Received: 3 November 2017 | Revised: 22 December 2017 | Accepted: 9 February 2018
DOI: 10.1002/cpe.4469

WILEY

SPECIAL ISSUE PAPER

Blockchain-based fair three-party contract signing protocol for fog computing

Hui Huang[1,2] | Kuan-Ching Li[3] | Xiaofeng Chen[1]

[1]State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, China
[2]College of Computer, Minnan Normal University, Zhangzhou, China
[3]Department of Computer Science and Information Engineering, Providence University, Taichung, Taiwan

Correspondence
Xiaofeng Chen, State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an 710071, China.
Email: xfchen@xidian.edu.cn

Funding information
National Natural Science Foundation of China, Grant/Award Number: 61572382; Key Project of Natural Science Basic Research Plan in Shaanxi Province of China, Grant/Award Number: 2016JZ021; China 111 Project, Grant/Award Number: B16037

Summary

Fog computing is a new computing paradigm that can provide flexible resources and services at the edge of network. It is an extension of cloud computing and usually cooperated with cloud computing. Therefore, end users, fog nodes, and cloud servers can form a three-layer service model in practical application. In this model, they should have an agreement on a service contract, which contains every party's rights and obligations before the beginning of the service. However, due to lack of trust, it will suffer from some fairness problems during signing a service contract. Contract signing protocol allows two or more mutual distrust entities to sign a predefined digital contract in a fair and effective way. In this paper, we propose a fair three-party contract signing protocol based on the primitive of blockchain, which can be applied to the scenario of fog computing. Our proposed construction allows the participants to sign a contract in a fair way without the involvement of an arbitrator. Moreover, the privacy of the contract content can be preserved on the public chain. Finally, we realize the proposed protocol through the private blockchain and provide the experimental simulation that analyzes the efficiency and effectiveness.

KEYWORDS

blockchain, contract signing, fog computing

1 | INTRODUCTION

In order to meet the needs of applications such as Internet of things and artificial intelligence, the concept of fog computing emerges as an extension of cloud computing. Fog computing is a distributed computing model and can provide new applications for all kinds of network users. Due to its flexibility, fog computing can solve the problems that are still unsolved in cloud computing, such as network congestion, poor mobility, and high latency.[1] In fog computing, besides resource-rich servers, fog nodes can also be resource-poor devices, like set-top box, gateways, small base station, and wireless router. As cloud computing, fog also provides services to end users such as data storage,[2,3] computation,[4] and other applications. However, fog computing comes closer to end users than cloud computing.[5-7]

Fog computing is the extension of rather than the competition with cloud computing. In practical scenarios, end users, fog nodes, and cloud servers can form a three-parties model (see Figure 1), which can support a series of applications such as data analysis, content delivery, and augmented reality. In the three-layer service model, there are three parties. The first one is the cloud service provider, Alice, who has resource-rich servers and wants to extent his service to the edge of network. The second one is the Internet service provider, Bob, who conducts fog nodes at his infrastructures. These fog nodes are distributed in different positions of network. Alice and Bob work together and provide services for end user, Charlie, who wants to require services.

In this work, we consider the scenario in which there are three parties, Alice, Bob, and Charlie. They should have an agreement on a service contract, which contains every party's rights and obligations before the beginning of the service. That is, they should sign a digital service contract through network. Actually, signing a contract over a network is more complicated than signing in physical world. Especially in the fog environment,

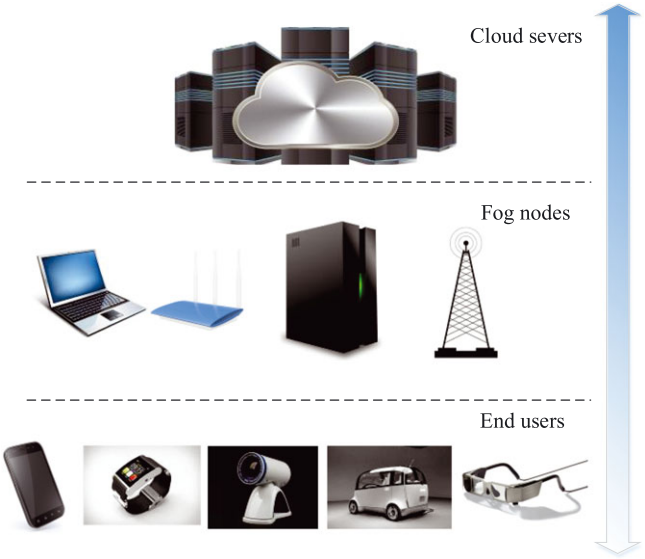

FIGURE 1 Three-layer service model[1,8]

the nodes are more decentralized and this decentralization makes centralized control become more difficult. Therefore, how to design a fair contract signing protocol suitable for the fog environment should be taken into consideration.

A contract signing protocol is a communication protocol, which allows two or more participants to sign a digital contract. An important property of contract signing protocol is fairness, permitting the participants to exchange signed contract in an all-or-nothing way. Namely, the involved participants either obtain a valid contract or nothing useful at the end of the protocol. It means that dishonest participants cannot get more advantages than honest ones. The fairness is very easy to achieve in face-to-face contract signing. However, due to the lack of trust and the decentralization of the nodes in fog computing environment, there are security threats that cannot be ignored. Therefore, signing contract electronically in a fair way has become a challenging problem.

Plenty of electronic contract signing schemes have been proposed in the literature. There are two major solutions to the problem of fairness in existing protocols. The first one is TTP-free protocol, ie, the participants exchange their signatures by means of "bit by bit." In this scenario, parties in the protocol release their secrets gradually in multiple rounds. If any of the parties fail to complete the exchange, they can still complete the protocol by looking up the remaining bits. It has the advantage of removing the trusted third party (TTP), though its high communication cost makes it unsuitable for real-world application. Another solution is to introduce a trusted third party (TTP), who is to mediate the exchange of signature. In contract-signing protocols, a TTP can make the execution of the signing process efficient. According to the participating ways of the TTP, TTP-based protocols can be divided into two kinds, ie, the online TTP protocols and the off-line TTP protocols. The main problem of the online setting is that the TTP becomes a bottleneck of the system when there are multiple participants in the system. Therefore, the offline TTP setting is more practical. A contract signing protocol with an offline TTP, in which a TTP is only involved in the case of dispute, is called as optimistic contract signing.

It seems that existing solutions have disadvantages. Among these protocols, some require participants to run an extra protocol to guarantee the fairness, which could be much more expensive (especially in the communication cost) than a standard protocol; some need an external trusted party, eg, in the optimistic protocol; the party needs to contact with a trusted third party each time when a dispute occurs. In summary, all these protocols exist the weakness that the honest party has to make extra effort to prevent the adversary from cheating. Blockchain provides a solution to solving the trust problem, which was proposed as the underlying technology of Bitcoin.[9] Bitcoin is a peer-to-peer decentralized digital e-cash system and uses the cryptography tools to build a currency system without a trusted third party. Its novel and open design has attracted more attention. Due to the advantage of Bitcoin system, it is applicable to design fair protocols.[10] Recently, Andrychowicz et al[11,12] showed the advantages of Bitcoin when it came to design fair multiparty computations. To the best of our knowledge, it seems that there is no similar investigation on fair contract signing protocol based on blockchain.

In this paper, we propose a fair three-party contract signing protocol based on blockchain. Different from previous works, our proposed protocol does not need a third party to guarantee the fairness. The contributions of this paper are as follows.

- We construct a fair three-party contract signing protocol without a trusted third party, which is suitable for the decentralized fog computing environment. The proposed protocol combines threshold public key encryption with verifiable encryption to protect the privacy of contract content. Moreover, it uses the blockchain to finish the fair exchange. The proposed protocol enables a dishonest party to be monetarily penalized when it aborts after receiving the current output. Different from the previous works, there is not a bank or a trusted third party to guarantee the fairness. The setting of our structure is suitable for fog computing environment and it is more practical than the existing ones.

- Our contract signing protocol requires only constant number of communication rounds. Besides, our scheme can guarantee the fairness even when two parties are malicious and colluding. In setup phase, the same participants can use the same parameters to sign different contract without rerunning the setup phase. The setting of our structure is more efficient than the previous ones.
- The privacy of the contract can be protected on the public chain. The parties only publish the decryption shares to the blockchain. As a result, the nodes on the public chain only learn some decryption shares. Since the nodes never get the encrypted items, they cannot decrypt the encryption of signed contract.

This paper is an extension of our previous paper, which was presented at CSS 2017.[13] The main differences between this paper and the conference version are as follows. First, we add application scenario of our protocol and append the related work in Section 1. Second, we present a more elaborate protocol, add the high description of proposed protocol, and a flow chart in Section 3.2. We also give the security definitions for our proposed protocol in Section 3.1 and prove that our protocol satisfies these security properties in Section 4.1. Finally, we provide the experimental simulation of our protocol and time cost comparison between our scheme and previous work in Section 4.3.

1.1 | Related work

Fog computing. In 2012, Bonomi et al[14] introduced the concept of fog computing, which extends computation services to the edge of the network. Then, a few works have investigated the concept of fog computing.[15-18] Fog computing is viewed as a virtualized resource pool that provides computation, storage, and other services to end users. Vaquero and Rodero-Merino[19] considered fog computing as "a scenario in which a large number of decentralized devices work together to perform storage and processing tasks. These devices can be heterogeneous and ubiquitous. Moreover, they can cooperate with each other without the help of third parties." Stojmenovic and Wen[20] demonstrated the scenario of man-in-the-middle attack in fog computing. During the process of the attack, the gateway could be replaced by a fake one. Hong et al[15] introduced a high-level architecture for fog computing platform. Yi et al[8] pointed out that the applicable business model for fog computing should be considered, which can help the development of fog computing. In order to realize "Pay-as-you-go" model in fog computing, business models in place should be designed, such as how to sign service contract between servers and users and how to define the billing rules, etc.

Contract signing. Plenty of contract signing protocols have been proposed in the literature.[21-27] Based on the existence of TTP in the protocol, the existing construction can be divided into two types, ie, (1) protocol without a TTP and (2) optimistic model. The contract signing protocol without a TTP had been proposed by Goldreich.[28] This protocol enabled the parties to release their secret gradually in multiple rounds. It had the advantage of removing the TTP, but it was not suitable for real-world application due to its high communication cost. Ben-Or et al[29] proposed the idea of introducing an online TTP to guarantee fairness. However, the existence of online TTP becomes a bottleneck, especially in the multi-party setting.

The model of optimistic fair exchange was considered by Asokan et al.[30] In their construction, they introduced an offline third party, which only got involved in the protocol when there was a dispute. Later, this notion has been extended to contract signing protocols.[31,32] The first optimistic multi-party contract signing protocol was designed by Garay and MacKenzie.[21] However, due to the high communication cost, their construction was inefficient. Besides, Chadha et al[24] pointed out the solution in [21] was unfair and they gave an improved scheme. Baum-Waidner and Waidner[23] realized a more efficient one. However, the efficiency depended on the numbers of the dishonest participants. Furthermore, when the number of dishonest participants reached $n - 1$, there was no difference between [21] and [23] in efficiency. Mukhamedov and Ryan[26] introduced the abort chaining attacks, which could be used to prove that Chadha et al's protocol did not satisfy the fairness requirements if there were more than five parties. Mauw et al[27] developed a fair multi-party contract signing protocol with low message complexity based on the optimistic model. Kılınç and Küpçü[22] proposed a general construction for fair exchange which required a constant number of rounds and $O(n)$ messages. In addition, they showed that their construction can be used to design secure multi-party computation with little overhead.

Blockchain. The concept of Bitcoin has been proposed by Nakamoto.[9] Andrychowicz et al[11] showed that the advantages of Bitcoin transaction syntax could be used to design secure multiparty computation protocols. They constructed a Bitcoin-based timed commitment scheme. In their scheme, the committer should either reveal his secret in a limited time or pay a fine. They also realized a secure multiparty lotteries protocol based on Bitcoin. Blockchain, as the underlying technology of Bitcoin, is summarized recently by many researchers.[33,34] They describe the principles of electronic cash and blockchain. Zhao et al[35] analyzed the problems of coin network and the efficiency of network transaction. Tschorsch and Scheuermann[36] expound the impact on the field of science and the economic field after the emergence of blockchain. Blockchain has the features of open, centralization, and permanent. Due to the low degree freedom of the Bitcoin network, the concept of smart contract has been proposed by Christidis et al and Devetsikiotis.[37] Smart contract is a set of commitments defined in digital form, including the agreement that the contractor can perform on these commitments. Smart contract can solve the problem of single center; it can be used to solve the problem of high degree of freedom of the blockchain.

1.2 | Organization

The rest of this paper is organized as follows. Some preliminaries are given in Section 2, and the proposed fair three-party contract signing protocol is given in Section 3. The security analysis and performance analysis is given in Section 4, and finally, the conclusion remarks in Section 5.

2 | PRELIMINARIES

In this section, we first give an overview of threshold ElGamal encryption and verifiable encryption signature. Then, we introduce some notions of blockchain.

2.1 | Threshold ElGamal encryption

In a threshold public key cryptosystem,[38] a message is encrypted by a single public key, which is generated jointly by a set of decrypters. The secret key corresponding to this public key is share among them. Therefore, the ciphertext can be decrypted by a part of them working together. For example, consider a k out of n threshold encryption scheme; a single public key is used to do the encryption on a message, but the corresponding secret key is shared among a set of n decrypter. Only k of them work together; they can decrypt an encrypted message.

A threshold encryption scheme includes four probabilistic polynomial time algorithms, ie, *Key Generation*, *Verification*, *Decryption*, and *Encryption*.[22,38] In this paper, we will adopt the *ElGamal* threshold encryption (ie, $k = n$) scheme to build our protocol. Assume there are n participants in the scheme. They have previously agreed on the primes p and q, where $q \mid p-1$. Let g be a generator of $\mathbb{Z}_q^*$; $\mathbb{Z}_p$ is a group with the large prime order p.

- *Key Generation:* Each party randomly chooses $x_i \in \mathbb{Z}_p$ and the set of private keys are $SK = \{x_1, x_2, \cdots x_n\}$. Compute $h_i = g^{x_i}$. As scheme [22] and [38] are done, each party commits to h_i and broadcasts to all participants. Denote these public verification keys as $PVK = \{h_1, \ldots, h_n\}$. The public key is $PK = (g, h)$, where $h = \prod_{i=1}^{n} g^{x_i} = g^{\sum x_i}$.
- *Encryption:* Given a message m, the party randomly chooses $r \in \mathbb{Z}_q$ and computes $a = g^r, b = mh^r$. The ciphertext is $E = (a, b)$.
- *Verification:* The verification algorithm is run by given the verification key PVK, the ciphertext E, and the decryption share $d_i = g^{rx_i}$ belonging to party i. If $\log_g h_i = \log_g d_i$, it outputs *valid*. Otherwise, it outputs *invalid*.
- *Decryption:* The decryption algorithm takes input the n decryption $d_i, i \in \{1, \cdots n\}$. Then, it computes $d = \prod d_i$ and decrypts the ciphertext as

$$m = \frac{b}{d} = \frac{mh^r}{g^{r\sum x_i}} = \frac{mh^r}{h^r}. \tag{1}$$

2.2 | Verifiable encrypted signature

In a verifiable encrypted signature scheme, a signer can encrypt his message signature and the recipient can verify the signature without performing any decryption. Asokan et al[39] formally introduced the idea of verifiable encrypted signature. Boneh et al[40] presented the first non-interactive verifiable encrypted signature scheme by verifiable encrypting the BLS signature.[41] Recently, Shao and Gao[42] proposed the first verifiable encrypted discrete-log signature scheme. In this section, we will combine Shao and Gao's verifiable encrypted signature with the threshold encryption to describe the scheme.

As described in Section 2.1, considering a k out of n threshold encryption scheme, a single public key is used to perform the encryption on a message. In addition, n decrypters keep the corresponding secret key, respectively. Only k of them work together; they can decrypt a message. Assume every party has obtained the public key h and the corresponding secret key has been distributed between them. Here, we consider $k = n$. The party P_1 wants to run a verifiable encrypted signature scheme to send his encrypted signature to P_2. P_2 can use the public key h and the public key of P_1 to verify the signature without performing decryption. A verifiably encrypted signature scheme consists of five probabilistic polynomial time algorithms, ie, *Setup*, *KeyGen*, *VEsign*, *VEVerify*, and *Decryption*.[40,42]

- *Setup:* On input the secure parameters and generates public hash functions and the system parameters.
- *KeyGen:* Given the system parameters, the participants generates key pair (x_i', y_i), which is used for signing the contract. Then, they work together to generate the public key of threshold encryption h.
- *VEsign:* For a message $M \in \{0,1\}^*$, the signer P_i computes a signature (M, σ) with his private key x_i', then generates a VES_i with the public key of threshold encryption h.
- *VEVerify:* After receiving the verifiable encrypted signature of the message M, the verifier checks the verification equations for the VES_i with the respect to the public key y_i and h.
- *Decryption:* The decryption algorithm takes input the n decryption shares of all participants. Then, each participant extracts the signature with the decryption shares.

2.3 | Blockchain

A blockchain is essentially a distributed database on the peer-to-peer network. It maintains continuously growing blocks, which record the data and can prevent these data from tampering and revision. The blockchain was first used in Bitcoin, which was conceptualized by Satoshi Nakamoto

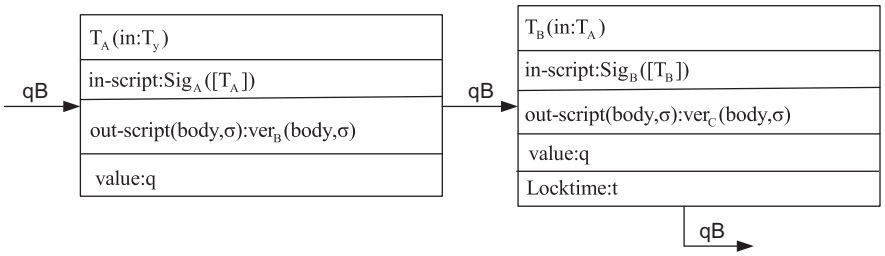

FIGURE 2 The graph shows the relationship of two transactions. Assume T_A and T_B are two related transactions produced by two users. *A* and *B* are two public key. T_y is a previous transaction can be redeemed by *A*. [T_A] denotes messages containing the input script of *T* and all content of T_A except the output script. This example describes that T_A can be redeemed by the public key *B* in time *t*

in 2008.[9] It was a core component of the Bitcoin system. The invention of the blockchain enabled Bitcoin to become the first e-cash system without the use of a trusted authority.[43]

The Bitcoin transactions are verified by network nodes and recorded in the data blocks. A data block is composed of Header and Body. The block body contains some transactions. When a new transaction broadcasts in the system, the nodes called miners collect this new transaction. When one of them finds a proof-of-work, it broadcasts the block to all nodes. This transaction is confirmed when this transaction appears on the public block, but the process needs 10 to 20 minutes or more.

Addresses and transactions are two components of Bitcoin system. Every user in the system has a key pair. A Bitcoin address is the hash of the user's public key. The user can use the private key to sign a transactions and use the public key to verify the signatures. A transaction has some input and output. The input script of the transactions is a signature. The output script is a verification algorithm, which controls the conditions whether the coins can be redeemed or not. In a real system, users can define the condition flexibly. In the standard transaction, input-script is just a signature with the sender's secret key, and out-script implements a signature verification with the recipient's public key. If the time *t* is 0, it means the transaction is locked and final. As an example, Figure 2 gives the relationship of two transactions.

3 | THE PROPOSED FAIR CONTRACT SIGNING PROTOCOL

3.1 | Security definitions

In this section, we introduce a concrete construction of three-party contract signing protocol. The three participants are denoted by P_A, P_B, and P_C. We assume all the participants can be either honest or malicious. We define that the party P_A obtains a valid contract only if P_A has received the other parties' signature on the contract. If P_A only received one party's signature, say P_B, then the contract is invalid.

We assume all messages between the participants are reliably transferred. Our protocol does not need the messages to be transferred in sequence. We also assume the participants use a resilient communication channel to communicate. The property of resilient means the communication model is asynchronous and there is no global clock. In this model, the messages can be delayed arbitrarily but will reach eventually in a finite time. In order to avoid controversy, the time to be delayed in the communication channel should be included in the deadline in our protocol.

The security properties of the proposed three-party contract signing protocol are defined in terms of *completeness*, *fairness*, *timeliness*, *non-repudiation*, and *confidentiality*.[31]

- *Completeness*. If every participant involved in the protocol is honest, the adversary cannot prevent all the parties exchanging their valid signature on the contract. Adversary can be allowed to access the signature oracle or can request any message except the contract. In addition, the adversary can arbitrarily delay the messages between the participants.
- *Fairness*. The fairness is defined as all of the parties either obtain the others' signature on the contract, or none of them obtains the valid signed contract. In our scheme, the fairness means that if the adversary obtains the valid signed contract, he will pay for the honest parties.
- *Timeliness*. Any participant involved in the protocol can be sure that the protocol have limited time to run and it will abort at a certain point of time. The certain deadline time will be agreed on in advance of the protocol. Once the protocol aborts, the fairness can be achieved for the honest parties whatever the other parties do.
- *Non-repudiation*. Given a signed contract, every participant involved in the protocol cannot deny having signed it. We say that non-repudiation is the precondition of fairness.
- *Confidentiality*. The process of signature exchange is secure only when the participants involved in the protocol are allowed to know the content of the contract. No one can have access to the plain contract.

3.2 | Main idea

We assume three parties $P_i, i \in \{A, B, C\}$ are involved in our protocol. They have come to an agreement on the content of the contract M. Then, each of them generates a signatures σ_i on the contract. At the end, they will obtain all the three signatures $\{\sigma_A, \sigma_B, \sigma_C\}$. First, all parties $P_i, i \in \{A, B, C\}$ use their secret share x_i, which is randomly selected by P_i, to generate a public key h together. Each party P_i uses this public key to encrypt his signature σ_i using the verifiable encryption signature scheme. Then, P_i sends the verifiable encryption signature VES_i to others. Note that no party can decrypt any VES_i because they do not know the decryption shares S_i. Here, each S_i contains the secret key x_i. Even if two parties corrupt with each other, they cannot decrypt. At the same time, each P_i should make a deposit transaction via blockchain network to commit that he will reveal the decryption share S_i. If he does not reveal by the deadline, he will be punished and the honest parties will be compensated. After all parties obtain all VES_i from others, they reveal their S_i via a claim transaction. At last, when P_i receives all the shared key, he can get all σ_i by decrypting each VES_i. Upon termination, the protocol achieves one of the following outcomes within an established deadline.

- Even if two parties collude together to cheat, the protocol can guarantee the fairness.
- All parties obtain the others' signature or none of them does.
- If a party obtains the valid signed contract, but the other parties do not, then they will be compensated. The dishonest one has to pay penalty.
- The nodes on the blockchain do not know the content of the contract, and they cannot obtain the signed contract.

3.3 | Our protocol

In this section, we present a fair three-party contract signing protocol with penalties. Before describing our protocol in detail, we introduce some notations first. There are three parties involved in our protocol, denoted by $P_i, i \in \{A, B, C\}$. They have agreed on a contract M in advance. Denote by σ_i the signature on contract M with the secret key of the party $P_i, i \in \{A, B, C\}$. VES_i is a verifiable encrypted signature of P_i's signed contract under the public key h. In our protocol, we use blockchain network to guarantee the fairness. In the following, we use T_i to denote a transaction. $P_i \xrightarrow[a,t]{S} P_j$ denotes a deposit transaction produced by P_i with q coins and it can be claimed by P_j only if it provides S_j. After the time t, P_i can get back the deposit.

Note that we do not describe the agreement on the contract and the amount of the coins in detail. It may require a number of rounds of communication between the involved parties through a security channel. All parties should agree on the deadline for each phase in the protocol. Our protocol has four phases. The details are as follows (see Figure 3).

- **Key Generation**. In this phase, all parties join together to generate a public key for the verifiable encrypted signature algorithm. They first agree on a prime p-order subgroup $\mathbb{Z}_p \subset \mathbb{Z}_q^*$; q is a large prime. Assume g is a generator of $\mathbb{Z}_p$. Then, each $P_i, i \in \{A, B, C\}$ does the following.
  - First, Party P_A randomly chooses $x_1 \in \mathbb{Z}_p$ as his secret key. He computes $h_1 = g^{x_1}$. Then, he randomly selects a string r_1 and makes commitment $C_1 = C(h_1, r_1)$ to h_1 as Kılınç[22] and Pedersen[38] done. Send C_1 to party P_B and P_C.
  - When all the two parties have received the commitment, P_A opens the commitment C_1. P_B and P_C obtain P_A's h_1.
  - P_B and P_C repeat the same process as P_A does. At the end, all the parties have the public key set $\{h_1, h_2, h_3\}$. The threshold encryption's public key h is computed as

$$h = \prod_1^3 h_i = \prod_1^3 g^{x_i} = g^{\sum_i x_i} = g^x.$$

    Here, we let $x = \sum_i x_i$.

  Note that all parties know the public key, but they cannot find the secret key $x = \sum_i x_i$ unless they all work together.
- **Signature Exchange**. In this phase, each $P_i, i \in \{A, B, C\}$ computes a signature σ_i on the contract M. Then, he runs the verifiable encrypted signature scheme described in Section 2.2 to encrypt σ_i. At last, P_i sends the encrypted signature VES_i to other parties. Assume the system has produced two public hash functions, ie, $H : \{0, 1\}^* \to \mathbb{Z}_q^*$ and $H_1 : \mathbb{Z}_p \to \mathbb{Z}_p^*$. P_i generates the verifiable encrypted signature as follows.
  - First, the party P_A randomly chooses $x_1' \in \mathbb{Z}_q^*$ as his private key and computes $y_1 = g^{x_1'}$. h is the public key, which has been generated in phase 1.
  - For a contract $M \in \{0, 1\}^*$, the party P_A randomly chooses three integers r_1, t_{11}, and $t_{12} \in \mathbb{Z}_q^*$. Compute the signature $\sigma_1 = H(M)^{x_1'}$, $a_1 = y_1^{r_1}, b_1 = h^{r_1}, c_1 = \sigma_1 b_1^{x_1'}, r_{11} = g^{t_{11}}, r_{12} = (H(M)b)^{t_{11}}, r_{13} = y_1^{t_{12}}, r_{14} = h^{t_{12}}, h_{11} = H_1(H(M), g, y_1, h, r_{11}, r_{12}, r_{13}, r_{14})$, $s_{11} = t_{11} - h_{11}x_1'$, and $s_{12} = t_{12} - h_{11}r_1$. The verifiable encrypted signature of the contract M is $VES_1 = (a_1, b_1, c_1, h_{11}, s_{11}, s_{12})$.
  - After receiving the the contract M's verifiable encrypted signature $VES_1 = (a_1, b_1, c_1, h_{11}, s_{11}, s_{12})$, the recipient $P_j, j \in \{B, C\}$ computes $r_{11}' = y_1^{h_{11}} g^{s_{11}}, r_{12}' = c_1^{h_{11}} (H(M)b_1)^{s_{11}}, r_{13}' = a_1^{h_{11}} y_1^{s_{12}}, r_{14}' = b_1^{h_{11}} h^{s_{12}}$, and $h_{11}' = H_1(H(M), g, y_1, h, r_{11}', r_{12}', r_{13}', r_{14}')$. If $h_{11} = h_{11}'$, P_j accepts the encrypted signature. Otherwise, he aborts the protocol.
  - $P_j, j \in \{B, C\}$ repeat the same process as P_A does. At the end, each $P_i, i \in \{A, B, C\}$ receives the verifiable encryption signature sets VES_i, $i \in \{1, 2, 3\}$.

  If anything goes wrong up to the time t_1, ie, other participant does not receive the verifiable encryption VES_i, he can abort the protocol.

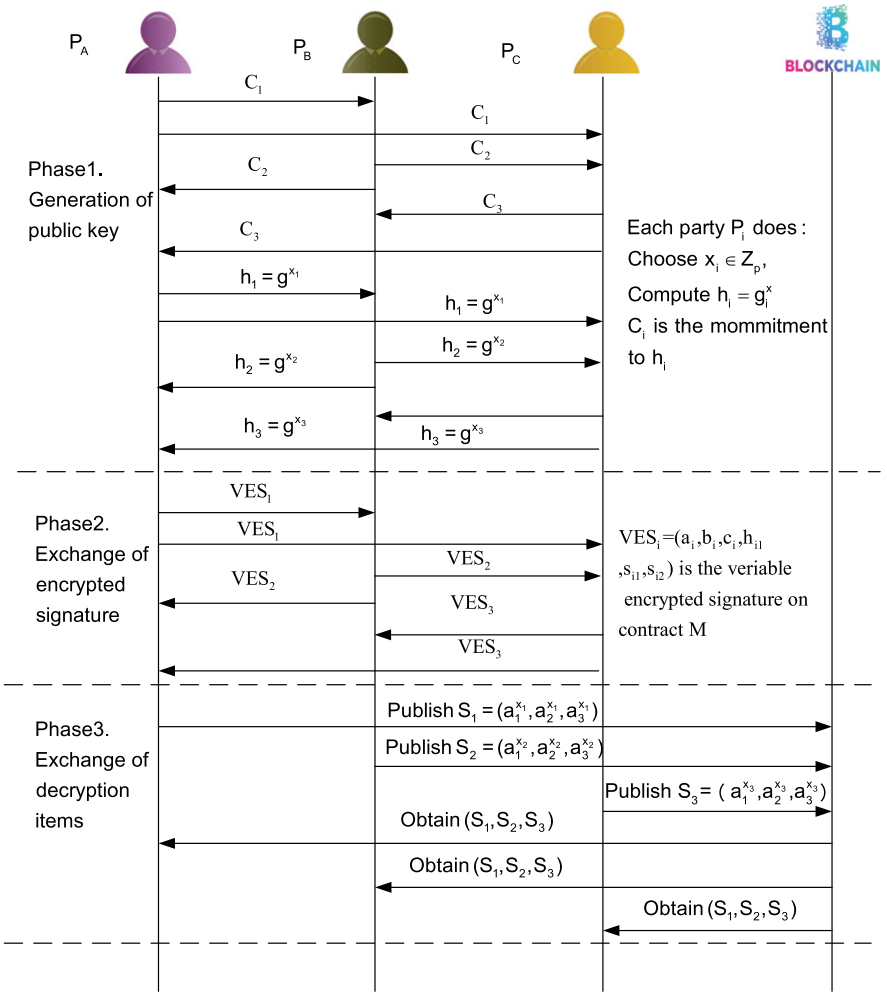

FIGURE 3 The process of our protocol

- **Shares Exchange**. In this phase, each party will send the decryption shares to other parties to decrypt each VES_i. Here, we will use the special nature of Bitcoin to guarantee the fairness of the exchange. If a dishonest party aborts after receiving all shares, he will be monetarily penalized. First, each P_i should make a deposit transaction via Bitcoin system to commit that he will reveal the share $S_i = (a_1^{x_i}, a_2^{x_i}, a_3^{x_i})$. If he does not reveal the secret share S_i by deadline, he will be punished and the other parties will be compensated. After all parties make the deposit, they reveal their secret share S_i via a claim transaction step by step.

  - **Deposit.** This process begins when each party receives all verifiable encryptions signatures VES_i from the others. If anything goes wrong or the protocol exchange has been aborted, it cannot be run. The process of deposit protocol can be described as Andrychowicz's construction[11]

$$P_A \xrightarrow[q,\tau_3]{S_1 \wedge S_2 \wedge S_3} P_C \tag{2}$$

$$P_B \xrightarrow[q,\tau_3]{S_1 \wedge S_2 \wedge S_3} P_C \tag{3}$$

$$P_C \xrightarrow[2q,\tau_2]{S_1 \wedge S_2} P_B \tag{4}$$

$$P_B \xrightarrow[q,\tau_1]{S_1} P_A. \tag{5}$$

  It can be divided into two steps. Formulas (1) and (2) are the first step. That is, parties P_A and P_B make a deposit of q coins to recipient P_C at the same time. This deposit transaction can be claimed by P_C only if he can produce the secret value S_1, S_2, S_3 in round τ_3. As the second step, party P_C makes a deposit of $2q$ coins to recipient P_B, which can be claimed by P_B when he produces S_1, S_2 in round τ_2. Party P_B makes a deposit of q coins to recipient P_A, which can be claimed by P_A when he produces S_1 in round τ_1. Here, $\tau_3 < \tau_2 < \tau_1$. The details are described in the following.

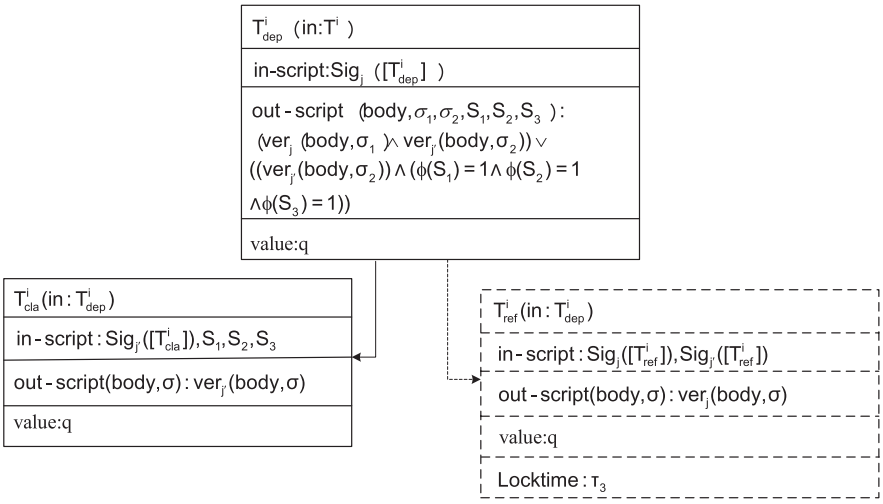

FIGURE 4 Description of formula (1) and (2) in Bitcoin syntax. Here, $i = 1, 2$ and $j' = C$. When $i = 1, j = A$. When $i = 1, j = B$

- For $i \in \{A, B\}$, P_i does the following.

  1. P_A prepares an unredeemed transaction T^1 that can be redeemed with a key known only to P_A. Then, P_A builds a new transaction T^1_{dep} with q coins according to some condition described in Figure 4. Moreover, he keeps it private and creates another transaction T^1_{ref}, which has a locked time τ_3.

  2. P_A sends the body $[T^1_{ref}]$ to P_C and asks P_C to sign. P_C signs $[T^1_{ref}]$ and sends back the signed transaction to P_A. If the signed $[T^1_{ref}]$ does not reach P_A by deadline, P_A aborts. After receiving the signed $[T^1_{ref}]$, P_A posts the transaction T^1_{dep} to the ledger. This shows that P_A can get his deposit back when P_C is dishonest.

  3. P_B makes the deposit transaction T^2_{dep} at the same time. The process is the same as the above step. In order to get a better understanding of this step, Figure 4 shows this step in Bitcoin syntax.

- This is the second step. This step begins after the transactions $T^i_{dep}, i \in \{1, 2\}$ appear on the ledger. For $k = C$ to B, P_k does the following.

  1. The party P_C prepares an unredeemed transaction T^3 that can be redeemed with a key known only to P_C. Then, P_C builds a new transaction T^3_{dep} with $2q$ coins according to some conditions, which are different from Step 1. P_C creates another transaction T^3_{ref}, which has a locked time τ_2. P_C sends the body $[T^3_{ref}]$ to P_B and asks P_B to sign. P_B signs $[T^3_{ref}]$ and sends back the signed transaction to P_C. If the signed $[T^3_{ref}]$ does not reach P_C by deadline, then P_C aborts. After receiving the signed $[T^3_{ref}]$, P_C posts the transaction T^3_{dep} to the ledger. Figure 5 shows the details of this step.

  2. Party P_B also prepares another unredeemed transaction $T^{2'}$ and builds a new transaction $T^{2'}_{dep}$ with q coins. Then, he repeats the same process as P_C does. P_B posts the transaction $T^{2'}_{dep}$ to the ledger. This step is illustrated in Figure 6. If anything goes wrong up to the time t_2, ie, other participant does not make deposit, the protocol can be aborted.

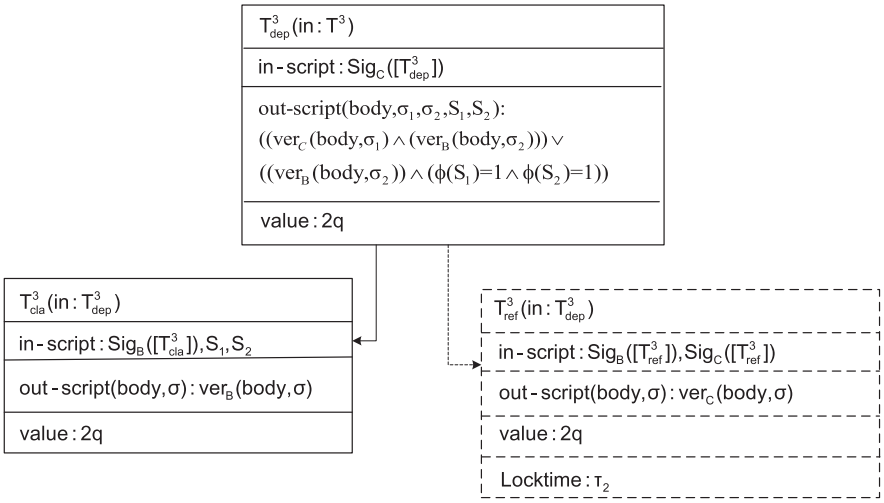

FIGURE 5 Description of formula (3) in Bitcoin syntax

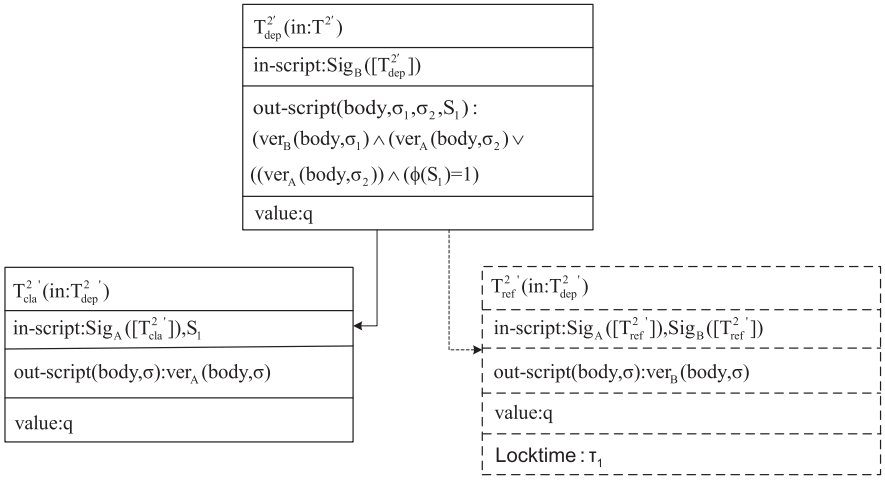

FIGURE 6 Description of formula (4) in Bitcoin syntax

- **Claim.** Wait until all parties have made deposit, all parties begin to make claim transaction in the reverse direction.
  - ∗ Party P_A first makes claim transaction as described next. He prepares a transaction $T^{2'}_{cla}$ and posts it to the ledge. When the transaction appears on the ledger, P_B and P_C collect S_1. If the transaction $T^{2'}_{cla}$ does not appear on the ledger, P_B gets his deposit back wait by round $\tau_1 + 1$.
  - ∗ Then, the party P_B uses S_1 and S_2 to create the transaction T^3_{cla} to claim his deposit. When the transaction appears on the ledger, P_A and P_C collect S_2. If the transaction T^3_{cla} does not appear on the ledger, P_B gets his deposit back wait by round $\tau_2 + 1$.
  - ∗ At last, the party P_C uses S_1 and S_2 to create the transaction T^1_{cla} and T^2_{cla} to claim his deposit. When the transaction appears on the ledger, P_A and P_B collect S_3. If the transaction T^1_{cla} and T^2_{cla} does not appear on the ledger, P_A and P_B get their deposit back wait by round $\tau_3 + 1$. P_A posts the transaction T^1_{ref} and gets his deposit back wait by round $\tau_3 + 1$.

- **Decryption**. In this phase, P_i can decrypt each VES_i after receiving all the necessary values. He can get the other parties signature on the contract M. The decryption for σ_i is as follows:

$$\sigma_i = c_i / \prod_1^3 a_i^{x_k} = \sigma_i b_i^{x'_i} / a_i^{\sum_1^3 x_k} = \sigma_i b_i^{x'_i} / a_i^x = \sigma_i a_i^x / a_i^x.$$

Remark 1. In our proposed protocol, the idea of phase 1 and phase 2 is inspired by [22]. The verifiable encryption signature scheme used in our proposed protocol is slightly modified from Shao and Gao's[42] scheme. We use the ElGamal threshold encryption scheme to encrypt the signature on the contract M. In phase 2, every party should verify the validity of VES_i by the equation $h_{i1} = h'_{i1}$. The verification equation is proof of knowledge of the equality of discrete logarithm (DL),[42,44] ie, $DL_{(H(M)b_i)}(c_i) = DL_g(y_i) = x'_i$ and $DL_{(y_i)}(a_i) = DL_g(b_i)$. The probability of $DL_{(H(M)b_i)}(c_i) \neq DL_g(y_i)$ is at most $1/p$, and the same is $DL_{(y_i)}(a_i) \neq DL_g(b_i)$.[45] From a_i, b_i and c_i, we can make sure that $a_i^x = b_i^{x'_i}$. This equation implies that if all the parties work together, they can extract a valid σ_i.

Remark 2. In phase 3, in order to realize our fair protocol, we require the script of transaction to contain operation of the equality of discrete logarithm. That is, denote a function $\phi(.)$. We define the function $\phi(.)$ as follows.

- $\phi(.)$: Assume the transaction T contains this function. Input t_1, t_2, t_3, and t_4, where t_i is a pair (t_{i1}, t_{i2}). If the following equation is true, then the output $\phi(t_1, t_2, t_3, t_4) = 1$

$$DL_{t_{11}}(t_{12}) = DL_{t_{21}}(t_{22}) = DL_{t_{31}}(t_{32}) = DL_{t_{41}}(t_{42}).$$

For example, the input script contains the encryption shares $S_i = \{(a_1, a_1^{x_i}), (a_2, a_2^{x_i}), (a_3, a_3^{x_i}), (g, h_i)\}$; the node on the blockchain can verify whether S_i satisfies the above equation. If it satisfies, then $\phi(S_i) = 1$. On the other hand, unfortunately, the current Bitcoin system does not support the discrete logarithm operation. However, we can use the platform of Ethereum or Hyperledger to implement and simulate our protocol. In order to interpret our protocol more clearly, we adopt the form of Bitcoin transaction to describe our sub-protocol and we show the main steps of the exchange of the decryption shares in Figures 4, 5, and 6.

4 | SECURITY AND PERFORMANCE ANALYSIS

4.1 | Security analysis

Theorem 1. *The proposed three-party contract signing protocol satisfies the property of completeness.*

Proof. If all parties are honest, they will successfully run the Signature Exchange protocol and obtain the verifiable encrypted signature of all parties. After protocol finishing, all the parties can obtain the decryption shares to decrypt the encrypted signature and obtain the valid signed contract. None of the parties have to pay any penalty. □

Theorem 2. *The proposed three-party contract signing protocol satisfies the property of fairness.*

Proof. In our protocol, the property of fairness is guaranteed by that (1) all parties obtain valid signed contract or they only obtain the others' verifiable signature on the contract. (2) If a dishonest party does not abide by the protocol, then the honest party is compensated and the dishonest one has to pay a fine.

We prove the first one. Assuming that threshold encryption scheme and the verifiable signature scheme are secure. We assume the party P_i is dishonest. He refused to send his VES_i to the other two parties after receiving the others' in the phase of signature exchange. In this case, the other two parties wait until the deadline and then they can abort the protocol. According to the protocol, the following phase cannot be run. Although P_i has obtained the other parties' VES_j, he did not have the shared key for the ciphertext. Therefore, if the phase 3 does not run, they only obtain the other one's verifiable signature on the contract.

Then, we prove the second one. At the end of the protocol, all honest parties will receive all the shares S_i without paying any fine. The honest party who publishes his share but does not obtain all the decryption shares is compensated by q coins. If each honest party who does not publish his own shares can get his deposit back, then no corrupt party obtains any shares. There are three cases.

- Party P_A does not publish his share S_1. It means that the Claim phase has not been executed. No party obtains any share.
- Party P_B does not publish his share S_2. There are two cases. First, P_A has made claim transaction and P_B receives the share S_1. In this case, P_B only needs to publish his own shares S_2 and gets his deposit back. Otherwise, he will pay q coins to P_A. Second, P_A did not publish his shares. In this case, P_B does not make claim transaction and no party obtains S_2.
- Party P_C does not publish his share S_3. In this case, it means that P_C is the first party who obtains all shares. If P_C is dishonest and he aborts the protocol, then P_A and P_B do not obtain P_C's shares. However, both of them get the deposit after time τ_3. Because when P_A published S_1, he gets q coins from P_B, while P_B gets $2q$ coins from P_C when he published S_2. At this point, both of them have been compensated. □

Theorem 3. *The proposed three-party contract signing protocol satisfies the property of timeliness.*

Proof. At any phase of our protocol, there is a deadline, which was agreed on in advance of the protocol. If every party involved is honest, our protocol requires constant rounds. Therefore, we can be sure that the protocol will abort in a concrete time. Moreover, in the deposit and claim phase, every transaction has a limited time. Therefore, we are sure that our protocol satisfies the timeliness requirement. □

Theorem 4. *The proposed three-party contract signing protocol satisfies the property of non-repudiation.*

Proof. First, we uses the ElGamal threshold encryption scheme to encrypt a valid undeniable signature $\sigma_i = H(M)^{x_i'}$. This construction has been slightly modified from Shao and Gao's[42] scheme. In phase 2 of our protocol, every party should verify the validity of VES_i. That is, they should verify if the equation $h_{i1} = h_{i1}'$ is satisfied or not. The verification equation is proof of knowledge of the equality of discrete logarithm (DL),[42,44] ie, $DL_{(H(M)b_i)}(c_i) = DL_g(y_i) = x_i'$ and $DL_{(y_i)}(a_i) = DL_g(b_i)$. It can guarantee that the signature can be extracted by the other parties. The probability of $DL_{(H(M)b_i)}(c_i) \neq DL_g(y_i)$ is at most $1/p$, and the same is $DL_{(y_i)}(a_i) \neq DL_g(b_i)$.[45] From a_i, b_i, and c_i, we can make sure that $a_i^x = b_i^{x_i'}$. This equation implies that the valid signature σ_i can be extracted from any valid VES_i with the key x.

On the other hand, the ElGamal threshold encryption scheme is used to generate the key x. That is, all parties work together to generate the secret key $x = \sum_i x_i$. They only know the public key, but they cannot find the secret key $x = \sum_i x_i$ unless they all work together. Assuming that the ElGamal threshold encryption scheme is a secure threshold encryption scheme and the discrete logarithm problem is hard, our proposed protocol is a secure protocol. In phase 3, every party exchanges the decryption shares. After every party obtaining the decryption shares, the undeniable signature can be extracted by each party through combining all decryption shares. Therefore, the signer cannot generate any valid VES_i such that the other party cannot extract the signature from the other way. The undeniable cannot be denied by the signer through any disavowal protocol. □

Theorem 5. *The proposed three-party contract signing protocol satisfies the property of confidentiality.*

Proof. In our protocol, the process of signature exchange is secure. From our construction, the content of the contract has been encrypted and only the participants involved in the protocol are allowed to know the content of the contract in the signature exchange phase. On the other hand, the fairness is guaranteed by exchanging the decryption shares in the Bitcoin network. The parties should broadcast the transactions containing the decryption items to the network, and the nodes in the network must verify the correctness of the shares. Although the nodes only learn the decryption shares, they cannot decrypt the encrypted signature since he never gets the items. Therefore, we can say that the confidentiality is satisfied. □

TABLE 1 Comparison with the previous scheme

Scheme	Message Complexity	Communication Complexity	TTP
Scheme [21]	$O(n^3)$	$O(n^2)$	Yes
Scheme [23]	$O(n^3)$	$O(n)$	Yes
Scheme [26]	$O(n^3)$	$O(n)$	Yes
Scheme [27]	$O(n^2)$	$O(n)$	Yes
Our scheme	$O(n^2)$	$O(1)$	No

TABLE 2 Comparison among schemes

	Scheme				
Properties	Scheme [21]	Scheme [23]	Scheme [26]	Scheme[27]	Our scheme
Fairness	Weak	No	Yes	Yes	Yes
Timeliness	Yes	Yes	Yes	Yes	Yes
Non-repudiation	No	Yes	Yes	Yes	Yes
Confidentiality	No	Yes	No	Yes	Yes

4.2 | Performance analysis

In our proposed protocol, each party P_i sends one verifiable encryption to the other two parties, and, in the end, he broadcasts the transactions containing the decryption shares to the blockchain. Therefore, the messages complexity in the shares exchange phase is $O(n)$. If there are n parties, the total message complexity is $O(n^2)$. We can see that $n = 3$ in our protocol. Therefore, the total message complexity is 9.

Table 1 presents the comparison between our protocol and the previous works. n is the numbers of parties. First, we can see, from Table 1, our proposed protocol is based on blockchain without a TTP. All of the others need a TTP to resolve the dispute. The existence of TTP will become a bottleneck, since it can be a single point of failure or suffer from external or internal attack. Second, scheme [21] is the first solutions for fair optimistic multi-party contract signing protocol. In their protocol, the total number of messages needed are $O(n^3)$ and the communication complexity is $O(n^2)$ rounds. Here, n is the number of participants. When $n = 3$, the total number of messages needed are 27. Garay and MacKenzie's[21] solution was very inefficient. Scheme [23] proposed a more efficient one. Its efficiency depended on the numbers of the dishonest parties. We consider the worst case that the number of dishonest parties reached $n-1$; it needs $(n+1)n(n+1)(O(n^3))$ messages and $n+1$ rounds. The round complexity was decreased by scheme [26] to $O(n)$ but also needs $O(n^3)$ messages. When $n = 3$, the total number of messages needed are 24. Scheme [27] introduced a linear fair multi-party protocol based on the optimistic model, which needs only $O(n^2)$ messages. When $n = 3$, the total number of messages needed are 18. This is the same as our protocol, but the round complexity was not constant. In summary, our proposed protocol is more efficient than the existing ones.

As seen in Table 2, scheme [21] has been proved that it did not satisfy the fairness property when $n > 4$.[24] Scheme [23] assumed that TTP should know the number of the dishonest parties and all honest parties could not abort the protocol. In that case, if some honest parties are to abort, fairness could not be guaranteed for other honest ones, and therefore, scheme [23] was weak in fairness. In summary, our proposed scheme can satisfy all of the security properties.

4.2.1 | Communication cost in the blockchain

In this section, we will analyze the communication cost of each party in this phase. Party P_A should broadcoast 2 transactions to the blockchain. He also sends the body of transactions to P_C and 1 signature to P_B, so the total communication cost of P_A is 2 unicast messages and 2 broadcast messages. Then, party P_B should broadcoast 4 transactions to the blockchain. Moreover, he also sends 2 bodies of transactions to P_A and P_C, respectively. He also sends 1 signature to P_C, so the total communication cost of P_B is 3 unicast messages and 4 broadcast messages. At last, the party P_C should broadcoast 2 transactions to the blockchain. Moreover, he also sends the body of transactions to P_B and 2 signature to P_A and P_C, so the total communication cost of P_B is 3 unicast messages and 2 broadcast messages. The communication cost of each party has been shown in Table 3.

TABLE 3 The communication cost of each parties in phase 3

Party	Unicast	Broadcast
Party P_A	2	2
Party P_B	3	4
Party P_C	3	2

4.3 | Simulation

In this section, we provide the experimental simulation of the proposed contract signing protocol. We only simulate the realization of the exchange decryption shares. Considering that the transactions in Figures 4, 5, and 6 are the same in the performance, we only simulate the formula (4) (ie, Figure 6). We have tested these on a PC with Intel(R) Core(TM) i3-3240 CPU, which clocks at 3.40 GHz, 4GB memory and Ubuntu 14.04 operating system.

To realize the smart contract functionality for facilitating online contractual agreements, we use the platform of Ethereum to implement and simulate our protocol. Ethereum is the best development platform for blockchain. For achieving our performance, we use Truffle to test framework and asset pipeline for Ethereum and adopt solidity language to write smart contracts to realize the transaction in our protocol. Then, we run the contract on the privacy blockchain. The smart contract we build for our protocol is as the flow chart described in Figure 7.

In the experiment, since the transactions in Figures 4, 5, and 6 are all the same, we only simulate the transaction in the sub-figure of Figure 6. Recall that this phase contains two parties, ie, P_A and P_B. In the deposit, P_B prepares a transaction $T_{dep}^{2'}$ with q coins and can be redeemed according to the conditions that "(P_A's signature AND P_B's signature) OR (S_1 for which $\phi(S_1) = 1$ AND P_A's signature)." Then, P_A makes claim transaction publish S_1 and gets the q coins. We simulate the process of deposit and claim transaction by smart contract on the privacy blockchain. First, we use the solidity language to write the contracts we needed. Then, we simulate the Ethereum environment by *TestRPC* in the local memory. At last, we debug each function of the smart contract in the frame of truffle. The flow chart of our experiment in realizing the deposit and claim transaction is described in Figure 8.

In our smart contract, the script should contain the verification of the equality of discrete logarithm, ie, compute the value $\phi(.)$. Due to the limitation of Ethereum, an extra processing is adopted to solve the problem in the flow chart. This verification process is realized in the local memory. After this processing, the smart contract can obtain the certifiable parameters from local computation.

We simulate the process of deposit and claim transaction by smart contract on the privacy blockchain. We denote by T as the total time cost of realizing one transaction on the privacy blockchain. T is made up of 5 parties. T_1, T_2, T_4 is the communication time that interaction between participant and smart contract; contract requests local computation and the local computation responds to the contract. T_3 is the time that the local memory verifies the equality of discrete logarithm, ie, it will output the value of $\phi(.)$. T_5 is the time that smart contract verify the certifiable parameters. Our simulation run on privacy blockchain, so the time cost T_1, T_2, T_4, T_5 can be negligible. We only analyze the time cost of T_3. We adopt Python to realize the function $\phi(.)$ in Ubuntu 14.04. As defined before, the functionality of $\phi(.)$ is that input t_1, t_2, t_3, t_4, where t_i is a pair (t_{i1}, t_{i2}). If the following equation is true, then the output $\phi(t_1, t_2, t_3, t_4) = 1$

$$DL_{t_{11}}(t_{12}) = DL_{t_{21}}(t_{22}) = DL_{t_{31}}(t_{32}) = DL_{t_{41}}(t_{42}).$$

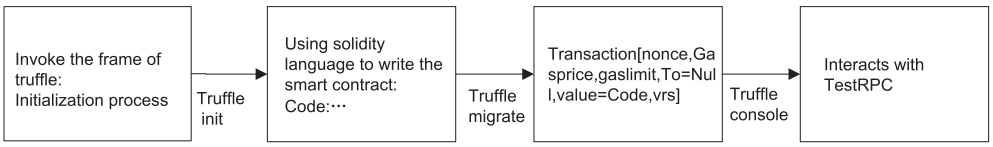

FIGURE 7 The construction of smart contract

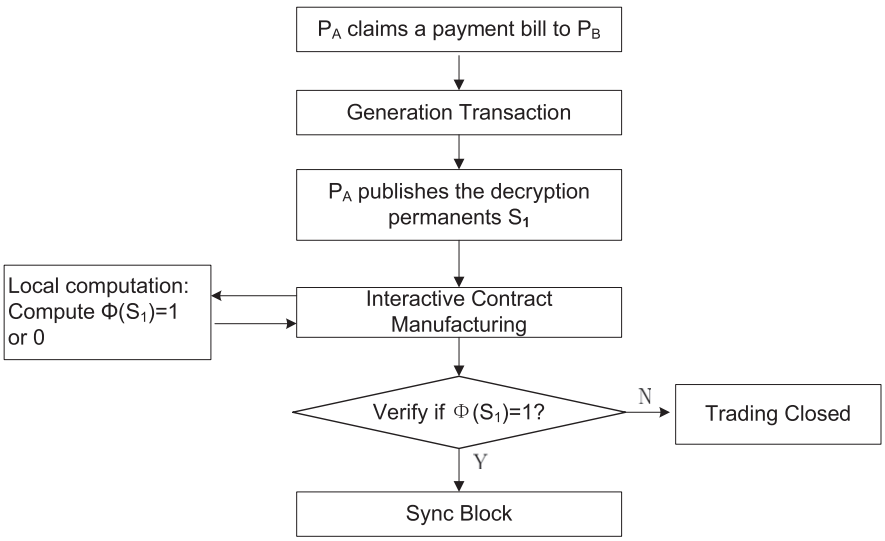

FIGURE 8 The flow chart of experiment

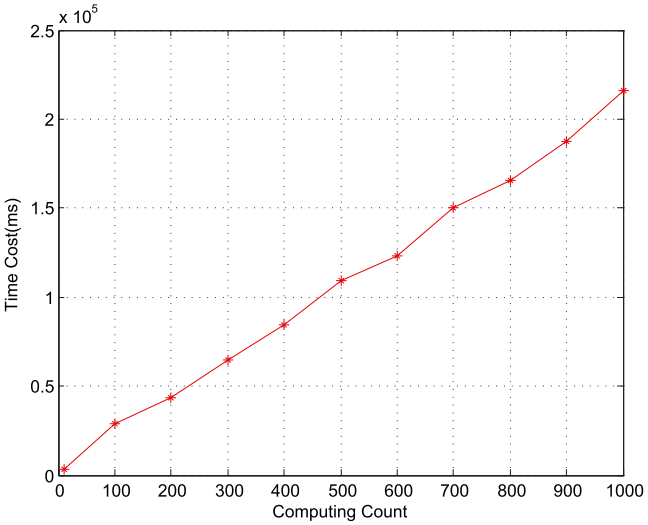

FIGURE 9 Time cost of shares verification on privacy blockchain

TABLE 4 The comparison of the computation overhead

Scheme	Scheme [22]	Our Proposed Scheme
Model	optimal model	blockchain model
Signature Exchange	$1S + 2E + 2V$	$1S + 2E + 2V$
Shares Exchange	$1VE + 2V + 2VD$	$2T$

Due to the randomness of the parameter generation, we run the program more than once to test the computation time cost. Figure 9 shows the time cost of running the program n (from 100 to 1000) times. We can estimate the time cost of $T_3 \approx 0.2s$.

We compare the computational overhead of each party between our scheme and scheme [22] in Table 4. We denote by S a signature on contract, by VE a verifiable encryption, by V the verification of VE, by VD the verification of discrete logarithm, and by T a transaction prepared by parties. We provide the time costs simulation for schemes [22] and our scheme in Figures 10, 11, and 12. The time cost of each party in the phase of Signature Exchange and Shares Exchange for two schemes are shown in Figures 11 and 12, respectively. From the two figures, we can see that the time cost of each party in Signature Exchange and Shares Exchange of our scheme is small compared with scheme [22]. The main reason is that we use the blockchain to finish the signature exchange in our scheme. In our protocol, there is no trust third party. Therefore, the computation cost of the

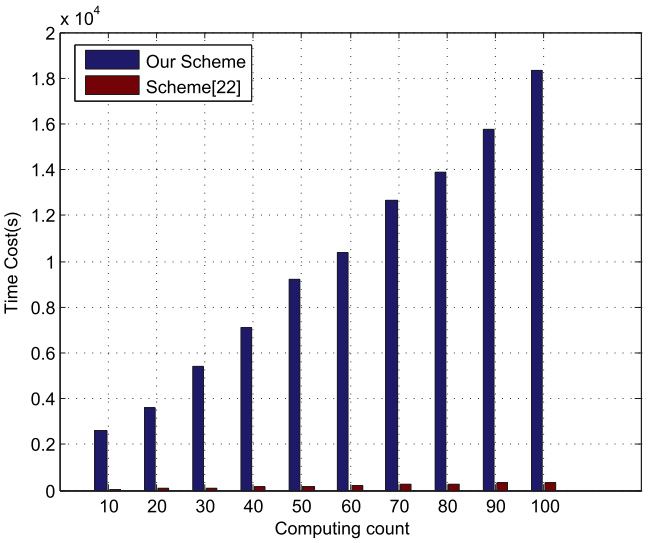

FIGURE 10 Time cost in Shares Exchange

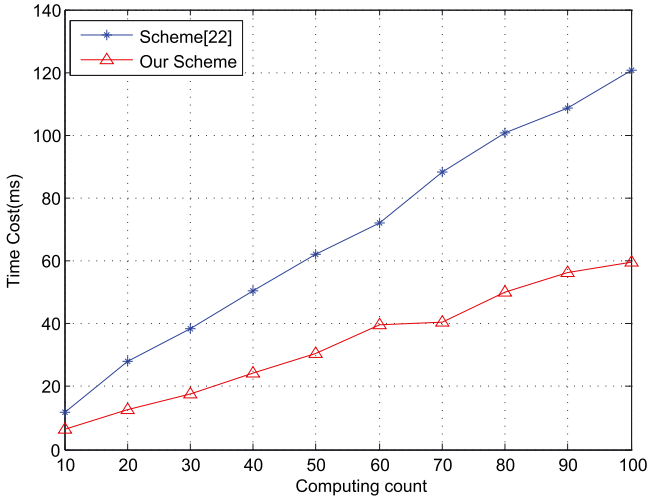

FIGURE 11 Time cost comparison of each in Shares Exchange

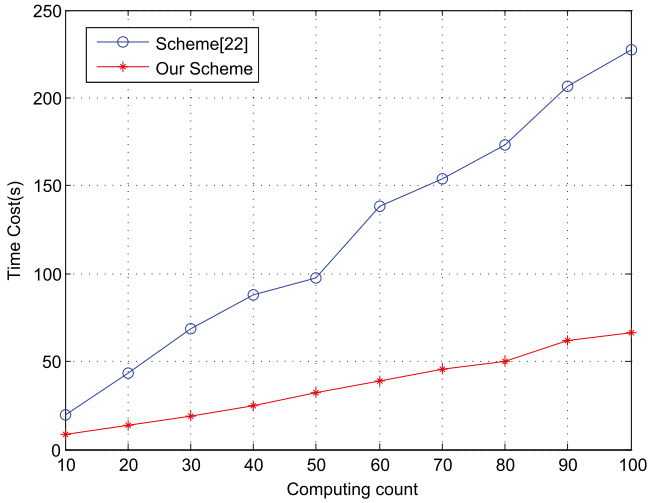

FIGURE 12 Time cost comparison of each in Signature Exchange

participants side in our scheme is smaller than scheme [22]. Figure 10 shows the total time cost simulation of the Shares Exchange phase. The simulation results in Figure 10 indicate that our scheme takes more time than scheme [22]. The main reason is the verification should be finished in the public chain. In the process of the verification, we should wait the nodes in the blockchain to affirm. However, we only simulate the case that every party in scheme [22] is all honest. In the case of existing dishonest parties, the honest one needs to contact the trusted third party to solve the dispute. From that perspective, the time cost of scheme [22] is similar to our scheme. However, our scheme without a trusted third party is more secure and most suitable for real-world applications.

5 | CONCLUSION AND FUTURE WORK

Fog computing emerges as a new application schema that can provide flexible services at the edge of network. In the decentralized fog surroundings, how to sign a service contract between the participants through a network fairly has become more difficult. Multi-party contract signing is a practical applications of electronic data exchange. A crucial property for a contract signing protocol is fairness. The existing solutions to the problem of fairness need a TTP to involve in the dispute. These solutions are not suitable for fog computing. Blockchain, as the underlying technology of Bitcoin, provides a solution to solve the trust problem of the third party. In this paper, we have proposed a fair three-party contract signing protocol based on blockchain for fog computing. The proposed construction makes use of the verifiable encrypted signature and the blockchain to accomplish the fair exchange. As result, a dishonest party will be monetarily penalized as it aborts after receiving the current output. Meanwhile, it can protect the privacy of the contract content. In future work, we will focus on *n* party contract signing protocol based on blockchain.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under grant 61572382, Key Project of Natural Science Basic Research Plan in Shaanxi Province of China under grant 2016JZ021, and China 111 Project under grant B16037.

ORCID

Hui Huang http://orcid.org/0000-0001-6387-5734

REFERENCES

1. Stojmenovic I, Wen S, Huang X, Luan H. An overview of fog computing and its security issues. *Concurrency Computat Pract Exper*. 2016;28(10):2991-3005.
2. Chen X, Li J, Weng J, Ma J, Lou W. Verifiable computation over large database with incremental updates. *IEEE Trans Comput*. 2016;65(10):3184-3195.
3. Chen X, Li J, Huang X, Ma J, Lou W. New publicly verifiable databases with efficient updates. *IEEE Trans Dependable Sec Comput*. 2015;12(5):546-556.
4. Chen X, Li J, Ma J, Tang Q, Lou W. New algorithms for secure outsourcing of modular exponentiations. *IEEE Trans Parallel Distrib Syst*. 2014;25(9):2386-2396.
5. Li J, Zhang Y, Chen X, Xiang Y. Secure attribute-based data sharing for resource-limited users in cloud computing. *Comput Secur*. 2018;72:1-12.
6. Li P, Li J, Huang Z, Gao CZ, Chen WB, Chen K. Privacy-preserving outsourced classification in cloud computing. *Cluster Comput*. 2017;2017:1-10.
7. Li J, Huang X, Li J, Chen X, Xiang Y. Securely outsourcing attribute-based encryption with checkability. *IEEE Trans Parallel Distrib Syst*. 2014;25(8):2201-2210.
8. Yi S, Hao Z, Qin Z, Li Q. Fog computing: platform and applications. Paper presented at: Proceedings of the 3rd IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb'15); 2015; Washington, DC.
9. Nakamoto S. Bitcoin: A peer-to-peer electronic cash system; 2008.
10. Huang H, Chen X, Wu Q, Huang X, Shen J. Bitcoin-based fair payments for outsourcing computations of fog devices. *Futur Gener Comput Syst*. 2018;78:850-858.
11. Andrychowicz M, Dziembowski S, Malinowski D, Mazurek L. Secure multiparty computations on Bitcoin. Paper presented at: Proceedings of 2014 Symposium on Security and Privacy (SP'14); 2014; San Jose, CA.
12. Andrychowicz M, Dziembowski S, Malinowski D, Mazurek L. Secure multiparty computations on Bitcoin. *Commun ACM*. 2016;59(4):76-84.
13. Huang H, Li KC, Chen X. A fair three-party contract signing protocol based on Blockchain. Paper presented at: Proceedings of the 9th International Symposium on Cyberspace Safety and Security (CSS'17); 2017; Xi'an, China.
14. Bonomi F, Milito RA, Zhu J, Addepalli S. Fog computing and its role in the internet of things. Paper presented at: Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing (MCC@SIGCOMM'12); 2012; Helsinki, Finland.
15. Hong K, Lillethun DJ, Ramachandran U, Ottenwälder B, Koldehofe B. Opportunistic spatio-temporal event processing for mobile situation awareness. Paper presented at: Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (DEBS '13); 2013; Arlington, TX.
16. Madsen H, Burtschy B, Albeanu G, Popentiu-Vladicescu FL. Reliability in the utility computing era: Towards reliable fog computing. Paper presented at: Proceedings of the 20th International Conference on Systems, Signals and Image Processing (IWSSIP'13); 2013; Bucharest, Romania.
17. Nishio T, Shinkuma R, Takahashi T, Mandayam NB. Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud. Paper presented at: Proceedings of the 1st International Workshop on Mobile Cloud Computing and Networking (MobileCloud'13); 2013; Bangalore, India.
18. Ottenwälder B, Koldehofe B, Rothermel K, Ramachandran U. MigCEP: Operator migration for mobility driven distributed complex event processing. Paper presented at: Proceedings of the 7th ACM International Conference on Distributed Event-based Systems (DEBS'13); 2013; Arlington, TX.
19. Vaquero LM, Rodero-Merino L. Finding your way in the fog: towards a comprehensive definition of fog computing. *Comput Commun Rev*. 2014;44(5):27-32.
20. Stojmenovic I, Wen S. The fog computing paradigm: Scenarios and security issues. Paper presented at: 2014 Federated Conference on Computer Science and Information Systems (FedCSIS'14); 2014; Warsaw, Poland.
21. Garay JA, MacKenzie P. Abuse-free multi-party contract signing. Paper presented at: 13th International Symposium on Distributed Computing (DISC'99); 1999; Bratislava, Slovakia.
22. Kılınç H, Küpçü A. Optimally efficient multi-party fair exchange and fair secure multi-party computation. Paper presented at: Cryptographer's Track at the RSA Conference 2015 (CT-RSA'15); 2015; San Francisco, CA.
23. Baum-Waidner B, Waidner M. Round-optimal and abuse-free optimistic multi-party contract signing. Paper presented at: 27th International Colloquium on Automata, Languages, and Programming (ICALP'00); 2000; Geneva, Switzerland.
24. Chadha R, Kremer S, Scedrov A. Formal analysis of multiparty contract signing. *J Autom Reason*. 2006;36(1-2):39-83.
25. Chen X, Zhang F, Tian H, et al. Three-round abuse-free optimistic contract signing with everlasting secrecy. Paper presented at: 14th International Conference on Financial Cryptography and Data Security; 2010; Tenerife, Spain.
26. Mukhamedov A, Ryan MD. Fair multi-party contract signing using private contract signatures. *Inf Comput*. 2008;206(2-4):272-290.
27. Mauw S, Radomirovic S, Dashti MT. Minimal message complexity of asynchronous multi-party contract signing. Paper presented at: 22nd IEEE Computer Security Foundations Symposium (CSF'09); 2009; New York, NY.
28. Goldreich O. A simple protocol for signing contracts. Paper presented at: Advances in Cryptology (CRYPTO'83); 1983; Santa Barbara, CA.
29. Ben-Or M, Goldreich O, Micali S, Rivest RL. A fair protocol for signing contracts. *IEEE Trans Inf Theory*. 1990;36(1):40-46.

30. Asokan N, Schunter M, Waidner M. Optimistic protocols for fair exchange. Paper presented at: Proceedings of the 4th ACM Conference on Computer and Communications Security (CCS'97); 1997; Zurich, Switzerland.
31. Asokan N, Shoup V, Waidner M. Optimistic fair exchange of digital signatures. Paper presented at: Advances in Cryptology (EUROCRYPT); 1998; Espoo, Finland.
32. Ateniese G. Efficient verifiable encryption (and fair exchange) of digital signatures. Paper presented at: Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS'99); 1999; Singapore.
33. Narayanan A, Bonneau J, Felten EW, Miller A, Goldfeder S. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton, NJ: Princeton University Press; 2016.
34. Swan M. *Blockchain: Blueprint for a New Economy*. Sebastopol, CA: O'Reilly Media Inc; 2015.
35. Zhao JL, Fan S, Yan J. Overview of business innovations and research opportunities in blockchain and introduction to the special issue. *Financ Innov*. 2016;2(1):28.
36. Tschorsch F, Scheuermann B. Bitcoin and beyond: a technical survey on decentralized digital currencies. *IEEE Commun Surveys Tuts*. 2016;18(3):2084-2123.
37. Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things. *IEEE Access*. 2016;4:2292-2303.
38. Pedersen TP. A threshold cryptosystem without a trusted party (extended abstract). Paper presented at: Advances in Cryptology (EUROCRYPT'91); 1991; Brighton, UK.
39. Asokan N, Shoup V, Waidner M. Optimistic fair exchange of digital signatures. *IEEE J Sel Areas Commun*. 2000;18(4):593-610.
40. Boneh D, Gentry C, Lynn B, Shacham H. Aggregate and verifiably encrypted signatures from bilinear maps. Paper presented at: Advances in Cryptology (EUROCRYPT'03); 2003; Warsaw, Poland.
41. Boneh D, Lynn B, Shacham H. Short signatures from the Weil pairing. *J Cryptol*. 2004;17(4):297-319.
42. Shao Z, Gao Y. Practical verifiably encrypted signatures based on discrete logarithms. *Secur Commun Netw*. 2016;9(18):5996-6003.
43. Staff E. Blockchains: The great chain of being sure about things. *The Economist*. 2016:18. https://www.economist.com/news/briefing/21677228-technology-behind-bitcoin-lets-people-who-do-not-know-or-trust-each-other-build-dependable
44. Chaum D, Pedersen TP. Wallet databases with observers. Paper presented at: Advances in Cryptology (CRYPTO'92); 1992; Santa Barbara, CA.
45. Goh EJ, Jarecki S. A signature scheme as secure as the Diffie-Hellman problem. Paper presented at: Advances in Cryptology (EUROCRYPT'03); Warsaw, Poland.

How to cite this article: Huang H, Li K-C, Chen X. Blockchain-based fair three-party contract signing protocol for fog computing. *Concurrency Computat Pract Exper*. 2018;e4469. https://doi.org/10.1002/cpe.4469