AUGMENTED EFFICIENT ZERO-KNOWLEDGE CONTINGENT PAYMENTS IN CRYPTOCURRENCIES WITHOUT SCRIPTS

Peifang Ni[1, 2]

[1]TCA Laboratory, Institute of Software,
Chinese Academy of Sciences, Beijing, China
[2]State Key Laboratory of Cryptology, Beijing, China

ABSTRACT

Zero-Knowledge Contingent Payment presents how Bitcoin contracts can provide a solution for the so-called fair exchange problem.Banasik, W. et al. first presented an efficient Zero-Knowledge Contingent Payment protocol for a large class of NP-relations, which is a protocol for selling witness. It obtains fairness in the following sense: if the seller aborts the protocol without broadcasting the final message then the buyer finally gets his payment back. However, we find that the seller in the protocol could refuse to broadcast the final signature of the transaction without any compensation for the buyer. As a result, the buyer cannot get the witness from the final signature of the transaction and has the payment for the witness locked until finishing the large computation for a secret signing key.

In this paper, we fix this problem by augmenting the efficient Zero-Knowledge Contingent Payment protocol. We present a new protocol where the seller needs to provide the deposit before the zero-knowledge proof of knowledge of the witness being sold. And then the buyer could obtain the seller's witness if the seller broadcasts the final signature of the transaction and gets the payment and his deposit. Otherwise, the buyer could get back the payment and obtain the seller's deposit. This new augmented protocol is constructed without any new assumptions.

KEYWORDS

fair exchange, Bitcoin, cryptocurrencies, zero-knowledge, without scripts.

1. INTRODUCTION

The concept of cryptocurrency emerged in the last few years and recently there has been a huge emphasis on constructing cryptocurrencies easy for circulation. The main valuable property of these cryptocurrencies is that their security does not need to rely on any single trusted third party. Bitcoin [1], the most prominent of the cryptocurrencies, is a decentralized payment system that is based on maintaining a public transaction ledger in a distributed manner. The list of transactions in this payment system is written on a public ledger, which is maintained jointly by the system users. With the public ledger, the system can implement an idea of the so-called smart-contracts. Consider the Zero-Knowledge Contingent Payment [2], which is a contract and shows that how Bitcoin contracts can provide a solution for the so-called fair exchange problem. With respect to [2], the Zero Knowledge Contingent Payment makes it possible to make payments using Bitcoin in a trustless manner where neither the payer or payee can cheat and that the payments are given

David C. Wyld et al. (Eds): AIFZ, SIGPRO, NWCOM, CLSB, MLNLP, DTMN, BDIOT, CSITY - 2022
pp. 53-67, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121605

to the payee only in the case that some knowledge is disclosed by the payee. The execution of this contract is guaranteed by the rules of the underlying Bitcoin system.

To be more specific, it is executed between two parties that do not trust each other: the Seller and the Buyer. The Buyer is looking for some value $x \in \{0,1\}^*$ and the valuable conditions of x for him can be described as a function $f:\{0,1\}^* \to \{true, false\}$ (in a form of a polynomial-time computer program), such that finding x satisfying $f(x) = true$ is much harder than verifying that $f(x) = true$ holds. Hence than, the Buyer is willing to pay for x in the conditions that x is valuable for him. Imagine now that the Buyer is approached by a Seller through the internet, who is claiming that he knows x satisfying the valuable condition $f(x) = true$ and willing to sell x. Then the parties face the following problem: they need to finish the transaction on the internet in a trustless manner where neither the Seller or the Buyer can cheat.

The Zero-Knowledge Contingent Payment protocol described in [2] is accomplished using the combination of a hash-locked transaction and some non-standard scripts so that the data revealed in the hashlock release is the data in need. Recently, It has been implemented in [3] for selling a proof of a sudoku solution.

Banasik, W. et al. first created non-trivial efficient smart contracts using only the standard transactions in the public ledger [4]. Under the assumption of semantically secure Paillier encryption and symmetric encryption, secure commitment and time-locked commitment schemes, the strongly unforgeable ECDSA scheme, and zero-knowledge proof of knowledge, they constructed efficient Zero-Knowledge Contingent Payment protocol for a large class of NP-relations, which is a protocol for selling the witness. In the protocol, the buyer first prepares a transaction T_1 sending the funds from his public key to a public key shared by the buyer and the seller, and then they sign the transaction T_2, sending the funds from the shared public key to the seller's public key, using secret-shared signing keys in cooperation with the seller knowing the final signature of the transaction but the buyer not. After the seller giving a zero-knowledge proof of the witness being sold, the buyer broadcasts T_1 and the seller broadcasts T_2. And then the buyer can reverse the witness from the final signature. However, we find that the seller in the protocol could refuse to broadcast the final signature of the transaction without any compensation for the buyer. As a result, the buyer cannot get the witness from the final signature of the transaction and has the payment for the witness locked until finishing the large computation for a secret signing key.

1.1. Our Results

In this paper, we fix this problem by augmenting the efficient Zero-Knowledge Contingent Payment protocol. We present a new protocol where the seller needs to provide the deposit before the zero-knowledge proof of knowledge of the witness being sold. And then the buyer could obtain the seller's witness if the seller broadcasts the final signature of the transaction and gets the payment and his deposit. Otherwise, the buyer could get back the payment and obtain the seller's deposit. This new augmented protocol is constructed without any new assumptions. A high-level overview is as the following.

The ECDSA signature scheme is denoted by $(ECGen, ECSign, ECVer)$. A user in Bitcoin system is identified by his public key pk in the ECDSA signature scheme and each key pk is called an address. A simple transaction denoted by $[T]$ simply sends some Bitcoins x from address pk_0 to pk_1. $[T]$ contains the following tuple $[T] \coloneqq (TXid(T'), value: x, from: pk_0, to: pk_1)$, where $TXid(T')$ denotes the identifier of transaction$[T']$ with value at least x that appeared earlier on the ledger and is redeemed by $[T]$.

A complete transaction denoted by T has a form $([T], ECSign_{sk_0}([T]))$. $TXid(T')$ is defined simply as a $SHA256$ hash of $([T], ECSign_{sk_0}([T]))$.

Under the assumption of semantically secure Paillier encryption and symmetric encryption, secure commitment and time-locked commitment schemes, the strongly unforgeable ECDSA scheme, and zero knowledge proof, our augmented efficient Zero-Knowledge Contingent Payment protocol consists of four stages.

In stage 1, the buyer and seller execute a key exchange protocol to generate two key pairs for the ECDSA signatures such that the secret keys are secret-shared between them. As a result, the buyer holds (PK, SK_0) , (PK', SK'_0) and seller holds $(PK, SK_1), (PK', SK'_1)$.

In stage 2, the seller prepares transaction T_3 sending the funds p from PK_S to PK' and T'_3sending the funds p from PK' to PK_S, and then sends the hash value of T_3 to the buyer. The buyer prepares transaction T_1 sending the funds v from PK_B to PK', T'_1 sending the funds $v + p$ from PK' to PK, and T_2 sending the funds $v + p$ from PK to the seller's public key PK_S. They execute the unique signature generation protocol to generate the signature of $[T'_1]$ for the buyer and the signatures of T_2 and $[T'_3]$ for the seller. And then the seller broadcasts T_3. If T_3 is not corresponding to the hash value of sent before, the buyer aborts. Otherwise, the parties execute the following stages.

In stage 3, the seller proves the knowledge of the witness being sold by executing a zero-knowledge proof of knowledge protocol with the buyer in the cut-and-choose technique. In the proof, the seller uses the signature of $[T_2]$ to compute the secret key for encryption of a set of challenges and responses with the witness, and then it sends the commitments of the encryption to the buyer. After receiving the subset of the challenges from the buyer, the seller opens the commitments asked to open. The proof is valid if the output of the buyer's verification is true. Otherwise, the buyer aborts.

In stage 4, if the buyer broadcasts the valid T_1 and T'_1, and the seller broadcasts the valid T_2, the buyer could reverse the witness from the signature of T_2. If the buyer refuses to broadcast T_1 and T'_1 or he broadcasts illegal T_1 and T'_1, then the seller could broadcast T'_3 to get the deposit back. If the buyer broadcasts the valid T_1 and T'_1, but the seller refuses to broadcast T_2, the buyer could finally obtain his own funds together with the seller's deposit after finishing a large computation for a secret signing key.

Therefore, it holds that:

assume the existence of semantically secure Paillier encryption and symmetric encryption, secure commitment and time-locked commitment schemes, the strongly unforgeable ECDSA scheme, and zero knowledge proof of knowledge. Then, there exists a secure efficient Zero-Knowledge Contingent Payment protocol in cryptocurrencies without scripts, which obtains fairness in the following sense: if the seller aborts protocol without broadcasting the final message then the buyer finally gets its payment back and gets an extra financial compensation from the seller.

1.2. Related Works

Relevant to our work are the works on smart contracts that provide solutions for fair protocols in the cryptocurrency systems. Bentov, I. and Kumaresan, R. studied secure computations in the following model of fairness: a malicious user who aborts protocol after receiving the output is forced to pay a mutually predefined monetary penalty [5]. They then showed how to use Bitcoin

system to achieve secure computations with the above defined fairness in two-party setting as well as the multiparty setting (with a honest majority) by simulating with a new ideal functionalities as they proposed.

Andrychowicz, M. et al. showed how to obtain fair two-party secure computation protocol via the Bitcoin system with the fairness in the following sense: if one party aborts the protocol after learning the output but the other one not, then the other party gets a financial compensation from the aborted one [6]. They constructed the protocol with the two-party protocol of Goldreich and Vainish [7] additionally secured against an active adversary with zero-knowledge proofs. And they presented one possible application of the protocol to the fair contract signing: each party is forced to either complete the protocol or pay a fine to the other party.

1.3. Outline

In Section 2, we define the notations and definitions that are used through the paper. In Section 3, we describe the subprotocols used in the Banasik's efficient Zero-Knowledge Contingent Payment and give the new subprotocols used in our protocol. In Section 4, we present our augmented efficient Zero-Knowledge Contingent Payment protocol. And Section 5 is conclusion.

2. PRELIMINARIES

2.1. Notations

We use n to denote the security parameter. We use $[k]$ for any $k \in N$ to denote the set $\{1, \dots, k\}$. For any probabilistic algorithm $A(\cdot)$, $A(x)$ is the result of executing A with input x and uniformly chosen randomness. We use $y = A(x)$ to denote that y is set to $A(x)$. For a set S, we use $y \in S$ (or $y \leftarrow S$) to denote that y is chosen from S. For any language L) and instance$x \in L$), we use $\{\mathcal{R}_L\}(x)$) to denote the set of witnesses for $x \in L$. A function $\mu: N \to R$ is negligible (in x) if for every positive integer c there exists an integer N_c such that for all x>N_c we have that $|\mu(x)| < \frac{1}{x_c}$.

2.2. Elliptic Curve Digital Signature Algorithm (ECDSA)

We recall the definition of digital signature algorithm [8, 4, 9].

Definition 1 (Digital Signature Scheme): A digital signature scheme consists of a triple of probabilistic polynomial-time algorithms $(Gen, Sign, Ver)$ satisfying the following conditions:

- Key-generator algorithm Gen: On input 1^n, Gen outputs a pair of keys(sk, vk);
- Signing algorithm $Sign$: On input secret key sk and a message α, $Sign$ outputs signatureσ;
- (Deterministic) Verifying algorithm Ver: On input public key vk, message α, and signature, σ, Ver returns $Output \in \{\text{accept}, \text{reject}\}$;
- For every pair (sk, vk) in the range of $Gen(1^n)$ and for every $\alpha \in \{0,1\}^*$, the signing and verification algorithms $Sign$ and Ver satisfy

$$\Pr\left[Ver\left(vk, \alpha, Sign(sk, \alpha)\right) = 1\right] = 1$$

where the probability is taken over the internal coin tosses of algorithms $Sign$ and Ver.

Definition 2 (Elliptic Curve Digital Signature Algorithm) The ECDSA signature scheme $(ECGen, ECSign, ECVer)$ is a variant of digital signature scheme algorithm, in which the algorithms $ECGen, ECSign, ECVer$ are defined as the followings.

- Key-generator algorithm $ECGen$: On input 1^n, $ECGen$ chooses an elliptic curve group $(G, O, g, +)$ over a prime field Z_p, where O is the neutral element, g is the generator of G, and the order of G is a prime number such that $\lceil log_2|G| \rceil = n$. And then $ECGen$ samples a random $d \in_R Z_{|G|}$, and computes $D \coloneqq d \cdot g$. The generated secret key is $(d, (G, O, g, +))$, and the public key is $(D, (G, O, g, +))$;
- Signing algorithm $ECSign$: Let H be a hash function and $f: G \to Z_{|G|}$ be a reduction function that we will define in a moment. On input secret key and messageα, $ECSign$ first chooses a random $k \in_R Z_{|G|}$, and then computes $r = f(k \cdot g)$ and $s = k^{-1}(H(\alpha) + d \cdot r) mod |G|$. If $r = 0$ or $s = 0$ then the algorithm aborts. Otherwise, the $ECSign$ outputs (r, s);
- (Deterministic) Verifying algorithm $ECVer$: On input public key, message α and signature (r, s), $ECVer$ first checks if r and s are non zero elements of $Z_{|G|}$ and then verifies if $r = f(H(\alpha) \cdot s^{-1} \cdot g + r \cdot s^{-1} \cdot D)$. If this holds, then $ECVer$ outputs ok, and otherwise it outputs$\perp$.

Recall that every element of G has the form $(x, y) \in Z_p^2$. The reduction function $f: G \to Z_{|G|}$ is defined as: on input (x, y) ignores y and produces as output $f(x, y) = x mod \ |G|$.

Security of the ECDSA signature scheme. The ECDSA signature scheme satisfies existentially strong unforgeability under a chosen message attack. We consider the following game played by a polynomial time adversary$A$. First, a key pair is sampled as $(vk, sk) \coloneqq ECGen(1^n)$ and the adversary A is given the public verification key vk. Then, A chooses a sequence of messages $\alpha_1, \dots, \alpha_k$ and learns each corresponding signature $\sigma_i \coloneqq ECSign_{sk}(\alpha)$. He does it in an adaptive way that he chooses each α_i after learning $\sigma_1, \dots, \sigma_{i-1}$. Finally, A outputs a pair $(\alpha_{k+1}, \sigma_{k+1})$. We say that A mauls a signature if the output $(\alpha_{k+1}, \sigma_{k+1})$ satisfies that $ECVer(\alpha_{k+1}, \sigma_{k+1}) = ok$ and σ_{k+1} has not been sent to A as one of the signatures in $\sigma_1, \dots, \sigma_k$. The ECDSA signature scheme is existentially strongly unforgeable under a chosen message attack if every polynomial-time adversary can maul a signature with at most negligible probability.

2.3. Homomorphic Encryption Schemes

We recall the definitions of encryption schemes in [4, 10, 11] and present the following definition.

Definition 3 (Public Encryption Scheme) A public encryption scheme consists of a triple of probabilistic polynomial-time algorithms $(EncGen, Enc, Dec)$:

- Key-generator algorithm $EncGen$: On input 1^n, $EncGen$ outputs a pair of keys (pk, sk);
- Encryption algorithm Enc: On input public key pk and message $\alpha \in \{0,1\}^*$, Enc outputs ciphertext $c = Enc_{pk}(\alpha) \in \{0,1\}^*$;
- Decryption algorithm Dec: On input private key sk, ciphertext c, Dec outputs $\alpha' = Dec_{sk}(c)$;
- For every pair (pk, sk) in the range of $EncGen(1^n)$ and for every $\alpha \in \{0,1\}^*$, the encryption and decryption algorithms Enc and Dec satisfy

$$\Pr\left[Dec_{sk}\left(Enc_{pk}(\alpha)\right) = \alpha\right] = 1$$

where the probability is taken over the internal coin tosses of algorithms Enc and Dec.

Security. To define security of an encryption scheme $(EncGen, Enc, Dec)$ consider a polynomial time adversary A produces a pair of messages (m_0, m_1) with access to oracle $Enc_{pk}(\cdot)$ and $Dec_{sk}(\cdot)$, where $(pk, sk) = EncGen(1^n)$. He then receives a ciphertext $c = Enc_{pk}(m_b)$ for a random $b \in \{0,1\}$, and produces $b' \in \{0,1\}$ without asking the oracle $Dec_{sk}(\cdot)$ to decrypt c. We say that A wins if $b' = b$. We say that the encryption scheme $(EncGen, Enc, Dec)$ is semantically secure if for every polynomial time adversary A the probability that he wins is at most $\frac{1}{2} + \mu(n)$, where μ is some negligible function (in other words: $Enc_{pk}(m_0)$ and $Enc_{pk}(m_1)$ are computationally indistinguishable).

The definition of a symmetric-key encryption scheme is the same as that of a public encryption scheme except that there is only one secret key k, which is usually sampled uniformly at random from some space $\mathcal{K}$ (that depends on 1^n). The adversary does not learn the key, but can try to get some partial information about k by choosing messages $m_1, \dots, m_n$ and learning the corresponding ciphertexts $Enc_{pk}(m_1), \dots, Enc_{pk}(m_n)$ in an adaptive way.

A public-key encryption scheme $(EncGen, Enc, Dec)$ is additively homomorphic if the set of valid messages for the public key pk is an additive group $(H_{pk}, +)$, where the key pair (pk, sk) is generated by $EncGen$. Moreover, the homomorphic algorithm we require is defined by an operation $\otimes: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \{\perp\}$, such that for every valid key pair (pk, sk) and every pair of messages $(m_0, m_1) \in H_{pk}$ we have that $Dec_{sk}\left(Enc_{pk}(m_0) \otimes Enc_{pk}(m_1)\right) = m_0 + m_1$.

2.4. Time-lock Commitment Schemes

We recall the definitions of time-lock commitment schemes [4, 12, 13]. A commitment scheme $(Commit, Open)$ executed between two parties (a committer and a receiver) consists of two phases, the commit phase and the open phase.

- Commit Phase. The committer takes as input a message $\alpha \in \{0,1\}^*$ and chooses a random $r \in \{0,1\}^*$. The committer then computes $c \coloneqq Commit(\alpha)$ with randomness r and sends it to the receiver;
- Open Phase. The committer reveals the message α committed and the randomness r used in the commit phase. And the receiver verifies $Open(c, \alpha, r)$.

Security. A commitment scheme is secure if it is binding and hiding. The hiding property means that for every $\alpha_0, \alpha_1 \in \{0,1\}^*$, $Commit(\alpha_0)$ and $Commit(\alpha_1)$ are computationally indistinguishable. The binding property means that for any PPT committer, he can reveal another message $\alpha' \neq \alpha$ such that $Open(c, \alpha, r) = \alpha'$ with negligible probability.

A commitment scheme $(Commit, Open)$ is a time-lock commitment scheme if the receiver can open the commitment by himself with a significant computational effort. Every time-lock commitment comes with two parameters, τ_0 and τ_1 (with $\tau_0 \leq \tau_1$), and is called (τ_0, τ_1)-secure, where τ_0 denotes the time that everybody, including very powerful adversaries, needs to force open the commitment, and τ_1 denotes the time needed by the honest users to force open the commitment.

2.5. Zero-knowledge Proofs of Knowledge

We recall the definitions of zero-knowledge protocols [4, 14, 15, 16] and give a simple introduction.

Definition 4 (Interactive Proof System) A pair of interactive Turing machines $\langle P, V\rangle$ is called an interactive proof system for any language L if machine V is polynomial-time and the following two conditions hold:

- Completeness: There exists a negligible function c such that for every $x \in L$,

$$\Pr[\langle P, V\rangle(x) = 1] > 1 - c(|x|)$$

- Soundness: There exists a negligible function s such that for every $x \notin L$ and every interactive machine B, it holds that

$$\Pr[\langle P, V\rangle(x) = 1] < s(|x|)$$

$c(\cdot)$ is called the completeness error and $s(\cdot)$ is the soundness error.

Zero-knowledge protocols are interactive proof systems with zero-knowledge property, which means that the prover knows a witness ω for some instance $x \in L$ and convinces the verifier that $x \in L$ without providing the verifier with any additional information beyond the fact that $x \in L$.

A zero-knowledge protocol is called a zero-knowledge proof of knowledge if $L \in NP$ and for every prover P^* there exists a polynomial-time knowledge extractor, that can output a witness ω for $x \in L$ by interacting with P^*. We follow the requirement in [4], suppose that the last two messages in the zero-knowledge proof of knowledge protocol are: a challenge c sent by the verifier to prover, and the prover's response r corresponding to the challenge c. The knowledge extractor extracts witness ω after being given transcripts of two accepting executions that are identical except that the challenges are different (and the responses may also be different).

2.6. Bitcoin Transaction Syntax

A complete transaction denoted by T has a form $([T], ECSign_{sk_0}([T]))$, where $[T] \coloneqq (TXid(T'), value{:}\, x, from{:}\, pk_0, to{:}\, pk_1)$. Another standard type of the transaction is called multisig transaction [4], where $[T]$ has a form $(TXid(T'), value{:}\, x, from{:}\, pk_0, to\ \ k - out - of m{:}\ \ pk_1, \dots, pk_m)$ and it is signed by sk_0. It can be redeemed by a transaction T'' with the form $([T''], \sigma_{i_1}, \dots, \sigma_{i_k})$, $1 \le i_1 < \cdots < i_k \le m$ and for every $1 \le j \le k$ it holds that $ECVer_{pk_{i_j}}\left([T''], \sigma_{i_j}\right) = ok$.

3. The Subprotocols in the Zero-Knowledge Contingent Payment Protocol

In this section, we describe the subprotocols used in Banasik's efficient Zero-Knowledge Contingent Payment system and give the new formalization used in our protocol.

3.1. The Two-party ECDSA Key Generation Protocol

The first ingredient is a protocol called $SharedKGen$, in which two parties, the Seller and Buyer, generate a key pair (public key, private key) for ECDSA, in such a way that the secret key is secret-shared between the Seller and Buyer. This protocol is still used in our final protocol.

To be more precise, fix an elliptic curve $(G, Og, +)$ constructed over a prime field Z_p and let $Com = (Commit, Open)$ be a secure (computational hiding) commitment scheme, both parties take as input a security parameter 1^n, the overview of $SharedKGen$ is as following:

- Seller: samples $d_S \leftarrow Z^*_{|G|}$, computes $D_S := d_S \cdot g$ and $c := Commit(D_S)$, and sends c to the Buyer
- Buyer: samples $d_B \leftarrow Z^*_{|G|}$, computes $D_B := d_B \cdot g$ and sends D_B to the Seller;
- Seller: sends $Open(D_S)$ to the Buyer;
- The Seller and Buyer compute the ECDSA public key separately: $pk := d_S \cdot D_B = d_B \cdot D_S$ and abort if pk =0. And the secret key $sk := d_S \cdot d_B$ is secret-shared between the two parties.

3.2. The Unique Signature Generation Protocol

We will present two formalizations of the unique signature generation protocol, USG_1 and USG_2. USG_1 is a little different from the USG protocol used in Banasik's efficient Zero-Knowledge Contingent Payment system [4].

The USG protocol uses $KSignGen$ as a subroutine, in which the parties jointly sign a message z using the secret-shared signing key generated by $SharedKGen$. Recall that G is an elliptic curve group for ECDSA signature scheme $ECGen, ECSign, ECVer$, $TLCom = (TLCommit, TLForceOpen)$ is a time-locked commitment scheme, and $(AddHomGen, \text{AddHomEnc}, \text{AddHomDec})$ is a Paillier encryption scheme which is additively homomorphic over $Z_{n'}$, where $n' > 2 \cdot |G|^4$. Let F be a one-way function. Protocol KSignGen is described as follows.

- Seller and Buyer: They jointly create signing randomness K using the same way in $SharedKGen$. The Seller holds $K_S \in Z^*_{|G|}$ and $K_S := k_s \cdot g$ and the Buyer holds $K_B \in Z^*_{|G|}$ and $K_B := k_B \cdot g$. Finally the parties both know the signing randomness $K := k_s \cdot k_B \cdot g$, parse K as (x, y), compute $r := x mod\ |G|$, and abort if$r = 0$;
- Seller: He creates a new key pair $(pk_{AH}, sk_{AH}) := \text{AddHomGen}(1^n)$ in the Paillier encryption scheme, computes the ciphertext $c_S := \text{AddHomEnc}_{pk_{AH}}(d_S)$, and sends the public key pk_{AH} and the ciphertext c_S;
- Buyer: On receiving pk_{AH} and c_S from the Seller, he computes $c_0 := k_B^{-1} \cdot H(z) mod |G|$, $c_1 := \text{AddHomEnc}_{pk_{AH}}(c_0)$, $t := k_B^{-1} \cdot r \cdot d_B mod\ |G|$, $c_2 := c_1 \otimes (c_S)^t$, samples $u \leftarrow \{1, \dots, |G|^2\}$, computes $c_B := c_2 \otimes \text{AddHomEnc}_{pk_{AH}}(u \cdot |G|)$, and sends the ciphertext c_B;
- Seller: On receiving c_B, he computes $S_0 := \text{AddHomDec}_{sk_{AH}}(c_B)$, $s := k_S^{-1} s_0\ mod\ |G|$ and aborts if s=0. The final signature of z is $\sigma := (r, s)$ if $\text{ECVer}_{pk}(z, \sigma) = ok$ and otherwise the Seller aborts. At the end he commits to $S = F(\sigma)$, creates a time-lock commitment to d_S, and sends $\Gamma_i := Commit(S)$ and $\Phi_i := \text{TLCommit}(d_S)$;

We stress that when the two parties execute $KSignGen$, the one who is supposed to obtain the final signature at the end of $KSignGen$ plays the role of ``Seller".

The USG protocol is executed after the parties generate a key pairs $(sk^1, pk^1), \dots, (sk^a, pk^a)$ using the SharedKGen protocol.

- Buyer: He chooses a random subset $J \subseteq \{1, \dots, a\}$, such that $|J| = a - b$. Let $j_1, \dots, j_b$ denote the set $\{1, \dots, a\} \backslash J$. And then he chooses message z to be signed and sends it to the Seller ;
- Seller and Buyer: For $i = 1$, the parties execute the $KSignGen(1^n)$ protocol with $pk_i := d_S^i \cdot d_B^i \cdot g$. And at the end of each execution, the Seller sends the commitment $\Gamma_i := \text{Commit}(S^i)$, where $S^i = F(\sigma_i)$, and the time-lock commitment $\Phi_i := \text{TLCommit}(d_S^i)$;
- Buyer: He sends J to Seller ;
- Seller: For each $j \in J$, the Seller opens the commitments to S^j and d_S^j, and sends σ^j, k_S^j and sk_{AH}^j to Buyer ;
- Buyer: He aborts if any of the commitments did not open correctly. Otherwise, for each $j \in J$, he verifies if the following holds:
  - $\text{ECVer}_{pk^j}(z, \sigma^j) = ok$;
  - $F(\sigma^j) = S^j$;
  - $d_S^j \cdot d_B^j \cdot g = pk^j$;
  - $\text{AddHomDec}_{sk_{AH}}^j(c_S^j) = d_S^j$.

  If the verification fails then the Buyer aborts. Otherwise, the unique signature is the set $\{\sigma^{j_i}\}$, where $i \in \{1, \dots, b\}$.

In this paper, the USG_1 protocol is the same as the USG protocol except that USG_1 outputs the set $\{\sigma^i\}$, where $i \in \{1, \dots, a\}$, as the unique signature. For the technique used in the final protocol, we assume that each $S^i$from the USG_1 protocol is divided into $2n$ parts $S^{i,1}, \dots, S^{i,2n}$ each of size n. In addition, we assume that each part $S^{i,j}$ is committed separately.

The USG_2 protocol is the same as the $\text{KSignGen}(1^n)$ protocol except that there is no further computation after obtaining the signature $\sigma := (r, s)$, of which the verification result is ok.

4. THE AUGMENTED ZERO-KNOWLEDGE CONTINGENT PAYMENT PROTOCOL

In this section we show how to use the subprotocols to construct the augmented two-party Zero-Knowledge Contingent Payment Protocol protocol with fairness in the following sense: if the malicious seller aborts protocol without broadcasting the final message then the buyer finally gets his payment back and an extra financial compensation from the seller.

4.1. Security Definition

In our protocol, the Seller sells to the Buyer a value x such that $f(x) = true$ (for some public function $f: \{0,1\}^* \to \{true, false\}$. We assume that the price of x is v Bitcoins and the deposit of the Seller for this transaction is p Bitcoins. Hence, before an execution of the protocol starts, there is some unspent transaction T_0 on the blockchain that sends v Bitcoins to pk_B and some unspent transaction T'_0 on the blockchain that sends p Bitcoins to pk_S. The parties initially share the following common input: security parameter 1^n, price v for the secret x, deposit value$p$ for the computation of a time-lock committed message, parameters $a, b \in N$ such that $a > b$, an elliptic curve group $G, O, g, +$ for an ECDSA signature scheme, such that $\lceil log_2 |G| \rceil =$

n, and parameters (τ_0, τ_1) for the time-lock commitment. We say that the final protocol is $\varepsilon - secure$ if, except with probability $\varepsilon + \mu(n)$, the following properties hold:

- if the honest Buyer loses his funds then he learns x'such that $f(x') = true$;
- if the honest Seller does not get Buyer's funds then the Buyer learns no information about x;
- if an honest Buyer is forced to open a time-lock commitment then he finally does not lose his funds and obtains a financial compensation from the Seller.

4.2. Instantiations and Assumptions

Instantiations. F is a one-way function and we use a standard symmetric encryption scheme $\mathrm{EncGen}, \mathrm{Enc}, \mathrm{Dec}$ and the additively-homomorphic public key encryption scheme $(\mathrm{AddHomGen}, \mathrm{AddHomEnc}, \mathrm{AddHomDec})$ introduced by Pascal Paillier [11]. The elements on which we perform the addition operations are the exponents in the elliptic curve group of the ECDSA scheme. Hence the Paillier encryption scheme is homomorphic over $Z_{n'}$, where $\mathrm{n}' > 2 \cdot |\mathrm{G}|^4$, and $\lceil log_2|G| \rceil = n$.

We use a standard commitment scheme $\mathrm{Com} = (\mathrm{Commit}, \mathrm{Open})$ that is based on hash function and secure in the random oracle model [4]. Let H be a hash function and a commitment to message x is defined as Commit (x):=H(x||r), where $r \in \{0,1\}^*$. And Open is to reveal(x, r). The binding property follows from the collision-resistance of H because that a commitment that can be open in two different ways would form a collision for H. And the hiding property follows from the fact that H(x||r) does not reveal any information about x. We use the classic (τ_0, τ_1)-secure timed commitments $\mathrm{TLCom} = (\mathrm{TLCommit}, \mathrm{TLForceOpen})$ in [16] and assume that $\tau_1 = 10 \cdot \tau_0$ as in [4].

We consider the two-party protocol, executed between a Buyer and a Seller, in which the Seller sells to the Buyer x such that $f(x) = true$, in the active security settings.The parties are connected by a secure channel, which can be easily obtained using the standard cryptographic techniques. One user in Bitcoin is identified by his public key in the ECDSA signature scheme, which helps to establish the secure channel between each other. Without loss of generality, we set (PK_B, SK_B) and (PK_S, SK_S) to be the ECDSA key pairs of the Buyer and Seller respectively.

Assumptions. We assume that Paillier encryption and symmetric encryption are semantically secure, Com and TLCom are secure commitment schemes, and the ECDSA scheme is strongly unforgeable. Hence, the subprotocols in Section 3 are secure.

We keep the form of the assumption of zero-knowledge proof of knowledge protocol in [4]. We also assume that the public function f has a zero-knowledge proof of knowledge protocol, denoted by $\mathcal{F}$, in which the Seller can prove the knowledge of x such that $f(x) = true$. Protocol $\mathcal{F}$ consists of two phases: the $Setup_{\mathcal{F}}$ phase and the $Challenge_{\mathcal{F}}$ phase. After executing the$Setup_{\mathcal{F}}$ phase, the views of the Seller and Buyer are denoted by $S_{\mathcal{F}}$ and $B_{\mathcal{F}}$ respectively.

In the $Challenge_{\mathcal{F}}$ phase, the Buyer generates challenge message as $c_{\mathcal{F}} = \mathrm{GenChallenge}_{\mathcal{F}}(B_{\mathcal{F}})$ and sends $c_{\mathcal{F}}$ to Seller. Then the Seller calculates the corresponding response $r_{\mathcal{F}} = \mathrm{GenResponse}_{\mathcal{F}}(x, S_{\mathcal{F}}, c_{\mathcal{F}})$ and sends $r_{\mathcal{F}}$ to the Buyer. At the end, the Buyer decides to accept or reject the proof according to the output of function $= \mathrm{VerifyResponse}_{\mathcal{F}}(B_{\mathcal{F}}, c_{\mathcal{F}}, r_{\mathcal{F}}) \in \{\mathrm{true}, \mathrm{false}\}$.

Protocol $\mathcal{F}$ is a proof of knowledge since we require that there is a knowledge extractor $\text{Extract}_{\mathcal{F}}$ such that $\text{Extract}_{\mathcal{F}}(B_{\mathcal{F}}, c_{\mathcal{F}}^1, r_{\mathcal{F}}^1, c_{\mathcal{F}}^2, r_{\mathcal{F}}^2)$ and $f(x') = true$ if only $= \text{VerifyResponse}_{\mathcal{F}}\left(B_{\mathcal{F}}, c_{\mathcal{F}}^i, r_{\mathcal{F}}^i\right) = true$ for $i = 1,2$ and $c_{\mathcal{F}}^1 \neq c_{\mathcal{F}}^2$. It means that a witness for f can be extracted from the correct answers corresponding to two different challenges. We also assume that the challenge $c_{\mathcal{F}}$ is sampled uniformly from the set $X_{A_{\mathcal{F}}} = \{0,1\}$ in Seller's view.

4.3. The Protocol

Our protocol consists of four stages and the final protocol called SellWitness is depicted on Fig.1.

- **Stage 1.** Using the two-party ECDSA key generation protocol SharedKGen, the Buyer and Seller jointly generate $a+1$ key pairs $(sk^1, \text{pk}^1), \ldots, (sk^{a+1}, \text{pk}^{a+1})$, where $\text{pk}^i := d_S^i \cdot d_B^i \cdot g$ and $sk^i := d_S^i \cdot d_B^i$. As a result, the Buyer holds $(PK, SK_0) := ((\text{pk}^1, \ldots, \text{pk}^a),$
- $(d_B^1, \ldots, d_B^a))$, $(PK', SK'_0) := (\text{pk}^{a+1}, d_B^{a+1}))$ and the Seller holds $(PK, SK_1) := ((\text{pk}^1, \ldots, \text{pk}^a), (d_S^1, \ldots, d_S^a))$, $(PK', SK'_1) := (\text{pk}^{a+1}, d_S^{a+1}))$;
- **Stage 2.** The parties respectively produce the messages to be signed. If one prepares a transaction T, then the message to be signed is $[T]$. The procedure is called GenMsg_T. The Seller prepares transaction T_3 sending the funds p from PK_S to PK' and T'_3 sending the funds p fromPK' to PK_S, and then sends the hash value ofT_3 to the Buyer. And then the Buyer prepares transaction T_1 sending the funds v from PK_B to PK', T'_1 sending the funds $v+p$ from PK' to a-out-of-$a+b-1PK, PK_B, \ldots, PK_B$ (consists of $b-1PK_B$), and T_2 sending the funds $v+p$ from PK to PK_S. They execute the unique signature generation protocol USG_2 using (PK', SK'_0) and (PK', SK'_1) to generate the signature of$[T'_1]$for the Buyer and the signature of $[T'_3]$ for the Seller. And then they execute the USG_1 using (PK, SK_0) and (PK, SK_1) to generate the signatures of $[T_2]$ for the Seller, in which the Seller will commit to SK_1 using TLCom and send to the Buyer. And then the Seller broadcasts T_3. If T_3 is not corresponding to the hash value of sent before, then the Buyer aborts; otherwise, the parties execute the following stages;
- **Stage 3.** The Seller proves the knowledge of x such that $f(x) = true$ by executing a zero-knowledge proof of knowledge protocol $\mathcal{F}$ with the Buyer in the cut-and-choose technique;
- **Stage 4.** The Buyer either extracts x such that $f(x) = true$ or takes his funds back and obtains the Seller's deposit.

Theorem. Suppose Paillier encryption and symmetric encryption are semantically secure, Com and TLCom are secure commitment schemes, and the ECDSA scheme used in the construction of the USG_1 and USG_2 is strongly unforgeability. Additionally, there is a zero knowledge proof $\mathcal{F}$ of knowledge of x s.t. $f(x) = true$ of the required form. Then the SellWitness (Fig.1) is ε-secure for $\varepsilon = (b/a)^b$.

Proof. Recall that SellWitness is ε -secure if, except with probability $\varepsilon + \mu(n)$, the following properties hold: (1) if an honest Buyer loses funds then he learns x' such that $f(x') = true$; (2) if an honest Seller does not get Buyer's funds then the Buyer learns no information about x; (3) if an honest Buyer is forced to open a time-lock commitment, then he finally does not lose his funds and obtains a financial compensation from the Seller. Hence then, we show the security analysis of SellWitness in two cases: (i) the Buyer is honest and the Seller is mailicious, and (ii) the Seller is honest and, while the Buyer is malicious.

At the beginning of this proof, we stress that the security of the unique signature generation protocol USG_1 follows that of the USG protocol used in Banasik's efficient Zero-Knowledge

Contingent Payment system [4], and the security of the USG_2 protocol follows that of the ECDSA.

For case (i), the honest Buyer loses his funds only if he broadcasts transactions T_1 and T'_1 is redeemed by the Seller or it just locks the Buyer's funds forever. Since with probability$\geq 1 - (b/a)^b - \mu_0(n)$ (for a negligible $\mu_0(n)$) at least one of the b chosen executions, consist of execution i for $i \in \{1, \dots, a\} \backslash j$, of the KSignGen procedure was completed honestly by the Seller (guaranteed by the security of USG_1 protocol). While if the Seller does not redeem T'_1, the honest Buyer will be able to redeem transaction T'_1 and get back his funds together with the Seller's deposit after that the Buyer force-opens one time-locked puzzle Φ_i in a honestly completed execution i of KSignGen, where $i \in \{1, \dots, a\} \backslash j$. Hence then, the property (3) follows and the Buyer's funds cannot be locked forever except with probability $(b/a)^b - \mu_0(n)$.

Assume that the Seller redeems transaction T'_1 and he can redeem T'_1 only via transaction T_2. Then the Buyer can use signatures $\hat{\sigma}_i$ to calculate secrets $S^{i,j}$. Then he decrypts all the values $\sigma^{i,j}$ to get all the challenges and responses $c_k^{i,j}$,$r_k^{i,j}$. At the end using any pair of responses he can calculate $x' = \text{Extract}_{\mathcal{F}}(B_{\mathcal{F}}^{i,j}, c_1^{i,j}, r_1^{i,j}, c_2^{i,j}, r_2^{i,j})$ s.t. $f(x') = true$ with probability $\geq 1 - \mu_1(n)$, where $\mu_1(n)$ is negligible (see the proof of Lemma 2 in [4]). Hence then, the property (1) follows.

For case (ii), if an honest Seller does not get the Buyer's funds, the property (2) is guaranteed by the zero-knowledge property of the zero-knowledge proof of knowledge protocol.

Common Input: function f: $\{0,1\}^* \to \{\text{true}, \text{false}\}$.
Private Input: Seller holds value x such that $f(x) = true$.

- **Stage 1:**
  - The parties execute the SharedKGen protocol $a+1$ times using the provided parameters. As a result, the Buyer holds $(PK, SK_0), (PK', SK'_0)$ and the Seller holds$(PK, SK_1), (PK', SK'_1)$ as defined previously.
- **Stage 2:**
  - The Seller generates transactions T_3 and T'_3, and then sends the hash value of T_3 to the Buyer;
  - The Buyer generates transactions T_1,T'_1 and T_2;
  - The Seller and buyer executeUSG_1 and USG_2 with the keys generated in Stage 1 to generate the corresponding signatures of $[T'_1]$, $[T'_3]$ and $[T_2]$. And the Buyer holds complete transaction T'_1 and Seller holds complete transactions T'_3 and T_2, and the unique signature $\{\sigma^i\}$ $(i \in \{1, \dots, a\})$ of $[T_2]$;
  - The Seller sends commitment $\Gamma_i \coloneqq \text{Commit}(S^i)$ of $S^i = \mathcal{F}(\sigma^i)$ and the time-lock commitment $\Phi_i \coloneqq \text{TLCommit}(d_S^i)$. Each S^i is divided into 2n parts $S^{i,1}, \dots, S^{i,2n}$ with size n and each part $S^{i,j}$ is committed separately. Let $\{S^i\}$, where $i \in \{1, \dots, a\} \backslash J$, denote the unopened set of $\Gamma_i \coloneqq \text{Commit}(S^i)$. And the Seller also broadcasts T_3. If T_3 is not corresponding to the hash value sent before, the Buyer aborts. Otherwise, the parties execute the following stages.
- **Stage 3:** for $i \in \{1, \dots, a\} \backslash J$,
  - For $j \in \{1, \dots, 2n\}$, the Seller and Buyer execute the $\text{Setup}_{\mathcal{F}}^{i,j}$ phase and learn the views $\text{S}_{\mathcal{F}}^{i,j}$ and $\text{B}_{\mathcal{F}}^{i,j}$ respectively;
  - For $j \in \{1, \dots, 2n\}$, the Seller calculates two challenges $c_1^{i,j}$ and $c_2^{i,j}$ in random order, which will be chosen by the Buyer later to calculate the responses $r_k^{i,j} = \text{GenRespose}_{\mathcal{F}}(x, B_{\mathcal{F}^{i,j}}, c_k^{i,j})$ for $k = 1.2$;
  - For $j \in \{1, \dots, 2n\}$,, the Seller uses $\text{S}_{\mathcal{F}}^{i,j}$ to encrypt $\Gamma_k^{i,j} = \text{Enc}_{S^{i,j}}(c_k^{i,j}, r_k^{i,j})$ for $k = 1,2$ And he commits to $\gamma_k^{i,j}$for $k = 1.2$;
  - The Buyer chooses random subset $\mathcal{J}^i \subseteq \{1, \dots, 2n\}$ of size n and sends $i, c_B^{i,j} \coloneqq \text{GenChallenge}_{\mathcal{F}}$ for $j \in \mathcal{J}^i$ to the Seller;
  - For $j \in \mathcal{J}^i$, the Seller opens his commitment to $S^{i,j}$ and checks that $c_B^{i,j} = c_k^{i,j}$ for $k = 1$ or $k = 2$. He opens the commitments to $\Gamma_k^{i,j}$ for only this k;
  - For $j \notin \mathcal{J}^i$, the Seller opens his commitments to $\Gamma_k^{i,j}$ for $\text{k} = 1,2$;
  - The Buyer verifies all the commitments;
  - For $j \in \mathcal{J}^i$, the Buyer decrypts $(c^{i,j}, r^{i,j}) = \text{Dec}_{S^{i,j}}(\gamma_k^{i,j})$, and checks that $c^{i,j} = c_B^{i,j}$ and $\text{VerifyResponse}_{\mathcal{F}}\left(B_{\mathcal{F}}^{i,j}, c_B^{i,j}, r^{i,j}\right) = true$.
- **Stage 4:**
  - The Buyer broadcasts T_1 and T'_1, and then the two parties wait until they are confirmed finally;
  - If transactions T_1 and T'_1 are valid as expected, the Seller broadcasts T_2 and uses signatures $\hat{\sigma}_1, \dots, \hat{\sigma}_a$ to get her payment, and otherwise he broadcasts T'_3 to has his deposit back;
  - If transaction T_2 is valid as expected, the Buyer uses signature $\hat{\sigma}_i$ to calculate secrets $S^{i,j}$ and then decrypts all the values $\gamma^{i,j}$ to get the corresponding challenges $c_k^{i,j}$and responses $r_k^{i,j}$. At the end, the Buyer uses any pair of responses to calculate $x' = \text{Extract}_{\mathcal{F}}(B_{\mathcal{F}}^{i,j}, c_1^{i,j}, r_1^{i,j}, c_2^{i,j}, r_2^{i,j})$. Otherwise, he force-opens time-locked puzzles Φ_i $(i \in \{1, \dots, a \backslash J\})$, and uses any of the opened values d_S^i to get his funds back and obtain the Seller's deposit as well.

Figure 1. The SellWitnessProtocol.

5. CONCLUSIONS

In this paper, with respect to the unfairness between the buyer and seller in the Zero-Knowledge Contingent Payment protocol, we present protocol SellWitness to achieve that, if the seller is malicious, then the honest buyer will get back his payment and some penalty paid by the seller. In

the further work, we will optimize SellWitness, e.g., replacing the time-lock building block with some timeless primitives to allow the parties to have unlimited number of valid transactions.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their valuable comments. This work is supported in part by the National Key R&D Program of China (No. 2020YFB1005801) and in part by the National Natural Science Foundation of China (No. 62172396).

REFERENCES

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. Decentralized Business Review, 2008: 21260.
[2] Campanelli M, Gennaro R, Goldfeder S, et al. Zero-knowledge contingent payments revisited: Attacks and payments for services[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 229-243.
[3] Greg Maxwell. "The first successful Zero-Knowledge ContingentPayment."https://bitcoincore.org/en/2016
[4] Banasik W, Dziembowski S, Malinowski D. Efficient zero-knowledge contingent payments in cryptocurrencies without scripts[C]//European symposium on research in computer security. Springer, Cham, 2016: 261-280.
[5] Bentov I, Kumaresan R. How to use bitcoin to design fair protocols[C]//Annual Cryptology Conference. Springer, Berlin, Heidelberg, 2014: 421-439.
[6] Andrychowicz M, Dziembowski S, Malinowski D, et al. Fair two-party computations via bitcoin deposits[C]//International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2014: 105-121.
[7] Goldrcich O, Vainish R. How to solve any protocol problem-an efficiency improvement[C]// Conference on the Theory and Application of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1987: 73-86.
[8] An J H, Dodis Y, Rabin T. On the security of joint signature and encryption[C]//International conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 2002: 83-107.
[9] Boneh D, Shen E, Waters B. Strongly unforgeable signatures based on computational Diffie-Hellman[C]//International Workshop on Public Key Cryptography. Springer, Berlin, Heidelberg, 2006: 229-240.
[10] Katz J, Lindell Y. Introduction to modern cryptography[M]. CRC press, 2020.
[11] Paillier P. Public-key cryptosystems based on composite degree residuosity classes[C]//International conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 1999: 223-238.
[12] Boneh D, Naor M. Timed commitments[C]//Annual international cryptology conference. Springer, Berlin, Heidelberg, 2000: 236-254.
[13] Rivest R L, Shamir A, Wagner D A. Time-lock puzzles and timed-release crypto[J]. 1996.
[14] Bellare M, Goldreich O. On defining proofs of knowledge[C]//Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1992: 390-420.
[15] Goldreich, O.: Foundations of Cryptography, vol. 1. Cambridge University Press, New York (2006). ISBN: 0521035368.
[16] Goldwasser S, Micali S, Rackoff C. The knowledge complexity of interactive proof-systems[M]//Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali. 2019: 203-225.

AUTHORS

Peifang Ni received the Ph.D. degree in the Institute of Information Engineering, Chinese Academy of Sciences, in 2020. She is currently a postdoctoral with the Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences. Her major research interests include applied cryptography, security protocol and blockchain consensus.