour. Only after being convinced will the prover open the commitment to the output label.
- Finally, the verifier will accept the proof if and only if the received value matches the output label associated to 1.

First of all, employing the above interactive protocol will help us avoid the burden of computing and verifying the crs, as there is no crs anymore. In addition, various techniques towards efficient GC and OT have been developed in the literature, e.g. in [46] or [4], that will help the protocol run more efficiently compared to SNARKs. We will elaborate on its integration into the information CP system in the following subsections.

4.1 Hybrid statements

Although the protocol from [35], which we call *ZK-from-GC*, suits us well as a building block for our information CP system, it can be directly used only when the relations can be transformed efficiently into arithmetic circuits. Unfortunately, some sophisticated applications of information CP may not fall into this category. For example, the buyer can be interested in an ECDSA signature by some specific authority on a contract, and the information CP used in this case could have the statement of the form: *"I have a preimage k of y under* SHA*, and the ciphertext c is an AES encryption (under k) of a valid ECDSA signature s"*, where (c, y) is the public input. Roughly speaking, the arithmetic clause can be proven efficiently using the ZK-from-GC protocol, while its algebraic counterpart can be carried out by standard proofs of knowledge.

There are serveral works that provide proof systems for such hybrid statements. For example, in [19] the authors combine the method from [35] with Σ-PoKs. Their idea is to commit to the private input by a Pedersen-like commitment, then execute the ZK-from-GC on the arithmetic clause and finally prove the algebraic clause *on the committed inputs*. As a novel contribution, we propose a different and natural approach to link from the *output labels* of the prover in ZK-from-GC to the PoKs of the algebraic part. Let G be a group with generator P and let $A \in G$. Given a boolean circuit $f : \{0,1\}^m \to \{0,1\}^n$ and given $com \in G$, in Figure 4 we propose a zero-knowledge proof system for:

$$\mathrm{PoK}\left\{(s, x, t)\ :\ com = s\,A + tP\ \wedge\ f(x) = s\right\}\ . \tag{3}$$

This system can be used, for example, to implement the contingent-payment relation for selling ECDSA signatures. More concretely, the prover may produce a commitment com of their signature s, encrypt it under a freshly generated key k, $c \leftarrow \mathsf{Enc}_k(s)$ and compute the hash of the key, $y = \mathsf{SHA}(k)$. It will then send (c, y) to the verifier and run the above protocol for function $f_{(c,y)}(k) \coloneqq \mathsf{Dec}_k(c)$ if $y = \mathsf{SHA}(k)$ else $\perp$. After this, the seller will perform a zero-knowledge proof of knowledge of an opening to com that is a valid ECDSA signature. This can be done very efficiently with a Σ-protocol given the algebraic nature of the ECDSA verification (see Section 5.1). Due to the binding property of the Pedersen commitment, the seller can only know one opening of com and thus, the buyer can safely assume that the value that satisfies the ECDSA signature is an output of $f_{(c,y)}$, which is all the buyer needs to know before performing the transaction.

4.2 Our construction

In Figure 4, we present our ZK construction for relation (3). Roughly, the verifier will produce a garbled circuit of function f and encrypt field elements $z_b^{(i)}$ using the output labels of the circuit. These elements are correlated in pairs satisfying $z_1^{(i)} - z_0^{(i)} = \delta$ for some value $\delta \in \mathbb{Z}_p$. The authenticity of the garbled circuit scheme guarantees that the prover can only recover one value of each pair from the evaluation of the garbled circuit (actually, $z_{s_i^*}^{(i)}$ for every $i \in [n]$, being $s^* = f(x^*)$, where x^* is the input requested in the COT). In order to convince the verifier, the prover will have to compute a group element $\hat{B}$ and, after receiving δ, argue knowledge (in zero-knowledge) of a representation of com and B in base (A, P) with

the same A-coefficient. Here, B is a group element defined as:

$$B := \delta^{-1}(\hat{B} - \sum_{i=1}^{n} 2^{i-1} z_0^{(i)} A) \ .$$

The prover can compute a representation of B in base (A, P) after receiving δ. Note that, for all s, $z_0^{(i)} = z_{s_i}^{(i)} - s_i\delta$ and thus:

$$B := \delta^{-1}(\hat{B} - \sum_{i=1}^{n} 2^{i-1} z_{s_i^*}^{(i)} A) + s^* A \ .$$

However, such representation must have the same A-coefficient as the representation known for *com*, but the prover does not know δ at the moment of choosing $\hat{B}$. Intuitively, the only way the prover can control the representation of B that they will know (after receiving δ) is to choose the representation of $\hat{B}$ such that $(\hat{B} - \sum_{i=1}^{n} 2^{i-1} z_{s_i^*}^{(i)} A)$, i.e., the factor that is multiplied by the inverse of δ, has a zero A-coefficient. Nevertheless, in that case the prover will know a representation of B where the A-coefficient is exactly s^*. Consequently, a successful prover must have requested COT inputs for a value x^* such that $s^* = f(x^*)$ is the A-coefficient of the given representation of *com*, as desired. Security against malicious verifiers is achieved because the prover verifies the garbled circuit before producing the final PoK. We formalize the above intuition with this theorem:

Theorem 4.1. *Let $\mathcal{GC} = (\mathsf{Garble}, \mathsf{Eval}, \mathsf{Verify})$ be a garbled circuit scheme following the definition in Section 2.5 and $\mathcal{F}_{COT}$ be the ideal functionality for committing oblivious transfer (as defined in Section 2.6). Let $H : \{0,1\}^* \to \mathbb{Z}_p$ be a random oracle. Then the protocol described in Figure 4 is a zero-knowledge proof of knowledge system for relation (3).*

Proof of Theorem 4.1. *Completeness.* If both parties are honest, the prover will receive the labels associated to $s = f(x)$, i.e., $\hat{z}_{s_i}^{(i)} = z_{s_i}^{(i)}$ for all $i \in [n]$, due to the correctness of the garbled circuit scheme. Also, note that $z_0^{(i)} = z_{s_i}^{(i)} - s_i\delta$, for every $i \in [n]$, therefore,

$$\begin{aligned} B &:= \delta^{-1}\left(\hat{t}\,P + \textstyle\sum_i 2^{i-1} z_{s_i}^{(i)} A - \sum_i 2^{i-1} z_0^{(i)} A\right) \\ &= \delta^{-1}\left(\hat{t}\,P + \textstyle\sum_i 2^{i-1} z_{s_i}^{(i)} A - \sum_i 2^{i-1} (z_{s_i}^{(i)} - s_i\delta) A\right) \\ &= s\,A + \delta^{-1}\hat{t}\,P \ . \end{aligned}$$

Consequently, given the completeness of ZK PoK at the last step of the protocol, the verifier will accept the proof.

Zero-knowledge. For every potentially malicious verifier, we construct a simulator, that (without a valid witness) simulates a real prover. In the first step of the protocol, the simulator performs the COT step by using an arbitrary value as chosen bits. The hiding property of the oblivious transfer makes the simulation indistinguishable from the real execution so far. The simulator then runs the code of the malicious verifier, producing a garbled circuit and proceeds with the transfer of the COT protocol (as the malicious verifier would do). The simulator simulates $\hat{B}$ as a uniformly chosen group element (the simulation is perfect, given that the blinding factor $\hat{t} \in \mathbb{Z}_p$ in the real execution is also chosen uniformly). The verifier now extracts the labels $\{L_0^{(i)}, L_1^{(i)}\}_{i\in[m]}$ from the $\mathcal{F}_{COT}$ functionality after open-all and, computes the values to be transmitted, δ and $\{Z_0^{(i)}, Z_1^{(i)}\}_{i\in[n]}$, as the verifier. It now performs the verification of the garbled circuit with the computed input and output labels, as the real prover would do. If the verification of the circuit fails, the simulator halts. Otherwise, it simulates the zero-knowledge proof of knowledge from the last step of the protocol. It is not hard to see that, if the garbled circuit scheme is verifiable, the simulation is indistinguishable from the interaction with a real prover, because when the circuit passes the verification, the labels recovered by the real prover can only take one value and (as shown in the proof of *completeness*) they will lead to a valid witness (s, t, t'), so the final ZK proof of the real prover will be accepted.

Extractability. Given a potentially malicious prover, we describe an extractor that produces a valid witness for certain statement with a comparable probability to the probability that the prover produces an accepting transcript on that statement. Assume that the possibly malicious prover runs in time τ, makes at most q queries to the random oracle and succeeds with probability ρ. Given *com*, A, f, run the prover and extract $\{x_i^*\}_{i\in[m]}$ from the first step of the protocol, by simulating (see Section 2.6) the $\mathcal{F}_{COT}$ functionality and define $s^* := f(x^*)$. Then, the extractor continues the protocol as an honest verifier and runs the prover as the prover's code indicates (computing $\hat{B}$, etc), it also simulates the random oracle queries made by the adversary by sampling uniformly random values from $\mathbb{Z}_p$. Now, assume that the prover completes an accepting transcript. The extractor will run the extractor of the PoK from the final step of the protocol, getting values (s, t, t') such that $com = s\,A + t\,P$ and $\hat{B} = s\,A + t'P$. If $s =_p s^*$, the extractor will output the witness (s^*, x^*, t). Otherwise, the extractor will halt. We can conclude the proof by arguing that the probability that the extractor does not halt is negligible. In particular, we will show that if ρ is non-negligible and τ is polynomial in the security parameter, the probability that the extractor halts is upper-bounded by $\mathsf{Adv}_G^{\mathrm{dlog}}(\kappa)$. To see this, we will leverage the above extractor to build an adversary $\mathcal{B}$ against the discrete logarithm problem. Assume $\mathcal{B}$ is given the discrete logarithm instance (p, G, P, A) and must find a such that $A = aP$. Adversary $\mathcal{B}$ will proceed as the above extractor, but it will additionally fork its execution after $\hat{B}$ has been fixed by the prover, by providing the prover with a different δ_2 (let δ_1 be the value of δ provided in the first execution). Adversary $\mathcal{B}$ then simulates the random oracle queries possibly made by the prover (after receiving δ_2) on values $Z_{\neg s_i}^{(i)}$ in order to satisfy the equations $c_1^{(i)} - c_0^{(i)} = \delta_2 + H(Z_1^{(i)}) - H(Z_0^{(i)})$, for all $i \in [n]$. Observe that, given the authenticity of the garbled circuit scheme, this second execution of the prover will be successful with probability at least $\rho' = \rho - \mathsf{Adv}_{\mathcal{GC}}^{\mathsf{auth}}(\kappa)$. If the prover is not successful after the fork, $\mathcal{B}$ will try again with a different value of δ_2 until the prover succeeds. By the Forking Lemma (Appendix A.3), in expected time $O(\tau q/\rho'^2)$, the adversary will obtain values (s_1, t_1, t_1') and (s_2, t_2, t_2') such that:

$$com = s_1\,A + t_1\,P \quad \delta_1^{-1}(\hat{B} - \textstyle\sum_{i=1}^{n} 2^{i-1} z_{s_i^*}^{(i)} A) = (s_1 - s^*)\,A + t_1' P$$

$$com = s_2\,A + t_2\,P \quad \delta_2^{-1}(\hat{B} - \textstyle\sum_{i=1}^{n} 2^{i-1} z_{s_i^*}^{(i)} A) = (s_2 - s^*)\,A + t_2' P \ .$$

Now, if $s_1 \neq_p s_2$, the discrete logarithm of A can be extracted as $a := (t_2 - t_1)/(s_1 - s_2)$. Otherwise, let $s := s_1\ (=_p s_2)$, and assume that $s \neq_p s^*$. We have,

$$\delta_1(s - s^*)\,A + \delta_1 t_1' P \ = \ \delta_2(s - s^*)\,A + \delta_2 t_2' P \ ,$$

so we can extract the discrete logarithm of A as

$$a := \frac{\delta_2 t_2' - \delta_1 t_1'}{(\delta_1 - \delta_2)(s - s^*)} \ .$$

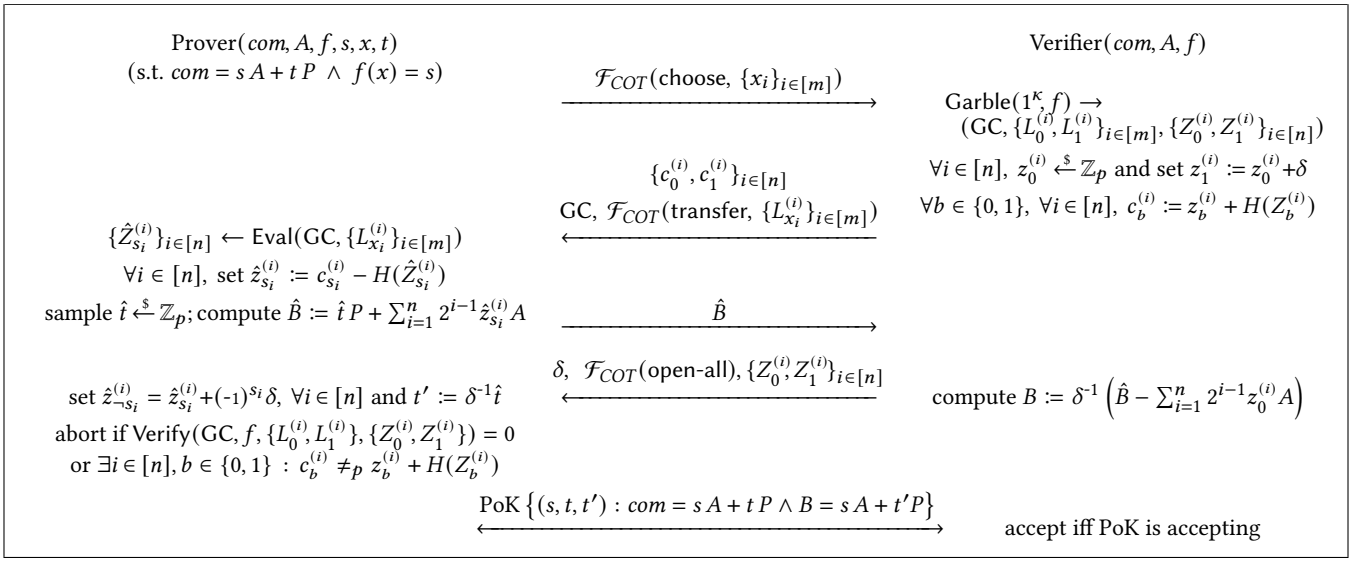

Figure 4: Description of our protocol for hybrid statements for $\text{PoK}\{(s,x,t) : com = s\,A + t\,P \wedge f(x) = s\}$**, where** $A, com \in G$ **for a group (of order** p**), generated by** P**, circuit** $f : \{0,1\}^m \to \{0,1\}^n$**, and function** $H : \{0,1\}^* \to \mathbb{Z}_p$**.**

When τ is polynomial in the security parameter and ρ is non-negligible, our adversary $\mathcal{B}$ runs in expected polynomial time. This completes the proof. □

5 IMPLEMENTATION

We implement a general library for zero-knowledge proofs on arithmetic circuits combined with algebraic statements, based on our protocol from Figure 4. Our library is written in C/C++17 (compiled using the g++ compiler from GCC) and uses the Relic-Toolkit [3] for elliptic curves and the EMP-Toolkit [43] for garbled circuits and committing oblivious transfer. Our code is publicly available and open source for reproducibility and verifiability.[6]

In this section, we present the implementation details and measurements of our library applied to the construction of an information CP system for various purposes. In particular, we explore the application of selling an ECDSA signature or RSA signature on an agreed message. In Sections 5.1 and 5.2 we propose two methods to argue (in zero-knowledge) knowledge of an ECDSA or RSA signature (respectively) by showing the knowledge of a discrete logarithm. Therefore these methods can be instantiated very efficiently using the celebrated Schnorr sigma-protocol [40].

In all cases, we assume the seller (the prover) has encrypted the secret s to be sold under some key k, producing a ciphertext $c \leftarrow \text{Enc}_k(s)$, computed $y := \text{SHA}(k)$ and shared both values (c, y) with the buyer (the verifier). The seller then committed to the secret s as $com = s\,A + t\,P$, for some t chosen uniformly at random, where A is an EC group element with unknown dlog (at least, unknown to the seller, A may be chosen by the verifier).

Our implementation consists of the computation of the previous values and the execution of the protocol from Figure 4, so that the verifier gets convinced about the fact that the value the seller committed to in com is the output of circuit $f_{(c,y)}$ on some input (on k), where f is defined as:

$$f_{(c,y)}(k) := \text{Dec}_k(c) \text{ if } y = \text{SHA}(k) \text{ else } \perp \ .$$

They will then execute an algebraic protocol where the prover convinces the verifier about the fact that com contains a valid (ECDSA or RSA) signature. In the next sections we describe how these algebraic protocols can be implemented as knowledge of a discrete logarithm.

5.1 ECDSA Signatures

First we recall the definition of ECDSA signatures. Key generation selects an elliptic curve of order p with generator P, samples $d \xleftarrow{\$} \mathbb{Z}_p$ and produces $(PK, SK) := (Q, d)$, where $Q = d\,P$.

$\text{Sign}(SK, m)$:
- $k \xleftarrow{\$} \mathbb{Z}_p$
- $K := kP := (r, K_y)$
- $s =_p k^{-1}\,(\text{SHA}(m) + r \cdot d)$
- output (r, s)

$\text{Verify}(PK, m, (r, s))$:
- $K' := s^{-1} \cdot \text{SHA}(m)\,P + s^{-1} \cdot r\,Q$
- accept iff $K'_x = r$

We propose a very efficient algebraic method to prove knowledge of an ECDSA signature. This method is not completely zero-knowledge in the sense that it reveals the first component of the signature r, whereas it is zero-knowledge for the second component. Such a method can still be useful to perform fair-exchange of a ECDSA signature, in scenarios where the signature is freshly generated (in those scenarios, r can be simulated by just taking the x coordinate of elliptic curve point chosen uniformly at random). More concretely, after computing a signature (r, s) on an agreed message m, the prover can share K (sampled during the signature generation) with the verifier and perform the zero-knowledge proof (the prover can use s^{-1} as the witness w):

$$\text{PoK}\{(w) \ : \ K = w\,(\text{SHA}(m)\,P + K_x\,Q)\} \quad . \tag{4}$$

[6] Source code at https://github.com/security-kouza/cont-pay.

Observe that the knowledge of K and such a w implies the knowledge of an ECDSA signature for message m: (K_x, w^{-1}).

5.2 RSA Signatures

Apart from the information CP system to sell ECDSA signatures, we also experiment with the Full Domain Hash RSA signature scheme [10]. We first recall the scheme specifications. On input κ as a security parameter, the key generation chooses two κ-bit primes p, q, computes $N \coloneqq pq$ and selects some $e \in \mathbb{Z}^*_{\varphi(N)}$, where φ represents Euler's totient function. It then computes $d \coloneqq e^{-1} \pmod{\varphi(N)}$, and the public key is defined as $PK = (N, e)$, whereas the secret key is set to $SK = d$. Let $H : \{0,1\}^* \to \mathbb{Z}^*_N$. The signing and verification procedures are as follows.

$\mathsf{Sign}(SK, m)$:	$\mathsf{Verify}(PK, m, \sigma)$:
$h \coloneqq H(m)$	$h \coloneqq H(m)$
$\sigma \coloneqq h^d \pmod N$	accept iff $h =_N \sigma^e$
output σ	

In order to prove in zero-knowledge the knowledge of an RSA signature σ on an agreed message m (let $h = H(m)$), the parties can proceed as follows. The seller will sample $r \xleftarrow{\$} \mathbb{Z}_N$, compute $\alpha \coloneqq h^r \pmod N$ and send $t \coloneqq \sigma\alpha \pmod N$, to the buyer. Since the distribution of α is statistically close to the uniform distribution over the subset of $\mathbb{Z}_N$ generated by h, α acts as a blinding factor on the signature σ (which lives in the same subgroup). The prover now performs a zero-knowledge proof of:

$$\mathrm{PoK}\left\{(r) \ : \ (t^e/h) =_N (h^e)^r\right\} \ . \tag{5}$$

Observe that the knowledge of dlog r of (t^e/h) in base (h^e) implies the knowledge of a valid signature for m. More precisely, one could compute a signature σ as $th^{-r} \pmod N$, since

$$\sigma^e =_N (th^{-r})^e =_N t^e(h^e)^{-r} =_N t^e h/t^e =_N h = H(m) \ .$$

Remark 6. Observe that this protocol can be applied to scenarios where the signed message is known to both parties. In those cases, the relation to be proven is between signature σ and element h. Therefore, the full domain hash or, alternatively, the encoding scheme like PKCS#1 padding or PSS encoding (used to convert a bit-string message into integer h) is not important for the application of this protocol.

5.3 Linking with the CP functionality

Observe that both in the case of ECDSA signatures and RSA signatures, we have reduced the problem of proving knowledge of a signature to the problem of proving knowledge of a discrete logarithm of an element with respect to a public basis. This will make the algebraic link with the CP functionality very simple.

- In the case of ECDSA signatures, the secret to which the prover will commit in *com* is s^{-1} so that the prover can then leverage[7] the sigma-protocol from Figure 5 to prove relation (4), which implies that the prover knows an opening to *com* which leads a valid ECDSA signature for m.

[7] With $C = com$, $A = A$, $B = P$, $C' = K$, $A' = \mathrm{SHA}(m)P + K_x Q$ and $B' = 0_G$.

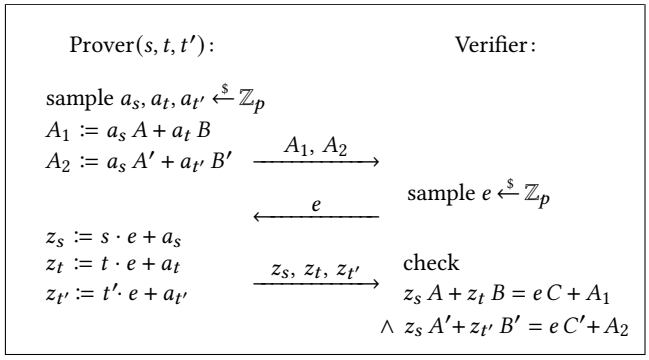

Figure 5: Sigma-protocol for the following relation, where C, C', A, A', B, B' are elements of a group G of order p: $\mathrm{PoK}\{(s,t,t') \ : \ C = sA + tB \ \wedge \ C' = sA' + t'B'\}$.

- In the case of RSA signatures, the secret to which the prover will commit is in *com* is r, the value used for masking the signature as $\sigma H(m)^r \pmod N$. It will then leverage[8] the sigma-protocol from Figure 5 to prove relation (5), which implies that the prover knows an opening to *com* that leads to a valid RSA signature for m.

Remark 7. The protocol from Figure 5 is described using additive notation (for the group law). In the case of RSA (computed over the integers), additions should be interpreted as multiplications, whereas multiplications should be interpreted as exponentiations. Furthermore, in the case of RSA, the exponent of the group is unknown to the verifier (it may also be unknown to the prover) for that, the protocol should be modified, to sample values $a_s, a_t, a_{t'}$ uniformly from $\mathbb{Z}_{N^2}$ and leave values $z_s, z_t, z_{t'}$ as integers (without modular reduction, since the group exponent may be unknown). The challenge e can be sampled from $\mathbb{Z}_N$.

5.4 Results

All experiments were executed on two different machines connected over a LAN of 1Gbps. The verifier ran on a 20-core machine, 3.50GHz Intel-Core i9-9900X CPU and 32 GB of RAM. The prover ran on a 8-core machine, 2.6GHz Intel-Core i7-6700HQ CPU and 16 GB of RAM. Both machines were running Ubuntu 18.04 LTS. Every experiment consists of the following steps:

(a) Both parties parse the circuit implementing f from file and parse a fixed predefined message. Such circuit has been predesigned before the protocol.
(b) The prover computes the corresponding (ECDSA or RSA) signature on the agreed message (the prover has previously generated a key pair and shared the public-key with the verifier).
(c) They perform the *preparation phase*. In the case of an ECDSA signature (r, s), the prover computes s^{-1} (the secret to be encrypted) and shares r with the verifier (see Section 5.1). In the case of an RSA signature σ, the prover samples $r \xleftarrow{\$} \mathbb{Z}_N$ (the secret to be encrypted), masks the signature $t \coloneqq \sigma h^r \pmod N$ and shares t with the verifier (see Section 5.2). The prover also encrypts the corresponding secret under a freshly generated

[8] With $C = com$, $A = A$, $B = P$, $C' = t^e/h$, $A' = h^e$ and $B' = 0_G$.

		Time (milliseconds)					Size				
		Prover		Verifier		Total	Data transmitted		Circuit size (#gates)		
Scheme	🗜	Algebraic	Arithmetic	Algebraic	Arithmetic		$P \to V$	$V \to P$	AND	XOR	INV
ECDSA	×	3.6 ± 0.2	53 ± 2.3	11 ± 4.0	58 ± 2.1	144 ± 3.2	1.14 KB	3.63 MB	23K	92K	2K
	✓		74 ± 2.1		131 ± 2.3	188 ± 3.5		0.81 MB			
RSA (2048)	×	27 ± 0.2	159 ± 3.0	85 ± 4.2	154 ± 3.3	2110 ± 3.3	3.63 KB	20.3 MB	134K	496K	30K
	✓		213 ± 4.1		476 ± 5.5	1010 ± 6.9		4.51 MB			
RSA (4096)	×	161 ± 0.2	230 ± 3.5	329 ± 3.2	190 ± 3.1	3955 ± 4.4	6.63 KB	37.0 MB	244K	900K	57K
	✓		334 ± 2.0		720 ± 5.9	1907 ± 6.4		8.25 MB			

Table 1: Experimental results of protocol applied to CP for selling an ECDSA/RSA signature on an agreed message. We consider the algebraic and arithmetic computation time of both parties as well as the total time for the protocol execution (which includes the data transmission time over the LAN). We present two rows per scheme, representing whether we have used (✓) compression (🗜) of data during the transmission or not (×).

key k, sending both the ciphertext and $\mathrm{SHA}(k)$ to the verifier. Finally, the prover computes a Pedersen-commitment *com* of the secret and sends it to the verifier.

(d) They engage in the protocol described in Figure 4. Then, the prover leverages the sigma-protocol from Figure 5 to build a proof of knowledge of a secret that is both the value committed in *com* and a satisfying solution to equation (4) in the case of ECDSA or to equation (5) in the case of RSA.

(e) The verifier will accept the proof iff both proofs from the previous step are successful.

We have implemented ECDSA signatures over the NIST P256 Curve and the same curve is used for the Pedersen commitments. Furthermore, for RSA, we have run experiments with both 2048 and 4096 bits modulus. In the case of ECDSA signatures, whose size can be encoded in 256-bits (given our choice of parameters), we have used OTP as the encryption scheme (since the key can be hashed with SHA in one block). In the case of RSA, in order to maintain short key size, we have chosen AES (128-bits key) in CBC mode to encrypt the corresponding secret. Observe how the encryption choice affects boolean circuit size (see Table 1).

In order to lift the honest-verifier ZK property of sigma-protocols to full ZK, in our implementation the verifier commits to the challenge before receiving the first message, using a Pedersen-commitment, which is perfectly-hiding. If the prover chooses the discrete logarithm of this Pedersen-commitment's base and reveals this value at the end of the protocol, we have a proof of knowledge.

The experimental results shown in Table 1 correspond to the execution of step (d) from the above list, which constitutes purely the zero-knowledge proof of knowledge for the CP functionality. Therefore, we do not present the times required for generating the signature and preparing (c, y). In all cases, we provide the average time of 20 executions of the protocol (with a 95% confidence interval). We separately present the execution time of both parties with respect to the algebraic and arithmetic operations performed by each of them. We also present details about the amount of data transferred from prover to verifier ($P \to V$) and from verifier to prover ($V \to P$) and the circuit size corresponding to the arithmetic part of the proof. Every experiment is run with and without compression of the transmitted data. Observe that compressing (and decompressing) the data requires time. However, in the case of RSA, compression can be a better choice, given the size of the data being transmitted.

6 CONCLUSION

Given recent arguments on the security of subversion witness-indistinguishable contingent payment protocols, we proved that a presence of a trapdoor in the WI property suffices to formally prove their security. We also presented an efficient interactive zero-knowledge proof system for contingent payment protocol as an alternative to SNARKs. We developed novel techniques to integrate arithmetic (or non-algebraic) and algebraic statements on an interactive zero-knowledge proof system based on garbled circuits. Its performance is demonstrated with a prototype for a quite practical and general scenario of selling ECDSA or RSA signatures. In the case of ECDSA signatures, given less than 150 ms of total execution time, i.e., both seller's and buyer's, time including transmission of less than 5MB of data over the LAN, we conclude that efficient interactive zero-knowledge proofs can be a reasonable alternative to the general approach with SNARKs for scenarios prepared in advance.

ACKNOWLEDGMENTS

We would like to thank Matteo Campanelli and Antonio Faonio for fruitful discussions and their helpful comments about this work; and Antonio Nappa, for his expert advice and help with running our experiments over LAN. We would also like to thank Foteini Baldimtsi and the anonymous reviewers of ACM CCS 2020 for their valuable time and feedback.

REFERENCES

[1] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. 2014. Fair Two-Party Computations via Bitcoin Deposits. In *FC 2014 Workshops (LNCS, Vol. 8438)*, Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith (Eds.). Springer, Heidelberg, 105–121. https://doi.org/10.1007/978-3-662-44774-1_8

[2] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. 2015. On the Malleability of Bitcoin Transactions. In *FC 2015 Workshops (LNCS, Vol. 8976)*, Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff (Eds.). Springer, Heidelberg, 1–18. https://doi.org/10.1007/978-3-662-48051-9_1

[3] D. F. Aranha et al. [n.d.]. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.

[4] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. 2013. More efficient oblivious transfer and extensions for faster secure computation. In *ACM CCS 2013*, Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung (Eds.). ACM Press, 535–548. https://doi.org/10.1145/2508859.2516738

[5] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. 2017. More Efficient Oblivious Transfer Extensions. *Journal of Cryptology* 30, 3 (July 2017), 805–858. https://doi.org/10.1007/s00145-016-9236-6

[6] Waclaw Banasik, Stefan Dziembowski, and Daniel Malinowski. 2016. Efficient Zero-Knowledge Contingent Payments in Cryptocurrencies Without Scripts. In *ESORICS 2016, Part II (LNCS, Vol. 9879)*, Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows (Eds.). Springer, Heidelberg, 261–280. https://doi.org/10.1007/978-3-319-45741-3_14

[7] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. 2016. NIZKs with an Untrusted CRS: Security in the Face of Parameter Subversion. In *ASIACRYPT 2016, Part II (LNCS, Vol. 10032)*, Jung Hee Cheon and Tsuyoshi Takagi (Eds.). Springer, Heidelberg, 777–804. https://doi.org/10.1007/978-3-662-53890-6_26

[8] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. 2012. Foundations of garbled circuits. In *ACM CCS 2012*, Ting Yu, George Danezis, and Virgil D. Gligor (Eds.). ACM Press, 784–796. https://doi.org/10.1145/2382196.2382279

[9] Mihir Bellare and Gregory Neven. 2006. Multi-signatures in the plain public-Key model and a general forking lemma. In *ACM CCS 2006*, Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM Press, 390–399. https://doi.org/10.1145/1180405.1180453

[10] Mihir Bellare and Phillip Rogaway. 1993. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM CCS 93*, Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby (Eds.). ACM Press, 62–73. https://doi.org/10.1145/168588.168596

[11] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. 2015. Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In *2015 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 287–304. https://doi.org/10.1109/SP.2015.25

[12] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. 2014. Scalable Zero Knowledge via Cycles of Elliptic Curves. In *CRYPTO 2014, Part II (LNCS, Vol. 8617)*, Juan A. Garay and Rosario Gennaro (Eds.). Springer, Heidelberg, 276–294. https://doi.org/10.1007/978-3-662-44381-1_16

[13] Manuel Blum, Paul Feldman, and Silvio Micali. 1988. Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract). In *20th ACM STOC*. ACM Press, 103–112. https://doi.org/10.1145/62212.62222

[14] Sean Bowe. 2016. Pay-to-sudoku. https://github.com/zcash-hackworks/pay-to-sudoku.

[15] Sean Bowe, Ariel Gabizon, and Matthew D. Green. 2019. A Multi-party Protocol for Constructing the Public Parameters of the Pinocchio zk-SNARK. In *FC 2018 Workshops (LNCS, Vol. 10958)*, Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala (Eds.). Springer, Heidelberg, 64–77. https://doi.org/10.1007/978-3-662-58820-8_5

[16] Vitalik Buterin. 2013. A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper. *White Paper* 3 (2013). Issue 37.

[17] Jan Camenisch and Markus Stadler. 1997. Proof Systems for General Statements about Discrete Logarithms.

[18] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. 2017. Zero-Knowledge Contingent Payments Revisited: Attacks and Payments for Services. In *ACM CCS 2017*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM Press, 229–243. https://doi.org/10.1145/3133956.3134060

[19] Melissa Chase, Chaya Ganesh, and Payman Mohassel. 2016. Efficient Zero-Knowledge Proof of Algebraic and Non-Algebraic Statements with Applications to Privacy Preserving Credentials. In *CRYPTO 2016, Part III (LNCS, Vol. 9816)*, Matthew Robshaw and Jonathan Katz (Eds.). Springer, Heidelberg, 499–530. https://doi.org/10.1007/978-3-662-53015-3_18

[20] Tung Chou and Claudio Orlandi. 2015. The Simplest Protocol for Oblivious Transfer. In *LATINCRYPT 2015 (LNCS, Vol. 9230)*, Kristin E. Lauter and Francisco Rodríguez-Henríquez (Eds.). Springer, Heidelberg, 40–58. https://doi.org/10.1007/978-3-319-22174-8_3

[21] Richard Cleve. 1986. Limits on the Security of Coin Flips when Half the Processors Are Faulty (Extended Abstract). In *18th ACM STOC*. ACM Press, 364–369. https://doi.org/10.1145/12130.12168

[22] Ronald Cramer. 1997. *Modular Design of Secure yet Practical Cryptographic Protocols*. Ph.D. Dissertation. University of Amsterdam.

[23] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. 2014. Square Span Programs with Applications to Succinct NIZK Arguments. In *ASIACRYPT 2014, Part I (LNCS, Vol. 8873)*, Palash Sarkar and Tetsu Iwata (Eds.). Springer, Heidelberg, 532–550. https://doi.org/10.1007/978-3-662-45611-8_28

[24] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. 2013. An Algebraic Framework for Diffie-Hellman Assumptions. In *CRYPTO 2013, Part II (LNCS, Vol. 8043)*, Ran Canetti and Juan A. Garay (Eds.). Springer, Heidelberg, 129–147. https://doi.org/10.1007/978-3-642-40084-1_8

[25] Uriel Feige and Adi Shamir. 1990. Witness Indistinguishable and Witness Hiding Protocols. In *22nd ACM STOC*. ACM Press, 416–426. https://doi.org/10.1145/100216.100272

[26] Georg Fuchsbauer. 2018. Subversion-Zero-Knowledge SNARKs. In *PKC 2018, Part I (LNCS, Vol. 10769)*, Michel Abdalla and Ricardo Dahab (Eds.). Springer, Heidelberg, 315–347. https://doi.org/10.1007/978-3-319-76578-5_11

[27] Georg Fuchsbauer. 2019. WI Is Not Enough: Zero-Knowledge Contingent (Service) Payments Revisited. In *ACM CCS 2019*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM Press, 49–62. https://doi.org/10.1145/3319535.3354234

[28] Ariel Gabizon. 2019. On the security of the BCTV Pinocchio zk-SNARK variant. Cryptology ePrint Archive, Report 2019/119. https://eprint.iacr.org/2019/119.

[29] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. 2013. Quadratic Span Programs and Succinct NIZKs without PCPs. In *EUROCRYPT 2013 (LNCS, Vol. 7881)*, Thomas Johansson and Phong Q. Nguyen (Eds.). Springer, Heidelberg, 626–645. https://doi.org/10.1007/978-3-642-38348-9_37

[30] Oded Goldreich and Yair Oren. 1994. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology* 7, 1 (Dec. 1994), 1–32. https://doi.org/10.1007/BF00195207

[31] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. 1985. The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract). In *17th ACM STOC*. ACM Press, 291–304. https://doi.org/10.1145/22145.22178

[32] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. 1989. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.* 18, 1 (1989), 186–208.

[33] Jens Groth. 2016. On the Size of Pairing-Based Non-interactive Arguments. In *EUROCRYPT 2016, Part II (LNCS, Vol. 9666)*, Marc Fischlin and Jean-Sébastien Coron (Eds.). Springer, Heidelberg, 305–326. https://doi.org/10.1007/978-3-662-49896-5_11

[34] Jens Groth and Markulf Kohlweiss. 2015. One-Out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin. In *EUROCRYPT 2015, Part II (LNCS, Vol. 9057)*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer, Heidelberg, 253–280. https://doi.org/10.1007/978-3-662-46803-6_9

[35] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. 2013. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *ACM CCS 2013*, Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung (Eds.). ACM Press, 955–966. https://doi.org/10.1145/2508859.2516662

[36] Gregory Maxwell. 2011. Zero Knowledge Contingent Payment. https://en.bitcoin.it/wiki/Zero_Knowledge_Contingent_Payment.

[37] Satoshi Nakamoto. 2009. Bitcoin: A Peer-to-Peer Electronic Cash System. *Cryptography Mailing list at https://metzdowd.com* (03 2009).

[38] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. 2013. Pinocchio: Nearly Practical Verifiable Computation. In *2013 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 238–252. https://doi.org/10.1109/SP.2013.47

[39] David Pointcheval and Jacques Stern. 2000. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology* 13, 3 (June 2000), 361–396. https://doi.org/10.1007/s001450010003

[40] Claus-Peter Schnorr. 1991. Efficient Signature Generation by Smart Cards. *Journal of Cryptology* 4, 3 (Jan. 1991), 161–174. https://doi.org/10.1007/BF00196725

[41] Nick Szabo. 1997. Formalizing and Securing Relationships on Public Networks. https://firstmonday.org/ojs/index.php/fm/article/view/548. *First Monday* 2, 9 (Sep. 1997). https://doi.org/10.5210/fm.v2i9.548

[42] Florian Tramer, Fan Zhang, Huang Lin, Jean-Pierre Hubaux, Ari Juels, and Elaine Shi. 2016. Sealed-Glass Proofs: Using Transparent Enclaves to Prove and Sell Knowledge. Cryptology ePrint Archive, Report 2016/635. http://eprint.iacr.org/2016/635.

[43] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. 2016. EMP-toolkit: Efficient MultiParty computation toolkit. https://github.com/emp-toolkit.

[44] The Bitcoin Wiki. 2019. Hashlock. https://en.bitcoin.it/wiki/Hashlock.

[45] Gavin Wood. 2014. Ethereum: A secure decentralized generalized transaction ledger. https://gavwood.com/paper.pdf.

[46] Samee Zahur, Mike Rosulek, and David Evans. 2015. Two Halves Make a Whole - Reducing Data Transfer in Garbled Circuits Using Half Gates. In *EUROCRYPT 2015, Part II (LNCS, Vol. 9057)*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer, Heidelberg, 220–250. https://doi.org/10.1007/978-3-662-46803-6_8

A ADDITIONAL DEFINITIONS

A.1 Witness indistinguishability

In the security game for standard witness indistinguishability, an adversary $\mathcal{A}$ can adaptively query an oracle that takes a statement x and two witnesses for it, w_0, w_1, and answers back with a proof generated with one of the witnesses (the oracle always uses the first witness or always the second). The adversary has to decide which witness the oracle is using.

Definition A.1 (WI). A NI proof system Π with respect to a relation generator $\mathcal{RG}$ is *witness indistinguishable* if for every ppt adversary $\mathcal{A}$, the following advantage is negligible in κ,

$$\mathsf{Adv}^{\mathrm{WI}}_{\Pi,\mathcal{A}}(\kappa) \coloneqq \left|\Pr\left[\mathrm{WI}^{\Pi,\mathcal{RG}}_{\mathcal{A}}(\kappa)\right] - \frac{1}{2}\right| \ .$$

Subversion witness indistinguishability. Subversion witness indistinguishability [7] requires that the proof system remain WI even when the crs is maliciously generated by the adversary.

Definition A.2 (S-WI). A NI proof system Π with respect to a relation generator $\mathcal{RG}$ is *subversion witness indistinguishable* if for every ppt adversary $\mathcal{A}$, the following advantage is negligible in κ,

$$\mathsf{Adv}^{\text{S-WI}}_{\Pi,\mathcal{A}}(\kappa) \coloneqq \left|\Pr\left[\text{S-WI}^{\Pi,\mathcal{RG}}_{\mathcal{A}}(\kappa)\right] - \frac{1}{2}\right| \ .$$

A.2 Discrete logarithm

Definition A.3 (Dlog). The *discrete logarithm assumption* holds with respect to a group generator $\mathcal{G}$ if for every ppt adversary $\mathcal{A}$, the following advantage is negligible in κ,

$$\mathsf{Adv}^{\mathrm{dlog}}_{\mathcal{G},\mathcal{A}}(\kappa) \coloneqq \Pr\left[\begin{array}{r}(p,G,P) \leftarrow \mathcal{G}(1^\kappa);\\ x \xleftarrow{\$} \mathbb{Z}_p\end{array} \ : \ x \leftarrow \mathcal{A}(p,G,P,xP)\right] \ .$$

A.3 Forking Lemma

This lemma was first introduced by Pointcheval and Stern [39] and later generalized by Bellare and Neven [9].

Lemma A.4 (Forking Lemma). *Let $q \geq 1$ be an integer and let H be a set, with $|H| \geq 2$. Let $\mathcal{A}$ be a randomized algorithm that on input $x, h_1, \ldots, h_q$, returns a pair, where the first element is an integer in $[0, q]$ and the second is called the* side output. *Let $\mathcal{X}$ be a randomized algorithm (called the* input generator*). Let the accepting probability of $\mathcal{A}$, denoted by ρ, be defined as:*

$$\Pr\left[x \leftarrow \mathcal{X};\ h_1,\ldots,h_q \xleftarrow{\$} H;\ (J,\mu) \leftarrow \mathcal{A}(x,h_1,\ldots,h_q) \ :\ J \geq 1\right] \ .$$

Let the forking algorithm $\mathcal{E}_{\mathcal{A}}$ (based on $\mathcal{A}$) be defined as follows:

$\mathcal{E}_{\mathcal{A}}(x)$:
pick coins r for $\mathcal{A}$ at random
sample $h_1,\ldots,h_q \xleftarrow{\$} H$
run $(I,\mu_1) \leftarrow \mathcal{A}(x,h_1,\ldots,h_q;r)$
if $I = 0$, return $(0,\perp)$
sample $\hat{h}_I,\ldots,\hat{h}_q \xleftarrow{\$} H$
run $(I_2,\mu_2) \leftarrow \mathcal{A}(x,h_1,\ldots,h_{I-1},\hat{h}_I,\ldots,\hat{h}_q;r)$
if $I = I_2$ and $h_I \neq \hat{h}_I$:
return $(1,(\mu_1,\mu_2))$
return $(0,\perp)$

It holds:

$$\Pr\left[x \leftarrow \mathcal{X};\ (b,\vec{\mu}) \leftarrow \mathcal{E}_{\mathcal{A}}(x) \ :\ b = 1\right] \geq \frac{\rho^2}{q} - 2\frac{\rho}{|H|} \ .$$

Proof. The proof of this lemma can be done by using the fact that for any real-valued random variable X that only takes positive values, it holds $E[X^2] \geq E[X]^2$. This is a fact that can be derived from Jensen's inequality[9], since function $\varphi(x) \coloneqq x^2$ is convex. We refer to [9] for a complete proof. □

B PROOFS OF THE MAIN BODY

B.1 Proof of Theorem 3.3

In order to prove Theorem 3.3, we will use the following lemma.

Lemma B.1. *Let $\mathcal{E} \coloneqq (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ be an encryption scheme and let $F_\kappa : \{0,1\}^* \to \{0,1\}^\kappa$ be a random oracle for every $\kappa \geq 1$. Let O_0 and O_1 be oracles that take an input in $\{0,1\}^*$ and are defined as:*

$O_0(s)$: $r \leftarrow \mathsf{Gen}(1^\kappa);\ \mathsf{return}\big(\mathsf{Enc}_r(s),\ F_\kappa(r)\big)$
$O_1(s)$: $u \xleftarrow{\$} \{0,1\}^\kappa;\ r \leftarrow \mathsf{Gen}(1^\kappa);\ \mathsf{return}\big(\mathsf{Enc}_r(0^{|s|}),\ u\big)$

Then, for every ppt adversary $\mathcal{A}$ making at most q_O queries to its O oracle and q_F queries to the random oracle, the difference

$$\left|\Pr\left[\mathcal{A}^{O_0(\cdot),F_\kappa(\cdot)}(1^\kappa) = 1\right] - \Pr\left[\mathcal{A}^{O_1(\cdot),F_\kappa(\cdot)}(1^\kappa) = 1\right]\right|$$

is upper-bounded by $2q_O\big(q_\mathsf{F}\mathsf{Adv}^{\text{key-rec}}_{\mathcal{E}}(\kappa) + \mathsf{Adv}^{\mathsf{eav}}_{\mathcal{E}}(\kappa)\big)$.

Proof of Lemma B.1. Fix an arbitrary $s \in \{0,1\}^*$ and a ppt algorithm $\mathcal{A}$ that makes at most q_F queries to the random oracle. Without loss of generality, we assume $\mathcal{A}$ never queries the random oracle on the same input. Let the events:

$$E_{\mathsf{real}}(\kappa) \coloneqq r \leftarrow \mathsf{Gen}(1^\kappa) : \mathcal{A}^{\mathsf{F}_\kappa(\cdot)}\big(\mathsf{Enc}_r(s),\ \mathsf{F}_\kappa(r)\big) = 1$$
$$E_{\mathsf{rand}}(\kappa) \coloneqq u \xleftarrow{\$} \{0,1\}^\kappa;\ r \leftarrow \mathsf{Gen}(1^\kappa) \ : \mathcal{A}^{\mathsf{F}_\kappa(\cdot)}\big(\mathsf{Enc}_r(0^{|s|}),\ u\big) = 1 \ .$$

First, we prove that

$$|\Pr[E_{\mathsf{real}}(\kappa)] - \Pr[E_{\mathsf{rand}}(\kappa)]| \leq 2\big(q_\mathsf{F}\mathsf{Adv}^{\text{key-rec}}_{\mathcal{E}}(\kappa) + \mathsf{Adv}^{\mathsf{eav}}_{\mathcal{E}}(\kappa)\big). \tag{6}$$

To do so, we define the event E_{call} as *"one of the calls by $\mathcal{A}$ to the random oracle F_κ is on input r (the random value sampled in E_{real})"*. We can rewrite $\Pr[E_{\mathsf{real}}(\kappa)]$ as:

$$\Pr[E_{\mathsf{real}}(\kappa) \mid E_{\mathsf{call}}]\Pr[E_{\mathsf{call}}] \ + \ \Pr[E_{\mathsf{real}}(\kappa) \mid \neg E_{\mathsf{call}}]\Pr[\neg E_{\mathsf{call}}] \ .$$

Also, observe that:

- $\Pr[E_{\mathsf{call}}] \ \leq \ q_\mathsf{F}\mathsf{Adv}^{\text{key-rec}}_{\mathcal{E}}(\kappa)$. (7)
- $|\Pr[E_{\mathsf{real}}(\kappa) \mid \neg E_{\mathsf{call}}] - \Pr[E_{\mathsf{rand}}(\kappa)]| \ \leq \ 2\mathsf{Adv}^{\mathsf{eav}}_{\mathcal{E}}(\kappa)$. (8)

To see (7) note that we can build an adversary $\mathcal{B}$ against the key-rec security game. On input κ, $\mathcal{B}$ sends message s to its challenger, receiving $c \coloneqq \mathsf{Enc}_r(s)$ for some unknown and freshly generated key r. Now, $\mathcal{B}$ samples $u \xleftarrow{\$} \{0,1\}^\kappa$ and runs $\mathcal{A}$ on input (c,u). On every query by $\mathcal{A}$ to its random oracle, say on value v_i, $\mathcal{B}$ will answer back with a uniformly sampled value from $\{0,1\}^\kappa$ and will store v_i in a record. When $\mathcal{A}$ interrupts or finishes its execution, $\mathcal{B}$ will output one of the stored values v_i, selected uniformly at

[9] Which states that for any random variable X and any convex function φ, it holds $E[\varphi(X)] \geq \varphi(E[X])$.

random. It is clear that $\mathsf{Adv}^{\text{key-rec}}_{\mathcal{E},\mathcal{B}}(\kappa) \geq \Pr[E_{\text{call}}]/q_{\mathsf{F}}$. To see (8) it is enough to consider an adversary $\mathcal{B}$ against the eav security game that on input κ, sends two messages s and $0^{|s|}$ to its challenger, receiving c which is either $\mathsf{Enc}_r(s)$ or $\mathsf{Enc}_r(0^{|s|})$ for some fresh key r. Then, $\mathcal{B}$ samples $u \xleftarrow{\$} \{0,1\}^\kappa$ and runs $\mathcal{A}$ on input (c,u), simulating the random oracle by sampling values uniformly from $\{0,1\}^\kappa$ and mimics $\mathcal{A}$'s output. It is not hard to see that

$$\mathsf{Adv}^{\text{eav}}_{\mathcal{E},\mathcal{B}}(\kappa) = \frac{1}{2}\left|\Pr\left[E_{\text{real}}(\kappa) \mid \neg E_{\text{call}}\right] - \Pr\left[E_{\text{rand}}(\kappa)\right]\right| \ .$$

Now, to conclude the proof of (6), let us denote $\Pr[E_{\text{call}}]$ by ε for compactness, we have:

$$\begin{aligned}
&\left|\Pr\left[E_{\text{real}}(\kappa)\right] - \Pr\left[E_{\text{rand}}(\kappa)\right]\right| \\
&\quad\leq \varepsilon + \left|\Pr\left[E_{\text{real}}(\kappa) \mid \neg E_{\text{call}}\right](1-\varepsilon) - \Pr\left[E_{\text{rand}}(\kappa)\right]\right| \\
&\quad\leq \varepsilon + 2\mathsf{Adv}^{\text{eav}}_{\mathcal{E}}(\kappa) + \varepsilon\Pr\left[E_{\text{real}}(\kappa) \mid \neg E_{\text{call}}\right] \leq 2\varepsilon + 2\mathsf{Adv}^{\text{eav}}_{\mathcal{E}}(\kappa) \\
&\quad\leq 2\left(q_{\mathsf{F}}\mathsf{Adv}^{\text{key-rec}}_{\mathcal{E}}(\kappa) + \mathsf{Adv}^{\text{eav}}_{\mathcal{E}}(\kappa)\right) \ .
\end{aligned}$$

Finally, to conclude the proof of the lemma, consider an auxiliary oracle O'_i, for every $i \in [0, q_O]$, defined as O_1 on the first i invocations and as O_0 on the rest. Clearly, O'_0 is equivalent to O_0, whereas O'_{q_O} is equivalent to O_1. We conclude the proof by a standard hybrid argument, where we just need to show that for every $i \in [q_O]$ and every ppt adversary $\mathcal{A}$ making at most q_{F} queries to its random oracle, the difference

$$\left|\Pr\left[\mathcal{A}^{O'_{i-1}(\cdot),\mathsf{F}_\kappa(\cdot)}(\kappa) = 1\right] - \Pr\left[\mathcal{A}^{O'_i(\cdot),\mathsf{F}_\kappa(\cdot)}(\kappa) = 1\right]\right|$$

is upper-bounded by $2\left(q_{\mathsf{F}}\mathsf{Adv}^{\text{key-rec}}_{\mathcal{E}}(\kappa) + \mathsf{Adv}^{\text{eav}}_{\mathcal{E}}(\kappa)\right)$, but this follows easily from (6), since it holds for every $s \in \{0,1\}^*$ and given the independence of the sampling on every oracle call. □

Proof of Theorem 3.3. Consider an auxiliary security game, InfoCP′, which is identical to the one described in Figure 2 with the exception that the adversary $\mathcal{A}$ is given access to a modified oracle $O'(R, \mathsf{crs}, b, \tau_k, \cdot)$ defined as:

$O'(R, \mathsf{crs}, b, \tau_k, s)$:
  if $f(s) = 0$, return $\perp$
  $r \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa)$; $c := \mathsf{Enc}_r(s)$; $y := \mathsf{F}_\kappa(r)$
  if $b = 1$: overwrite r with $\mathsf{H}^{-1}_\kappa(\tau_k, y)$
  $\pi \leftarrow \Pi.\mathsf{Prove}(R, \mathsf{crs}, (c,y), r)$
  return (c, y, π)

Let $\mathcal{A}$ be an adversary against the InfoCP game that makes at most q_O queries to its oracle and at most q_{F} queries to the random oracle. For compactness, denote by $W_{\mathcal{A}}$ and $W'_{\mathcal{A}}$ the events $\text{InfoCP}^{\Pi}_{\mathcal{A}}(\kappa)$ and $\text{InfoCP}'^{\Pi}_{\mathcal{A}}(\kappa)$ respectively. We have,

$$\begin{aligned}
\mathsf{Adv}^{\text{InfoCP}}_{\Pi,\mathcal{A}}(\kappa) &:= \left|\Pr[W_{\mathcal{A}}] - \frac{1}{2}\right| \\
&\leq \left|\Pr[W_{\mathcal{A}}] - \Pr[W'_{\mathcal{A}}]\right| + \left|\Pr[W'_{\mathcal{A}}] - \frac{1}{2}\right| \\
&= \left|\Pr[W_{\mathcal{A}}] - \Pr[W'_{\mathcal{A}}]\right| + \mathsf{Adv}^{\text{InfoCP}'}_{\Pi,\mathcal{A}}(\kappa) \qquad (9)
\end{aligned}$$

Now, consider the following distinguisher $\mathcal{D}$ with access to oracle O_b of Lemma B.1 (for some $b \in \{0,1\}$) and the random oracle F_κ:

$\mathcal{D}^{O_b(\cdot),\mathsf{F}_\kappa(\cdot)}(1^\kappa)$:
  run $\mathcal{RG}(1^\kappa)$ *to get* (R, τ_k)
  call $\mathcal{A}$ *on input* R, *getting* (crs, σ)
  call $\mathcal{A}$ *on input* σ
  on every query (s) *from* $\mathcal{A}$ *to its oracle:*
    query (s) *to the* O_b *oracle to get* (c,y)
    compute $\pi \leftarrow \Pi.\mathsf{Prove}(R, \mathsf{crs}, (c,y), \mathsf{H}^{-1}_\kappa(\tau_k, y))$
    send π *to* $\mathcal{A}$
  on every call by $\mathcal{A}$ *to the random oracle:*
    $\mathcal{D}$ *forwards the query to its own random oracle and copies the output to* $\mathcal{A}$
  when $\mathcal{A}$ *outputs* $b'_{\mathcal{A}}$, *return* $b'_{\mathcal{A}}$

It is clear that $\mathcal{D}$ performs at most q_{F} queries to its random oracle and at most q_O queries its O_b oracle. Moreover, if $b = 0$, the distinguisher $\mathcal{D}$ is perfectly simulating to $\mathcal{A}$ the InfoCP′ game, where the internal challenge bit, $b_{\text{InfoCP}'}$, is 1. Furthermore, if $b = 1$, $\mathcal{D}$ is simulating to $\mathcal{A}$ the InfoCP game, where the internal challenge bit, b_{InfoCP}, is 1. This is because the following distributions are identical, given that $\mathsf{H}_\kappa(k,\cdot)$ is a permutation:

$$\left\{w \xleftarrow{\$} \{0,1\}^\kappa;\ \left(\mathsf{H}_\kappa(k,w),\ w\right)\right\} \equiv \left\{u \xleftarrow{\$} \{0,1\}^\kappa;\ \left(u,\ \mathsf{H}^{-1}_\kappa(k,u)\right)\right\} \ .$$

We therefore have,

$$\begin{aligned}
\Pr\left[\mathcal{D}^{O_0(\cdot),F_\kappa(\cdot)}(\kappa) = 1\right] &= \Pr\left[W'_{\mathcal{A}} \mid b_{\text{InfoCP}'} = 1\right] \text{ and} \\
\Pr\left[\mathcal{D}^{O_1(\cdot),F_\kappa(\cdot)}(\kappa) = 1\right] &= \Pr\left[W_{\mathcal{A}} \mid b_{\text{InfoCP}} = 1\right] \ .
\end{aligned}$$

Moreover, observe that

$$\Pr\left[W_{\mathcal{A}} \mid b_{\text{InfoCP}} = 0\right] = \Pr\left[W'_{\mathcal{A}} \mid b_{\text{InfoCP}'} = 0\right] \ ,$$

since both experiments are identical when the challenge bit is 0. Consequently, and in virtue of Lemma B.1,

$$\begin{aligned}
&\left|\Pr[W_{\mathcal{A}}] - \Pr[W'_{\mathcal{A}}]\right| \\
&= \frac{1}{2}\left|\Pr\left[W_{\mathcal{A}} \mid b_{\text{InfoCP}} = 1\right] - \Pr\left[W'_{\mathcal{A}} \mid b_{\text{InfoCP}'} = 1\right]\right| \\
&= \frac{1}{2}\left|\Pr\left[\mathcal{D}^{O_1(\cdot),F_\kappa(\cdot)}(\kappa) = 1\right] - \Pr\left[\mathcal{D}^{O_0(\cdot),F_\kappa(\cdot)}(\kappa) = 1\right]\right| \\
&\leq q_O\left(q_{\mathsf{F}}\mathsf{Adv}^{\text{key-rec}}_{\mathcal{E}}(\kappa) + \mathsf{Adv}^{\text{eav}}_{\mathcal{E}}(\kappa)\right) \ .
\end{aligned}$$

Combining this fact with equation (9), we can conclude the proof of the theorem by showing that

$$\mathsf{Adv}^{\text{InfoCP}'}_{\Pi,\mathcal{A}}(\kappa) \leq \mathsf{Adv}^{\text{tS-WI}}_{\Pi,\mathcal{B}}(\kappa)$$

for some adversary $\mathcal{B}$ against the tS-WI game. We provide such a $\mathcal{B}$ in Figure 6. It is clear that the distribution of $\mathcal{B}$'s answers to $\mathcal{A}$'s queries is the same as the one by the oracle in Game InfoCP′. Therefore, if $\mathcal{A}$ outputs the correct result, so does $\mathcal{B}$. We actually have $\mathsf{Adv}^{\text{InfoCP}'}_{\Pi,\mathcal{A}}(\kappa) = \mathsf{Adv}^{\text{tS-WI}}_{\Pi,\mathcal{B}}(\kappa)$. □

B.2 Proof of the sigma-protocol from Figure 5

Proof. It is not hard to see that the protocol from Figure 5 is perfectly complete. Furthermore, it satisfies 2-special soundness. Indeed, from two different accepting transcripts with the same

$\mathcal{B}(R, \tau_k)$:
call $\mathcal{A}$ on input R, getting (crs, σ) and let $\hat{\sigma} := (\sigma, \tau_k)$
output $(\text{crs}, \hat{\sigma})$ to the tS-WI challenger

$\mathcal{B}^{\mathcal{O}(R,\text{crs},b,\cdot,\cdot,\cdot)}(\hat{\sigma})$:
parse $\hat{\sigma}$ as (σ, τ_k) and run $\mathcal{A}$ on input σ
on every query (s) by $\mathcal{A}$ to its oracle:
  if $f(s) = 0$, return $\perp$
  sample $r_0 \xleftarrow{\$} \{0,1\}^\kappa$
  set $c := \text{Enc}_{r_0}(s)$; $y := \mathsf{F}_\kappa(r_0)$; set $r_1 := \mathsf{H}_\kappa^{-1}(\tau_k, y)$
  query $((c, y), r_0, r_1)$ to the tS-WI oracle, to get π
  send (c, y, π) to $\mathcal{A}$
when $\mathcal{A}$ outputs $b'_{\mathcal{A}}$, return $b' := b'_{\mathcal{A}}$

Figure 6: Adversary $\mathcal{B}$ against the tS-WI game, based on adversary $\mathcal{A}$ against the InfoCP′ game.

first message, but different challenges, say $(A_1, A_2, e, z_s, z_t, z_{t'})$ and $(A_1, A_2, \hat{e}, \hat{z}_s, \hat{z}_t, \hat{z}_{t'})$, where $e \neq_p \hat{e}$, let:

$$s := (z_s - \hat{z}_s)/(e - \hat{e}) \quad t := (z_t - \hat{z}_t)/(e - \hat{e}) \quad t' := (z_{t'} - \hat{z}_{t'})/(e - \hat{e}) \ .$$

It turns out that (s, t, t') is a valid witness. Finally, the protocol is honest-verifier zero-knowledge by a simulator that outputs an accepting transcript (A_1, A_2, c, z_s, z_t), identically distributed to a real one, as follows: on input $e \in \mathbb{Z}_p$, sample $z_s, z_t, z_{t'} \xleftarrow{\$} \mathbb{Z}_p$ and compute $A_1 := z_s\, A + z_t\, B - e\, C$ and $A_2 := z_s\, A' + z_{t'}\, B' - e\, C'$. □