Future Generation Computer Systems 78 (2018) 850–858

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Bitcoin-based fair payments for outsourcing computations of fog devices

Hui Huang [a], Xiaofeng Chen [a,b,*], Qianhong Wu [c], Xinyi Huang [d], Jian Shen [e]

[a] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China
[b] Guangxi Cooperative Innovation Center of cloud computing and Big Data, Guilin University of Electronic Technology, Guilin, China
[c] School of Electronic Information Engineering, Beihang University, Beijing, China
[d] Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou, China
[e] School of Computer and Software, Nanjing University of Information Science & Technology (NUIST), Nanjing, China

HIGHLIGHTS

- We propose a bitcoin-based fair payment protocol for outsourcing computations.
- The commitment-based sampling scheme is used to solve the trust problem in workers.
- Bitcoin script is used to gurantee the fairness between workers and outsourcer.

ARTICLE INFO

Article history:
Received 9 June 2016
Received in revised form
15 November 2016
Accepted 11 December 2016
Available online 16 December 2016

Keywords:
Fog computing
Outsourcing computations
Bitcoin contract
Commitment-based sampling

ABSTRACT

Fog computing can be viewed as an extension of cloud computing that enables transactions and resources at the edge of the network. In the paradigms of fog computing, the fog user (outsourcer) with resource-constraint devices can outsource the distributed computation tasks to the untrusted fog nodes (workers) and pays for them. Recently, plenty of research work has been done on fair payments. However, all existing solutions adopt the traditional e-cash system to generate payment token, which needs a trusted authority (i.e. a bank) to prevent double-spending. The bank will become the bottleneck of the payments system. In this paper, we propose a new fair payment scheme for outsourcing computations based on Bitcoin. Due to the advantages of Bitcoin syntax, the users can transact directly without needing a bank. Besides, the proposed construction can guarantee that no matter how a malicious outsourcer behaves, the honest workers will be paid if he completed the computing tasks.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Fog computing, also called edge computing, enables computing and storage to be realized at the edge of the network. Its prominent advantage is providing new applications and services for various users [1]. Vaquero et al. [2] have defined fog computing as that the storage and processing tasks can be performed by large-scale, ubiquitous and decentralized devices. Fog devices can communicate and potentially cooperate with each other without the intervention of third parties. These devices can be either a resource-constrained fog node built on existing network devices like WiFi access point or home set-top box, or a resource-rich fog node as specific high-end servers with powerful CPU, larger memory and storage. Fog computing is an extension of the cloud computing [3,4] and usually cooperates with cloud computing. As a result, end users, fog nodes and cloud together form a three layer service delivery model, supporting a series of applications such as web content delivery [5] and augmented reality [6].

We consider the scenario that the heavy computation tasks can be most efficiently executed by a large number of fog nodes. That is, the computation tasks are divided into smaller ones and assigned to different fog nodes. Each node finishes the corresponding computation and sends the result to the outsourcer. For simplicity, we denote outsourcer by O and the fog nodes (also called workers) by W. In commercial settings, the worker W will receive remuneration from outsourcer O after accomplishing the whole job before the deadline. Due to the lack

* Corresponding author at: State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China.
E-mail addresses: hhui323@163.com (H. Huang), xfchen@xidian.edu.cn (X. Chen), qianhong.wu@buaa.edu.cn (Q. Wu), xyhuang81@yahoo.com (X. Huang), s_shenjian@126.com (J. Shen).

http://dx.doi.org/10.1016/j.future.2016.12.016
0167-739X/© 2016 Elsevier B.V. All rights reserved.

of the trust between the outsourcer and workers, there are two security problems should be considered. Firstly, O does not trust that the computation tasks will be carried out properly by W. Due to the economic profit, W might not complete the actual computation and just send back a plausible result that only needs less work. Secondly, W lacks trust in O who might be anyone on the Internet and he will try to cheat W. In this manner, W has honestly completed the whole tasks whereas O refused to pay.

To solve the trust problems between O and W, a number of solutions have been presented recently. On one hand, the outsourcer should verify the computation result when he pays the remuneration to the workers. For different computing tasks, there are different solutions for verification. Monrose et al. [7] used computation proofs to detect the worker behaviors. Golle et al. [8] adopted duplicating computations to verify the correctness of computation results, but the disadvantage of his solution is the communication with high complexity. Szajda et al. [9] and Sarmenta et al. [10] presented probabilistic verification mechanisms to detect cheaters, but the cost of computation is high. Later, they proposed a method to decreases the computation costs [11]. Carbunar et al. [12] used a predefined witnesses to guarantee the completion quality of their work. For the computing tasks of "inversion of one-way function", Golle et al. [13] first proposed the concept of ringer to solve the trust problem on workers. Du et al. [14] pointed out that there existed some limitations in the ringer scheme and proposed a novel approach to solve the trust problem, which uses the Merkle hash tree to make commitment for the computed result.

On the other hand, workers should be paid after they has completed the computation task. Carbunar et al. [15] first considered the payment problem in the scenario of outsourcing computation. Based on cut-and-choose protocol and secret sharing, they proposed a fair payments scheme. However, the solution is very inefficient for the actual applications. Later, they improved their work and introduced a new scheme for the payments [16]. But the efficiency of the scheme was not advanced more. It also can be seen as a specific examples of conditional e-payments [17,18]. Recently, Chen et al. [19] first considered the third trust problem that the worker W will not send the computation results to the outsourcer O. They introduced the lazy-and-partially-dishonest workers in the same model as [15,16]. And then, they proposed a new fair payment scheme, which only used traditional e-cash scheme to produce payment token. Their solution performed more effectively than the prior schemes.

It seems that all existing solutions in the payment scheme for outsourced computations adopt the idea of ringer to solve the verification problem of computation result. However, the idea of ringer is designed for a class of computations that called inversion of a one-way function. Du et al. [14] pointed out that the ringer scheme is limited to a specific computation and presented a new solution called commitment-based sampling which can be used for generic computations. Besides, all the existing methods to solve the trust problem of retrieving payments are based on the traditional e-cash and need a bank to generate payment token. However, the bank is the bottleneck of the payments system. If the transaction costs is too high in the system, the bank is unwilling to implement. Different from the traditional e-cash, Bitcoin is a decentralized crypto-currency system and users can transact directly without needing a bank. To the best of our knowledge, it seems that there is no works on fair payments based on Bitcoin.

1.1. Contribution

In this paper, we propose a fair Bitcoin-based payment scheme for outsourcing computations of fog devices. In our proposed protocol, the outsourcer O and the workers W do not trust each other. We use the idea of Bitcoin contract to guarantee that no matter how a malicious O behaves, W will be paid if he is honest. In the case of malicious workers, a third party T is introduced to help O to get his deposit back. However, T is not fully trusted while only involved in the case of dispute. The contributions of this paper are as follow:

1. We construct a fair payment protocol integrated with Bitcoin. The proposed protocol enables an automatic penalty to W if O does not pay as he promised. Different from the previous works, there is not a bank to issue a coin in our system. The setting of our structure is simpler than the existing ones.
2. We use commitment-based sampling scheme instead of ringer to solve the trust problem in W. Taking the advantages of commitment-based sampling scheme, our structure can be used for generic computations in the outsourcing computing. The proposed protocol is also efficient in communication as well as in computation.

1.2. Organization

The rest of the paper is organized as follows: Some preliminaries are given in Section 2. In Section 3, we propose our efficient fair payments protocol based on bitcoin. The security analysis of the conditional payments scheme and comparison with existing schemes are discussed in Section 4. Finally, we give a brief conclusion.

2. Preliminaries

In this section, we first introduce the definition of outsourcing computation and the security properties should be satisfied in our scheme. Then, we give a short description of bitcoin and an overview of commitment-based sampling.

2.1. Definition of outsourcing computation

We consider the following general definition of outsourcing computation in which mutually distrusting participants are taking part. Formally, such computations are defined by the following elements [13]:

- A job $F = \langle f, D, M\rangle$. The outsourcer assigns a job $F = \langle f, D, M\rangle$ to workers. The task of workers is to find some $x \in D$ such that $f(x) \in M$. The outsourcer firstly partitions D into small subsets D_i. That is, the computation task has been split into subtask F_i. Then F_i assigned to worker i is to compute f on D_i. M_i is a set containing all $f(x)$ that an outsourcer needs. The work of every worker is to compute $f(x)$ for all $x \in D_i$. At last, he returns all of those x such that $f(x) \in M_i$.
- A screener $S_i(x, f(x))$. The screener S_i is a program that takes as input a pair $(x, f(x))$ for $x \in D_i$, and returns a string s. The purpose of S_i is to confirm the value of the output of f for outsourcer. We assume that the run time of S_i is negligible compared to the overhead of the evaluation of f.
- A payment scheme P_i. The payment P_i can be thought as an standard bitcoin transactions T_x that an amount v is transferred from outsourcer to workers in bitcoin system. Besides, outsourcer is required to prepare a certain amount of bitcoin, which will be given to workers if he refuses to pay within some time t when worker finishes his jobs.

In this paper, different from the previous works [19,20], there is not a bank contained in our protocol. We only consider three participants, an outsourcer O, a worker W, and a third party T. In our protocol, the third party T is semi-trusted. Thus he may collude with one participant to defraud another one. This assumption is the same as Chen's scheme [19]. But in our proposed payments scheme, T just acts as a sever that it only receives a request from O or W, and sends a response. Once the dispute between W and O arises, T interpose in the dispute to assure the fairness of protocol. In the following, we will consider the security properties of our payment protocol, which are defined the same as in [19].

- **Completeness**: The completeness is said that an honest O can obtain the computation result and he pays remuneration to the honest W. At last, O can take back his deposit. The dishonest party can communicate with T while he cannot prevent O and W from contacting with each other.
- **Fairness**: We assume a dishonest party plays with an honest party. Generally, we say that the dishonest one has the ability of completely controlling the network. He can interact with T as he wishes and also can delay the requests of honest party arbitrarily. In this sense, the fairness means that the dishonest O cannot obtain the computation results and the deposit in advance if the honest party W does not receive remuneration or the dishonest W cannot obtain remuneration and the deposit if the honest party O does not obtain the computation results.
- **Accountability**: The accountability means that T's cheating actions can be caught. If T misbehaves, it can be detected and proven.

2.2. Bitcoin

Bitcoin is a peer-to-peer decentralized digital currency system introduced by Satoshi Nakamoto [21] in 2008 and it is issued it as open-source software in 2009. Its novel and open design has attracted more and more attention. The system is peer-to-peer and users can transact directly without needing a bank like the traditional electronic cash. What is more, the special properties of the transaction syntax allow to create the so-called contracts. When mutually-distrusting parties participate in a common protocol to conduct finance transactions, the fairness can be guaranteed by Bitcoin contracts.

Andrychowicz et al. [22,23] showed that the advantages of Bitcoin transaction syntax can be used to design secure multiparty computation protocols. They constructed a Bitcoin-based timed commitment scheme, where the committer should publish his secret value in a limited time or be fined. They also proposed protocols for secure multiparty lotteries based on Bitcoin. Bentov and Kumaresan [24] showed how to use Bitcoin system to design fair secure computation protocols step by step. In their work, a new ideal functionalities was proposed for designing fair secure two-party or multiparty protocols with penalties. Ateniese et al. [25] addressed a accountable storage protocol which can be integrated with Bitcoin to support automatic compensations.

Transactions. Addresses and transactions are two important components of Bitcoin currency system. In this system, the elliptic curve digital signature algorithm (ECDSA) and hash function SHA-256 guarantee the security of the system. Every user in the system has a ECDSA key pair denoted by (PK, SK). A bitcoin addresses the hash value of the user's public key PK. The private key and the public key are used to sign and verify the transactions, respectively. Each user in the Bitcoin system must know one pair of keys or more that easily generated offline. We now assume Alice and Bob are two users of the bitcoin system. Their keys pair denote $X = (PK_X, SK_X)$, $X \in A, B$. Alice wants to pay a certain amount of bitcoins (we assume the value is v) to Bob. She prepares a transaction T_x:

$$T_x = (T_y, PK_B, v, t, Sig_A(T_y, PK_B, v, t))$$

where T_y is a previous transaction. $Sig_A(T_y, PK_B, v, t)$ is Alice's signature on $[T_x] = (T_y, PK_B, v, t)$ denoted by *body*. The transaction is valid only if (1) PK_A was a recipient of the transaction T_y, (2) the value of T_y was at least v, (3) the transaction T_y has not been redeemed earlier, and (4) the signature of Alice was correct. (5) the time t is reached. t was a time lock that can be set to zero.

In a real system, users can flexibility in defining the condition on how the transaction T_x be redeemed. Bitcoin uses a scripting system for transactions, which is simple, stack-based just as Fourth programming language, and processed from left to right, purposefully not Turing-complete, with no loops. It provides basic cryptographic functions like calculating hash function or verifying a signature. Similar to [22], the transaction T_x can be represented as a table.

T_x(in: T_y)
ins: σ_A
outs(body,σ): $ver_B(body, \sigma)$
value: v
Lock time: t

where T_y is a previous transaction, σ_A is Alice's signature on the *body* of $[T_x]$. Out-script controls the conditions whether T_x can be redeemed or not. 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.

2.3. A review of commitment-based sampling

Commitment-based sampling (CBS), introduced by Du et al. [14], is a solution to guarantee that computation tasks are carried out properly by workers, and can be used to detect cheating workers in outsourcing computation. We assume that the worker is given a domain D_i, and its task is to find all x for $f(x) \in M_i$ and return those $\{x_1, x_2, \ldots, x_n\}$ that O needs. Then O checks whether every x_i satisfies $f(x_i) \in M_i$In order to reduce communication overhead, O cannot ask W to send back all the outputs. By means of sampling techniques, O randomly chooses a small set of sample inputs and checks the result of these sample inputs. These results sent by W is calculated before he knows samples. Due to the use of Merkle tree to make commitment for the computation results, the workers cannot change the results after he learns the samples.

Merkle hash tree (MHT) [26], a specific binary tree, which contains a function *hash* and an assignment Φ can be used to authenticate data. The function *hash* is a one-way hash function. Φ maps a set of nodes to a set of a fixed-size string. In such a tree, the Φ value of each internal node is the hash value of the concatenation Φ value of its child nodes and each leaf nodes stores hash value $h(d_i)$ of the authenticated data. In Du's scheme [14], the worker constructs a Merkle hash tree with n leaves $\{L_1, \ldots, L_n\}$. The Φ is defined as the following:

$$\Phi(L_i) = f(x_i), \quad \text{for } i = 1, \ldots, n, \tag{1}$$

$$\Phi(V) = hash(\Phi(V_{left}) \parallel \Phi(V_{right})), \tag{2}$$

where "$\parallel$" represents the concatenation of two strings, V denotes an internal tree node, and V_{left}, V_{right} denotes V's two children. To make a commitment on all data on the leaves, the worker just needs to send $\Phi(R)$ to the outsourcer, where R is the root of the Merkle tree.

In the following, we will describe the commitment-based sampling (CBS) scheme:

- **Setup**: The outsourcer O prepares the outsourcing task $F_i = \langle f, D_i, M_i \rangle$. M_i is included in the screener S_i that is sent to worker W to decide what he must store for returning back to O once it finishes the task.

- **Computation**: The screener S_i takes as input a pair $(x, f(x))$ and tests whether $x \in D_i$ or not. If $x \in D_i$, then S_i outputs x; otherwise it outputs $\perp$. W finds each x satisfies $x \in M_i$, collects all the $\{x_1, x_2, \ldots, x_n\}$ output by S_i. W makes a commitment on x_k for $k \in \{1, \ldots, n\}$ as the following steps:
  1. Building Merkle Tree. W constructs n leaves $L_1, \ldots, L_n$ and uses these leaves to build a complete binary tree. At the end W sends $\Phi(R)$ to the outsourcer.
  2. Let h be a one-way hash function. W uses the following method to generate m numbers $i_1, \ldots, i_m$ in the $\{1, \ldots, n\}$: $i_k = (h^k(\Phi(R)) mod n) + 1$ for $k = 1, \ldots, m$, where $h^k(\Phi(R)) = \begin{cases} h(\Phi(R)), & \text{for } k = 1, \\ h(h^{k-1}(\Phi(R))), & \text{for } k = 2, \ldots, m. \end{cases}$ For $i \in \{i_1, \ldots, i_m\}$, x_i are the samples. For each $i \in \{i_1, \ldots, i_m\}$, W finds the path λ from the leaf node L_i to the root R. For each $v \in \lambda$, W sends to the outsourcer O v's sibling Φ, whose values are denoted by $\lambda_1, \ldots, \lambda_H$. Let S denotes the result set $\{x_1, x_2, \ldots, x_n\}$, S' denotes the set of samples points, P denotes the verified path set. At last, the worker also sent $S, S', P, f(x_i), \Phi(R)$ to the outsourcer.
- **Verification and Payment**: The outsourcer O verifies whether $x_i, i \in \{i_1, \ldots, i_m\}$ from worker is correct.
  1. If $f(x_i) \notin M_i$, that is, x_i is not the outsourcer needed. Then he aborts. The outsourcer refused to pay to the worker.
  2. If $f(x_i) \in M_i$, the outsourcer reconstructs the hash tree from x_i and $\lambda_1, \ldots, \lambda_H$ by using the recursive algorithm defined in section. He compares the root $\Phi(R')$ with $\Phi(R)$. If $\Phi(R) \neq \Phi(R')$, the outsourcer aborts. The worker is dishonest and the outsourcer refused to pay to the worker. If $\Phi(R) = \Phi(R')$, the worker is honest and the outsourcer should pay to him.
  3. For all $i \in \{i_1, \ldots, i_m\}$, the verification succeeds. Then, the outsourcer is assured that computation tasks are carried out properly by workers and the worker has not cheated. Then the worker obtains payment.

3. Fair payments protocol based on Bitcoin

3.1. High description

In this paper, we consider the securely outsourcing computations under the commercial mode, which is similar to [19,12,15,13]. There exist trust problems between the outsourcer O and the workers W in this scenario. O does not believe that W can accomplish the task correctly and W does not trust O can afford the remuneration. In the previous works, many solutions had been proposed to solve this type of trust issues, which are based on the traditional electronic cash to design payment scheme. While our proposed scheme is based on the Bitcoin system which does not contain a bank. More importantly, an attractive technology provided by the Bitcoin system makes the honest worker to ensure that no matter how a malicious O behaves, he can obtain the corresponding reward if he does not cheat.

The main process of our protocol is depicted in Fig. 1. In the first step, O and W firstly come to an agreement through Bitcoin system. O firstly puts aside a certain amount of money as his deposit. When W accomplished his tasks honestly without getting remuneration from O, he will get O's deposit automatically as his compensation. And the amount of money is more than the value of the payment that O will pay. After confirming the deposit transactions is indeed made by O, W begins executing the assigned computations tasks and then returns the results to O. In normal case, O will pay W and get the deposit back. If dispute arises among the two parties, a third party is introduced to resolve the dispute. The underlying idea of our protocol is that the fairness between O and W is guaranteed by the deposit guarantee mechanism. In our construction, we apply the security deposit of d' bitcoins by O via a special bitcoin transaction T_D and the related transactions T_P and T_G, described in the follow protocol. Using the transactions, O can be forced to commit some money called deposit that will be given to W if he refuses to pay.

3.2. The proposed fair payments protocol

We present a fair bitcoin-based payment scheme in this section. In the following, we will introduce some notations used in our protocol. Assume every party has a ECDSA key pair denoted by $X = (PK_X, SK_X), X \in \{O, W, T\}$ used in bitcoin system, respectively. All the parties have come to an agreement on a public key used in receiving bitcoin. Denote by $Sig_X(M)$ the signature on the message with the secret key SK_X of the party $X \in \{O, W, T\}$.

- **Setup**: The outsourcer O prepares the outsourcing instance F_i. Through a special transaction in Bitcoin system, O creates a Bitcoin contract to form agreements with worker W. O firstly commits $d', d' \geq d$ bitcoins as "security deposit". W will automatically obtain d' bitcoins as compensation if O does not pay him d bitcoins after he honestly finishes the computing job.
  - Assign task: The outsourcer O prepares the outsourcing task $F_i = \langle f, D_i, M_i \rangle$. M_i is included in the screener S_i that is sent to the worker W to decide what he must store for returning back to O once he finishes the task.
  - Contract building: O prepares an unredeemed transaction T_y that can be redeemed with a key which is only known to O. O builds a new transaction T_D described in the following table and posts it to the Ledger. This shows the conditions how the deposit can be redeemed.

T_D(in: T_y)
ins: $Sig_O([T_D])$
outs(body,σ_1,σ_2):
$(ver_O(body, \sigma_1) \wedge ver_W(body, \sigma_2)) \vee$
$(ver_O(body, \sigma_1) \wedge ver_T(body, \sigma_2))$
value: d'
Lock time: 0

Then O creates the bodies of the transactions T_P, signs it and sends the signed body $[T_P]$ to W. If T_D does not appear on the ledger and the signed transactions T_P does not arrive to W in time t_1, then he aborts.

T_P(in: T_D)
ins:
$Sig_O([T_P])$,$Sig_W([T_P])$, $\perp$
outs(body,σ):$ver_W(body, \sigma)$
value: d'
Lock time: t

- **Computation**:
  - W uses the screener S_i to output a set S, which contained all x satisfies $f(x) \in M_i$.
  - W makes the commitment for $x_k, k \in \{1, \ldots, n\}$ by using the Merkle hash tree as the steps described in Section 2. At last, W sends $\Phi(R)$ to O.
    (1) Then W generates m numbers $(i_1, \ldots, i_m)$ in $[1, n]$. Denoted the samples set as S'.
    (2) For each $i \in \{i_1, \ldots, i_m\}$, W finds the path λ from the leaf node L_i to the root R. For each $v \in \lambda$, v's sibling Φ, whose value are denoted by $\lambda_1, \ldots, \lambda_H$, is sent to the outsourcer O.
    (3) At the end, the worker sends (S, ev) to the outsourcer. Here $ev = (S', P, f(x_i), \Phi(R))$.

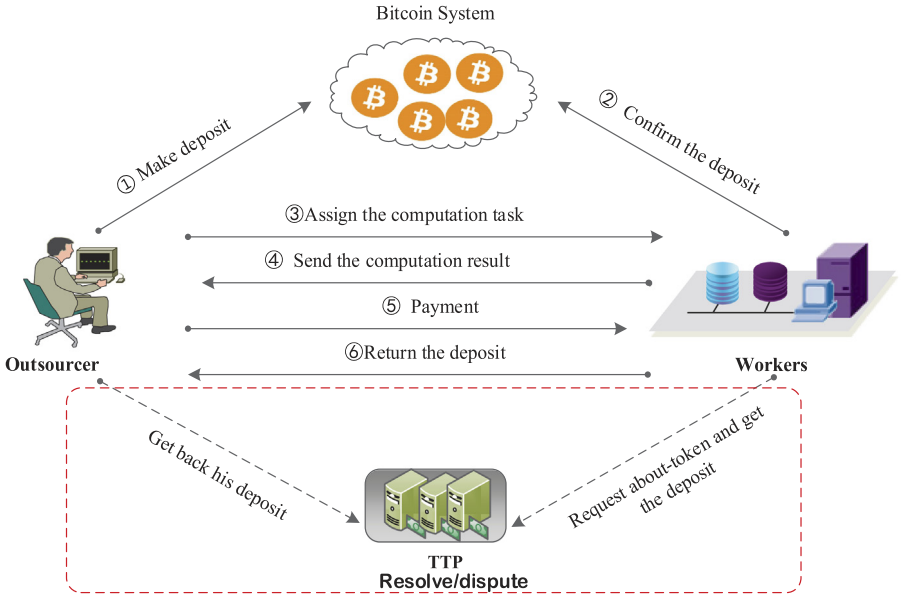


Fig. 1. The framework of our protocol.

- **Verification and Payment**: O and W are involved in a protocol, in which O verifies (S, ev), transfers d bitcoins to W and gets his deposit d' bitcoins back. The protocol is compose of three phases. In the normal case, only the *Phase 1* is performed. If O rejects to pay W after he receives the result before deadline time, W runs the *Phase 2* to ask T to publish a abort token. He waits until the deadline time and gets the deposit of O. If W cannot send (S, ev) to O or refuse to cooperate with O to redeem his deposit, O can perform the *Phase 3* to get his deposit with the help of T.
  - Phase 1: Running between O and W.
    (1) W sends (S, ev) to O before time t_2. If O does not receive (S, ev), he runs *Phase 3*
    (2) If (S, ev) is the valid solution for F_i, O prepares a new standard transactions T_{pay} and posts it to the Ledger. W receives d bitcoin as his reward.

    T_{pay}(in: $T_{y'}$)
    ins: $Sig_O([T_{pay}])$
    outs($body$,σ):$ver_W(body, \sigma)$
    value: d
    Lock time: 0

    (3) W creates the transactions T_G, signs it and sends the signed body of T_G to O, O posts the transaction T_G and he gets his deposit back.

    T_G(in: T_D)
    ins: $Sig_O([T_G]), Sig_W([T_G])$
    outs($body$,σ):
    $ver_O(body, \sigma)$
    value: d'
    Lock time: 0

  - Phase 2: If O does not pay W by time t_3.
    (1) W contacts with T. He first computes the signature $Sig(SK_W, abort \parallel (S, ev), F_i, S_i)$ and sends $Sig(SK_W, abort \parallel (S, ev), F_i, S_i), (S, ev), F_i, S_i$ to T.
    (2) If the current time does not reach the time t_2, the protocol is terminated. If the signature verification is correct and O has not run *Phase 3*, T publishes an abort token TK and sends (S, ev) to O.
    $TK = Sig(SK_T, SIG(SK_W, abort \parallel (S, ev), F_i, S_i))$
    (3) W waits until the deadline time t, he posts the transaction T_P and receives a compensation of d' bitcoins from O.
  - Phase 3: A protocol is run by O, W and T. If O did not receive (S, ev) by t_1 and W did not run Phase 2 or if W refused to sign the body of transactions T_G by time t_4.
    (1) O contacts with T. He sends the timestamped transcript and all the validation messages denoted by *proof*, which should be tamper-proof. O computes the signature $Sig(SK_O, proof)$ and sends *proof*, $Sig(SK_O, proof)$ to T.
    (2) T verifies weather the signature of the *proof* is valid or not. If the verification is correct, then there are two cases we considered:
    * If the current time does not reach t_3, T contacts with W and asks him to return the result. If W returned the computation result (S, ev) to T, then T refuses O's request and sends the result to O. Otherwise, T uses the secret key SK_T to sign on the transactions T_{get} and sends the signed body of T_{get} to O, O posts the transaction T_{get} and he gets his deposit back.
    * If the current time exceeded time t_4 and W didnot run **Phase 2**, T sends O the signed body of T_{get} and O gets his deposit back.

    T_{get}(in: T_D)
    ins: $Sig_O([T_{get}]), Sig_T([T_{get}])$
    outs($body$,σ):
    $ver_O(body, \sigma)$
    value: d'
    Lock time: 0

Remark 1. Bitcoin contract is a method of forming agreement between mutually distrusting parties on the bitcoin blockchain [27]. Based on the idea of smart contract described by Nick Szab [28], Mike Hearn [27] listed several application scenarios, just like creating the assurance contracts, the rapid micropayment, the multiparty lotteries [22]. In the real bitcoin system, signature can be verified flexibly, because the signature of transaction can be controlled by the SIGHASH symbols, which are appended to the signature. We can build special contracts by means of this symbols. In a transaction, one party only signs a part of the transaction and the remaining unsigned part can be signed without other

party's involvement. The SIGHASH flags consist of a mode and the ANYONECANPAY modifier. Bitcoin scripts can include the CHECKMULTISIG operation codes, which provides n-of-m signature verification: there are multiple public keys(i.e. m), but the number of valid signatures (i.e. n) should be defined, where $n < m$. There are two general patterns for safely creating bitcoin contracts in the bitcoin blockchain. A more detailed description of Bitcoin contracts is available on the Bitcoin wiki site [29]. In our protocol, there are two transactions: T_P and T_D. T_P, which can be viewed as the contract, is created and signed but not be immediately posted. T_D, which can be viewed as the payment transaction, is posted to the blockchain. This means that the deposit of O is "frozen" until O honestly pay W. If O does not pay W and then the contract is broadcasted. Until this moment, W obtains this frozen deposit.

Remark 2. Our protocol includes a third party T who is only involved in the case of dispute. This idea is similar to the fair exchange introduced by Asokan et al. [30–32,20]. For each *Phase i*, there is a deadline t_i to complete, $i \in \{1, 2, 3, 4\}$, where $t_1 < t_2 < t_3 < t_4$, $t \gg t_i$. We assume that all messages to be authenticated are signed by the parties to avoid tampering and neither O nor W can forge the timestamped transcript of the protocol.

Remark 3. The features of Bitcoin script forms the flexible properties of Bitcoin transactions. These attractive properties of transactions make it possible to deposit some amount of money which can only be redeem under certain conditions. In our paper, we also use the attractive properties of Bitcoin to build our fair protocol and solve the trust problems between O and W. The security of Bitcoin contract has been verified by Andrychowicz [33]. They proposed a framework for modeling the Bitcoin contracts by using the timed automata [34,35].

4. Analysis of our proposed protocol

4.1. Security analysis

In this section, we analyze the security of the proposed protocol. We assume that the outsourcer O and the worker W are mutually-distrusting. The semi-trusted third party T may collude with one party to cheat another one.

Theorem 1. *The proposed Bitcoin-based payments protocol satisfies the security requirement of completeness.*

Proof. In normal case, both O and W will successfully perform the *Phase 1*. O obtains computation result (S, ev) and W obtains d bitcoin from O for his reward. At last, O gets back his deposit d' bitcoin.

Theorem 2. *The proposed Bitcoin-based payments protocol satisfies the security requirement of fairness.*

Proof. Firstly, we assume that there are an honest O and a dishonest W. W can cheat successfully which means that W obtains deposit but O does not obtain (S, ev) before deadline time t.

Assume that O does not obtain (S, ev) before deadline time t, then W cannot obtain the reward. But W wants to obtain the deposit d' bitcoin from O which he should return back to O. Therefore, he must run the protocol *Phase 2* with T successfully. But O can also obtain (S, ev) from T. In this case, it derives a contradiction. Therefore, the successful probability that W cheats is negligible.

Secondly, we assume that there are a dishonest O and an honest W. We say that O can cheat successfully if and only if O obtains the computation result (S, ev) and gets back his deposit while W obtains nothing. Similarly, we also assume that the worker W receive neither his reward nor the deposit that they had come to agreement before. If the entire computation task is uncompleted before deadline, then O cannot receive the result (S, ev). Otherwise, W can perform protocol *Phase3* to receive the deposit successfully with the help of T. We say this conflicts with the hypothetic. Therefore, the successful probability that the outsourcer O does not pay W when W finishes the entire computation task is negligible.

Table 1
Comparison of the three algorithms.

	Scheme [15]	Scheme [19]	Our scheme
Bank	Yes	Yes	No
Optimistic	No	Yes	Yes
Ringer	Yes	Yes	No
Computation complexity	$O(n)$	$O(1)$	$O(1)$

Theorem 3. *The proposed Bitcoin-based payments protocol satisfies the security requirement of accountability.*

Proof. Firstly, we suppose that the *Phase 2* protocol is performed, but O does not receive the computation results (S, ev) from T before the deadline time. It means that T colludes with W or the computation task is uncompleted before deadline time. Another possibility is that W sends the results (S, ev) to T but T refuses to send it to O. Whatever the case, the *Phase 3* protocol should be performed. O obtains T's signature on the transactions T_D and O can successfully get his deposit back and W obtains nothing.

Similarly, we also suppose that the *Phase 3* is performed. If T colludes with O, it means that W has completed the whole task but he does not receive the reward. In this case, the *Phase 2* protocol should be performed. Then W obtains the abort token from T and the T's signature on the transactions T_D. He posts the transactions T_{get} to the bitcoin network but fails to get the deposit, he can catch $T's$ cheating actions. Because both *Phase 2* and *Phase 3* are successfully performed by T.

4.2. Comparison

In this section, we compare the proposed scheme with the previous schemes [15,19]. Table 1 presents the comparison among the three schemes.

Firstly, all of the three schemes discuss the solution in the same framework which is similar to the one presented in [13]. Secondly, our proposed protocol is based on Bitcoin system without containing a bank. All the other ones need a bank to be involved both in the payment generation and the redemption transactions. However, the bank is the system bottleneck. If the transaction overheads is too high in the system, the bank is unwilling to implement. Besides, in the aspect of computation complexity, scheme [15] used the secret sharing scheme to split the payment token into n shares. Therefore, the computation complexity of scheme [15] for payment generation and redemption is $O(n)$, where n is the number of ringer. Our scheme and scheme [19] is only $O(1)$. At last, we consider the communication complexity. Scheme [15] used the interactive cut-and-choose protocol to ensure their validity in the payment verification. Thus it requires at least 3 round of communications, whereas in our scheme, we only need 1 round of communications.

4.3. Performance evaluation

In this section, we provide the experimental evaluation of the proposed payment protocol. We have tested these on a Linux box with Intel (R) Core (TM) i7-4710 HQ CPU that clocks at 2.50 GHz

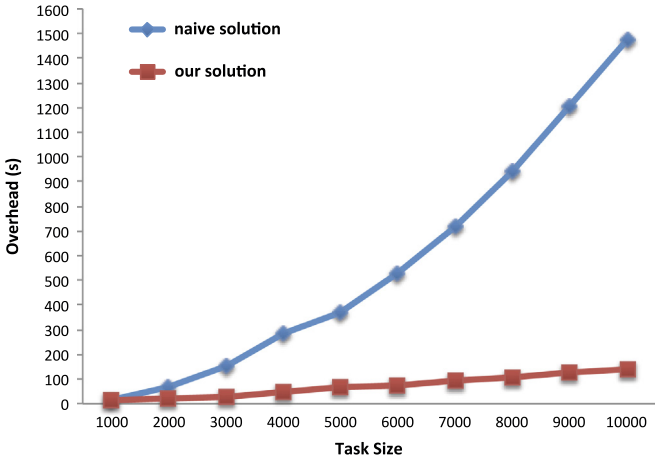


Fig. 2. Time cost of verification.

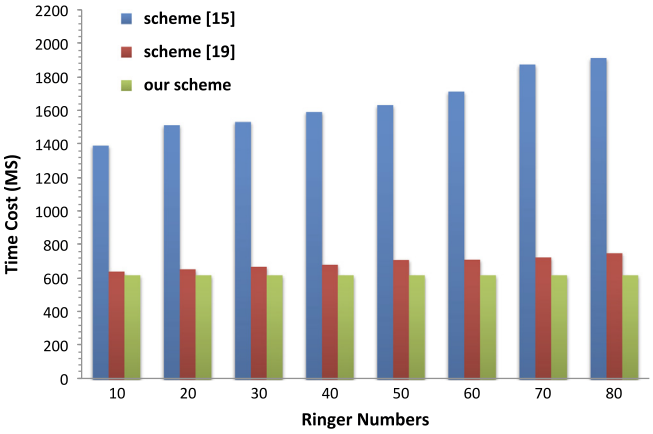


Fig. 3. Time cost of payment.

and has 16.0 GB of RAM. The code was written in ubuntul2.04 code Python. We used the Python built-in pycrypto module and the code for merkle tree was downloaded from GitHub. In our experiment, we only tested the approximate cost of payment processes and the overhead of the result verification for the outsourcer.

We consider the task type that is SHA-256 hash inversion. We define the task as a triple (SHA-256, D, M). The task consists of applying SHA-256 to each value from a given domain D. The result of the task consists of all input value $x \in D$ for which SHA-256$(x) = y$. In our proposed protocol, we use the commitment-based sampling scheme instead of ringer. In order to verify the result after the worker finishing the computations task, the outsourcer should restructure the merkle tree. Du et al. [14] proved the following theorem: The probability that the worker with honesty ratio r can cheat successfully is $Pr(\textit{cheating succeeds}) = (r + (1 - r)q)^m$, where m is the numbers of samples. Let D' be the set of results that are computed honestly by the worker, $r = \frac{|D'|}{|D|}$. q is the probability that the worker can guess the correct result of x. when the $Pr(\textit{cheating succeeds}) = 0.0001$, we only need 14 samples. In Fig. 2, we provide the time costs simulation for the result verification when the samples is 14. Compared with the naive solutions, it is obvious that our protocol is more efficient for the outsourcer side overhead.

Fig. 3 studies the approximate time costs in the payment generation for schemes [15,19] and our proposed scheme. In our experiment, we only consider the case that one outsourcer and one worker were contained in the protocol. Scheme [15] used the secret sharing scheme to split the payment token into n shares. The time cost of scheme [15] was composed of the generation of ringers, splitting a payment token, obfuscating the share and blinding each share with a ringer set. Just as described in [15], we provide the time taken by each component when the number of ringer numbers from 10 to 80. Different from scheme [15], there was not secret sharing/splitting scheme used in scheme [19]. Therefore, the time cost of payment generation was composed of the generation of ringers and payment-blinding to a job. It only contains 8 exponents operations, 1 modular inverse operation, 8 modular multiplications, 1 RSA signature. Our scheme is based on the Bitcoin system. So, we should only prepare 2 transactions and need only 2 ECDSA signatures in our scheme. Besides, our scheme does not use the idea of ringer. No matter how the ringer numbers increase, the time cost of the payment generation in our scheme remains unchanged. We can see from Fig. 3 that our scheme is more efficient than the others.

5. Conclusion and future work

In this paper, we propose a new fair payment for outsourcing computations based on Bitcoin currency. Our proposed construction enables the outsourcer O and the workers W can exchange the computation results and the payments in a manner of fairness. Different from the existing scheme, we adopt Bitcoin system to guarantee that no matter how a malicious O behaves, W will be paid if he does not cheat. The proposed protocol uses a semi-trusted third party T to help O to get his deposit back if W is malicious.

Bitcoin, an emerging digital currency, has recently gained noticeable popularity. Due to the advantage of Bitcoin system, it brings significant influence on economic and technology. Different from the traditional currency system, Bitcoin is decentralized crypto-currencies which lacks of supervision mechanism of a bank or an authority. Criminologists and security researchers have highlighted the risks of Bitcoins, such as fraud, money laundering and theft of bitcoins [36], which is commonly seen in digital payment systems [37,38]. In the future work, we will focus on forensic study on Bitcoin. Digital forensic for cloud and mobile devices is an emerging research area and scholars have done a lot of researches in this field [39–43]. The main works are focus on client forensics of cloud [44], digital forensic framework [45] and server-side applications [46]. The research on the forensic investigations of Bitcoin is just starting and little research has been conducted. In future, we will determine whether digital forensic techniques for cloud could be used for forensic investigations of Bitcoins.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61572382 and U1405255), China 111 Project (No. B16037), National High Technology Research and Development Program (863 Program) of China (No. 2015AA016007), Doctoral Fund of Ministry of Education of China (No. 20130203110004), Program for New Century Excellent Talents in University (No. NCET-13-0946), Natural Science Basic Research Plan in Shaanxi Province of China (No. 2016JZ021), Guangxi Cooperative Innovation Center of cloud computing and Big Data (No. YD16506), the CICAEET Fund and the PAPD Fund.

References

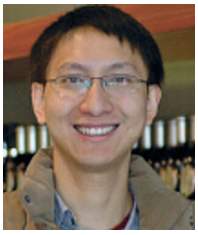
[1] F. Bonomi, R.A. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the Internet of things, in: Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing, MCC@SIGCOMM 2012, ACM, Helsinki, Finland, 2012, pp. 13–16.

[2] L.M.V. Gonzalez, L. Rodero-Merino, Finding your way in the fog: Towards a comprehensive definition of fog computing, Comput. Commun. Rev. 44 (5) (2014) 27–32.
[3] Z. Xia, X. Wang, X. Sun, Q. Wang, A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data, IEEE Trans. Parallel Distrib. Syst. 27 (2) (2016) 340–352.
[4] Z. Fu, K. Ren, J. Shu, X. Sun, F. Huang, Enabling personalized search over encrypted outsourced data with efficiency improvement, IEEE Trans. Parallel Distrib. Syst. 27 (9) (2016) 2546–2559.
[5] J. Zhu, D.S. Chan, M.S. Prabhu, P. Natarajan, H. Hu, F. Bonomi, Improving web sites performance using edge servers in fog computing architecture, in: Proceedings of the 7th International Symposium on Service-Oriented System Engineering, SOSE 2013, IEEE Computer Society, San Francisco, CA, USA, 2013, pp. 320–323.
[6] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, M. Satyanarayanan, Towards wearable cognitive assistance, in: Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2014, 2014, pp. 68–81.
[7] F. Monrose, P. Wyckoff, A.D. Rubin, Distributed execution with remote audit, in: Proceedings of the Network and Distributed System Security Symposium, NDSS 1999, The Internet Society, 1999.
[8] P. Golle, S.G. Stubblebine, Secure distributed computing in a commercial environment, in: Proceedings of the 5th International Conference on Financial Cryptography, Vol. 2339, FC 2001, Springer, 2001, pp. 279–294.
[9] D. Szajda, B.G. Lawson, J. Owen, Hardening functions for large scale distributed computations, in: Proceedings of 2003 Symposium on Security and Privacy, SP 2003, IEEE, 2003, pp. 216–224.
[10] L.F.G. Sarmenta, Sabotage-tolerance mechanisms for volunteer computing systems, Future Gener. Comput. Syst. 18 (4) (2002) 561–572.
[11] D. Szajda, B. Lawson, J. Owen, Toward an optimal redundancy strategy for distributed computations, in: Proceedings of 2005 International Conference on Cluster Computing, CLUSTER 2005, IEEE, 2005, pp. 1–11.
[12] B. Carbunar, R. Sion, Uncheatable reputation for distributed computation markets, in: Proceedings of the 10th International Conference on Financial Cryptography and Data Security, Vol. 4107, FC 2006, Springer, Anguilla, British West Indies, 2006, pp. 96–110.
[13] P. Golle, I. Mironov, Uncheatable distributed computations, in: Proceedings of the Cryptographer's Track at RSA Conference, Vol. 2020, CT-RSA 2001, Springer, 2001, pp. 425–440.
[14] W. Du, J. Jia, M. Mangal, M. Murugesan, Uncheatable grid computing, in: Proceedings of the 24th International Conference on Distributed Computing, Systems, ICDCS 2004, IEEE Computer Society, 2004, pp. 4–11.
[15] B. Carbunar, M.V. Tripunitara, Fair payments for outsourced computations, in: Proceedings of the 7th Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2010, IEEE, Boston, Massachusetts, USA, 2010, pp. 529–537.
[16] B. Carbunar, M.V. Tripunitara, Payments for outsourced computations, IEEE Trans. Parallel Distrib. Syst. 23 (2) (2012) 313–320.
[17] L. Shi, B. Carbunar, R. Sion, Conditional e-cash, in: Proceedings of the 11th International Conference on Financial Cryptography and Data Security, Vol. 4886, FC 2007, Springer, 2007, pp. 15–28.
[18] X. Chen, J. Li, J. Ma, W. Lou, D.S. Wong, New and efficient conditional e-payment systems with transferability, Future Gener. Comput. Syst. 37 (2014) 252–258.
[19] X. Chen, J. Li, W. Susilo, Efficient fair conditional payments for outsourcing computations, IEEE Trans. Inf. Forensics Secur. 7 (6) (2012) 1687–1694.
[20] N. Asokan, V. Shoup, M. Waidner, Optimistic fair exchange of digital signatures (extended abstract), in: Proceeding of the International Conference on the Theory and Application of Cryptographic Techniques, Vol. 1403, EUROCRYPT 1998, Springer, Espoo, Finland, 1998, pp. 591–606.
[21] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2008.
[22] M. Andrychowicz, S. Dziembowski, D. Malinowski, L. Mazurek, Secure multiparty computations on bitcoin, in: Proceedings of 2014 Symposium on Security and Privacy, SP 2014, IEEE, Berkeley, CA, USA, 2014, pp. 443–458.
[23] M. Andrychowicz, S. Dziembowski, D. Malinowski, L. Mazurek, Secure multiparty computations on bitcoin, Commun. ACM 59 (4) (2016) 76–84.
[24] I. Bentov, R. Kumaresan, How to use bitcoin to design fair protocols, in: Proceedings of the 34th Annual Cryptology Conference, Vol. 8617, CRYPTO 2014, Springer, Santa Barbara, CA, USA, 2014, pp. 421–439.
[25] G. Ateniese, M.T. Goodrich, V. Lekakis, C. Papamanthou, E. Paraskevas, R. Tamassia, Accountable storage, IACR Cryptology ePrint Archive 2014, 2014, 886.
[26] R.C. Merkle, Protocols for public key cryptosystems, in: Proceedings of the 1980 IEEE Symposium on Security and Privacy, ACM, 1980, pp. 122–134.
[27] Contract. https://en.bitcoin.it/wiki/Contract.
[28] N. Szabo, Formalizing and securing relationships on public networks, First Monday 2 (9) (1997).
[29] Bitcoin. http://en.bitcoin.it/wiki/.
[30] N. Asokan, V. Shoup, M. Waidner, Asynchronous protocols for optimistic fair exchange, in: Proceedings of 1998 Symposium on Security and Privacy, SP 1998, IEEE, Oakland, CA, USA, 1998, pp. 86–99.
[31] N. Asokan, V. Shoup, M. Waidner, Optimistic fair exchange of digital signatures, IEEE J. Sel. Areas Commun. 18 (4) (2000) 593–610.
[32] N. Asokan, M. Schunter, M. Waidner, Optimistic protocols for fair exchange, in: Proceedings of the 4th Conference on Computer and Communications Security, CCS 1997, ACM, Zurich, Switzerland, 1997, pp. 7–17.
[33] M. Andrychowicz, S. Dziembowski, D. Malinowski, L. Mazurek, Modeling bitcoin contracts by timed automata, in: Proceedings of the 12th International Conference on Formal Modeling and Analysis of Timed Systems, Vol. 8711, FORMATS 2014, Springer, Florence, Italy, 2014, pp. 7–22.
[34] R. Alur, D.L. Dill, Automata for modeling real-time systems, in: Proceedings of the 17th International Colloquium on Automata, Languages and Programming, Vol. 443, mICALP 1990, Springer, Warwick University, England, 1990, pp. 322–335.
[35] R. Alur, D.L. Dill, A theory of timed automata, Theoret. Comput. Sci. 126 (2) (1994) 183–235.
[36] K.R. Choo, (Chapter 15) - cryptocurrency and virtual currency: Corruption and money laundering/terrorism financing risks?, 2015, 283–307.
[37] K.R. Choo, New payment methods: A review of 2010c2012 fatf mutual evaluation reports, Comput. Secur. 36 (2013) 12–26.
[38] K.R. Choo, Designated non-financial businesses and professionals: A review and analysis of recent financial action task force on money laundering mutual evaluation reports, Secur. J. 27 (1) (2014) 1–26.
[39] D. Quick, K.R. Choo, Google drive: Forensic analysis of data remnants, J. Network Comput. Appl. 40 (2014) 179–193.
[40] D. Quick, K.R. Choo, Digital droplets: Microsoft skydrive forensic data remnants, Future Gener. Comput. Syst. 29 (6) (2013) 1378–1394.
[41] B. Martini, K.R. Choo, Remote programmatic vcloud forensics: A six-step collection process and a proof of concept, in: Proceeding of the 13th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2014, IEEE Computer Society, Beijing, China, 2014, pp. 935–942.
[42] A. Rahman, N. Hidayah, N.D.W. Cahyani, K.R. Choo, Cloud incident handling and forensic-by-design: cloud storage as a case study, Concurr. Comput.: Pract. Exper. (2016).
[43] N.D.W. Cahyani, B. Martini, K.-K.R. Choo, A.M.N. Al-Azhar, Forensic data acquisition from cloud-of-things devices: windows smartphones as a case study, Concurr. Comput.: Pract. Exper. (2016).
[44] D. Quick, K.R. Choo, Dropbox analysis: Data remnants on user machines, Digit. Investig. 10 (1) (2013) 3–18.
[45] B. Martini, K.R. Choo, An integrated conceptual digital forensic framework for cloud computing, Digit. Investig. 9 (2) (2012) 71–80.
[46] B. Martini, K.R. Choo, Cloud storage forensics: owncloud as a case study, Digit. Investig. 10 (4) (2013) 287–299.

Hui Huang received his M.S. degree on mathematics from Minnan Normal University in 2008. She is currently working toward his Ph.D. degree in cryptography in Xidian University. Her research interests include applied cryptography and cloud security.

Xiaofeng Chen received his B.S. and M.S. on Mathematics from Northwest University, China in 1998 and 2000, respectively. He got his Ph.D degree in Cryptography from Xidian University in 2003. Currently, he is a professor at Xidian University. His research interests include applied cryptography and cloud computing security. He has published over 100 research papers in refereed international conferences and journals. He is in the Editorial Board of Security and Communication Networks (SCN), Computing and Informatics (CAI), and International Journal of Embedded Systems (IJES) etc. He has served as the program/general chair or program committee member in over 30 international conferences.

Qianhong Wu received the Ph.D. degree from Xidian University in 2005. He is currently a professor at School of electronic information engineering, Beihang University, China. His research interests include applied cryptography and cloud computing security. He has published over 100 research papers in refereed international conferences and journals. He has served as the program/general chair or program committee member in over 10 international conferences.



Xinyi Huang received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently a professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, China. His research interests include cryptography and information security. He has published more than 60 research papers in refereed international conferences and journals. His work has been cited more than 1000 times at Google Scholar. He is in the Editorial Board of International Journal of Information Security (Springer) and has served as the program/general chair or program committee member in more than 40 international conferences.

Jian Shen received the B.E. degree from Nanjing University of Information Science and Technology, Nanjing, China, in 2007 and the M.E. degree in Computer Science from Chosun University, Gwangju, Korea, in 2009. Since 2009, he is working toward the Ph.D degree in Computer Science from Chosun University, Gwangju, Korea. Currently, he is a professor at Nanjing University of Information Science and Technology. His research interests include network security, security systems, mobile computing and networking, ad hoc networks and systems, and ubiquitous sensor networks.