1-1-2012

Efficient fair conditional payments for outsourcing computations

Xiaofeng Chen
Key Laboratory of Computer Networks and Information Security, Xidian University, China

Jin Li
Guangzhou University

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Recommended Citation

Chen, Xiaofeng; Li, Jin; and Susilo, Willy: Efficient fair conditional payments for outsourcing computations 2012, 1687-1694.
https://ro.uow.edu.au/infopapers/2470

Efficient fair conditional payments for outsourcing computations

Abstract

The outsourcing computations in distributed environments suffer from the trust problems between the outsourcer and the workers. All existing solutions only assume the rational lazybut- honest workers. In this paper, we first introduce the rational lazy-and-partially-dishonest workers in the outsourcing computation model. In addition, we propose a new fair conditional payment scheme for outsourcing computation that is only based on traditional electronic cash systems. The proposed construction uses a semitrusted third party T to achieve the fairness and efficiency. However, is only involved in the protocol in the exceptional case, namely in the case of disputes. Moreover, since neither the secret sharing/splitting scheme nor the cut-and-choose protocol is used for the generation or verification of the payment token, our solution clearly outperforms the existing schemes in terms of efficiency.

Keywords

era2015, computations, fair, conditional, payments, outsourcing, efficient

Disciplines

Physical Sciences and Mathematics

Publication Details

Chen, X., Li, J. & Susilo, W. (2012). Efficient fair conditional payments for outsourcing computations. IEEE Transactions on Information Forensics and Security, 7 (6), 1687-1694.

Efficient Fair Conditional Payments for Outsourcing Computations

Xiaofeng Chen, Jin Li, Willy Susilo, *Senior Member, IEEE*

Abstract

The outsourcing computations in distributed environments suffer from the trust problems between the outsourcer and the workers. All existing solutions only assume the rational lazy-but-honest workers. In this paper, we first introduce the rational lazy-and-partially-dishonest workers in the outsourcing computation model. Besides, we propose a new fair conditional payment scheme for outsourcing computation that is only based on traditional electronic cash systems. The proposed construction uses a semi-trusted third party T to achieve the fairness and efficiency. However, T is only involved in the protocol in the exceptional case, namely in the case of disputes. Moreover, since neither the secret sharing/splitting scheme nor the cut-and-choose protocol is used for the generation or verification of the payment token, our solution clearly outperforms the existing schemes in terms of efficiency.

Index Terms

Outsourcing computations, Electronic cash, Verifiable encryption, Ringers.

I. Introduction

Computationally expensive tasks that can be parallelized are most efficiently completed by distributing the computation among a large number of processors. For example, the *sieving* for factoring a 768-bit RSA modulus took almost two years on many hundreds of machines. However, on a single core 2.2GHz AMD

X. Chen is with the State Key Laboratory of Integrated Service Networks (ISN), Xidian University, P.R. China.

J. Li is with the School of Computer Science and Educational Software, Guangzhou University, P.R. China.

W. Susilo is with the School of Computer Science and Software Engineering, University of Wollongong, Australia. E-mail: wsusilo@uow.edu.au.

Opteron processor with 2GB RAM, sieving would have taken about fifteen hundred years [30]. Due to the rapid growth of the Internet, it is possible to invite any computer to participate in a distributed computation task. By far, plenty of large distributed computing projects, such as the search for prime numbers (GIMPS, PrimeGrid), the search for extra-terrestrial intelligence (SETI@home), climate forecast (Climateprediction.net), and protein folding (FOLDING@home), have already successfully taken advantages of the power of Internet computations.

Generally, such distributed computations can deploy the so-called "outsourcing paradigm" to accomplish the tasks efficiently. Informally, an outsourcer that has a computation job decomposes the computation tasks into smaller ones and assigns them to multiple volunteer workers. Each worker completes the corresponding parts of the job and sends the result to the outsourcer. Specifically, we consider the following computation model: A job takes as a function $f : D \longrightarrow M$ and requires the evaluation of f for all values in D. An outsourcer O partitions the domain D into subsets $D_1, \cdots, D_n$ and then allocates D_i, f, and a value $y \in M$ to a worker W. W computes $f(x)$ for all $x \in D_i$ and return those x such that $f(x) = y$. O pays W if and only if W indeed completes its job before the deadline.

The above model for distributed outsourcing computations has two security problems. Firstly, in a commercial setting where the pay for volunteer workers is proportional to their contribution, there is much incentive for workers to minimize the amount of their work in order to retrieve the full payments. Thus, O does not trust that W will do the whole job that he/she undertakes, $i.e.$, to compute $f(x)$ for every $x \in D_i$. Secondly, since O could be any entity of the Internet, W does not trust that it will be paid by O after W has accomplished its task. This may partially weaken the motivation of W to perform the outsourcing computations.

Golle and Mironov [27] first introduced the concept of ringers to present an efficient solution to the first trust problem. With ringers, O is able to ensure, with an overwhelming probability, that W indeed completed its entire computation task. However, the solution itself assumes that O is fully trusted by W, which violates the basic requirement of outsourcing computation. Carbunar and Tripunitara [18] first addressed the second trust problem and proposed a fair conditional payments for oursourcing computation. However, the solution uses the complicated (time-consuming) cut-and-choose protocol and secret sharing scheme and thus is very inefficient for real and practical applications. Recently, Carbunar and Tripunitara [19] proposed a new fair payment scheme for outsourcing computations that can be viewed as an instance of conditional e-payments [36]. However, the solution also uses the inefficient cut-and-choose protocol.

It seems that all of the existing solutions [18], [19], [27] only assume the rational "lazy-but-honest" workers. That is, a "lazy-but-honest" worker will try to minimize the amount of work it needs to perform in order to retrieve the payment, while it will provide the computation results to the outsourcer. This is not a reasonable assumption since W and O do not trust each other in the distributed computation environment. The idea of ringers can only ensure that W must complete its entire computation task in order to retrieve the payment. To the best of our knowledge, it seems that the *third* trust problem has not received much attention in the literature: O does not trust that W will provide the computation results. Actually, in the both solutions [18], [19], W can redeem the payment token from the bank even it does not send the computation results to O. From the standpoint of exchange, this is unfair to O and will defeat the aim of outsourcing computations. Therefore, it is more reasonable to assume the rational "lazy-and-partially-dishonest" workers in the outsourcing computation model. Loosely speaking, a rational "lazy-and-partially-dishonest" worker will try to minimize the amount of work it needs to perform, and will not provide the computation results to the outsourcer except in the case where it cannot retrieve the payment token. Trivially, if a worker is not willing to send the computation results to the outsourcer at the expense of the payment, we say that it is irrational.

The third trust problem in outsourcing computations can be viewed as a special instance of fair exchange protocols: O and W exchange the payment token and computation results in a fair manner. Early work on fair exchange focused on the gradual release of secrets to obtain fairness [7], [21], [24]. However, such a solution mainly has two disadvantages. Firstly, the fairness is based on an unrealistic assumption of equal computational effort for both parties. Secondly, the protocol is very inefficient since it requires plenty of rounds of interaction between the two parties. An alternative approach to fair exchange is to use an on-line trusted third party [23], [25], [39]. However, the major disadvantage of such protocols is that the third party is always involved in the exchange and thus becomes a bottleneck. Asokan et al. [2] introduced the idea of *optimistic* fair exchange, where the third party is only involved in the protocol in the exceptional case, namely in the case of dispute. This approach results in plenty of efficient fair exchange protocols [1], [4], [5], [9], [10], [20], [26], [29].

Our Contribution. In this paper, we propose a new fair conditional payments for distributed outsourcing computations, where the outsourcer and the rational "lazy-and-partially-dishonest" workers do not trust each other. The proposed construction uses a third party T to solve the trust problems. However, T is not fully trusted and is only involved in the protocol in the exceptional case, namely in the case of dispute. Our contributions are two-folds:

1. We first introduce the third trust problem in the outsourcing computation model: The outsourcer does not trust that the rational "lazy-and-partially-dishonest" workers will provide the computation results. Besides, we propose a new conditional payment scheme that can solve all the three trust problem between the outscourcer and the workers.

2. The proposed conditional payment scheme is only based on traditional electronic cash systems. That is, the bank can issue a coin without the setting of some complicated electronic cash systems such as endorsed e-cash [13] or conditional e-cash [36]. Moreover, since neither the secret sharing/splitting scheme nor the cut-and-choose protocol is used for the generation or verification of the payment token, our solution is much more efficient than the existing ones [18], [19].

A. Related Work

The problem of outsourcing computations in distributed environments has been well studied in several research communities. The security model for distributed computations in a commercial environment is presented in [27], [28]. The outsourcer distributes the work to different workers, verifies the computation result and gives the payments. Monrose et al. [31] proposed the idea of using computation proof to ensure correct worker behavior. Golle and Stubblebine [28] presented a solution to provide the result verification by duplicating computations. Szajda et al. [37] and Sarmena [34] proposed the probabilistic verification mechanisms for increasing the chance of detecting misbehavior. In the same setting, Szajda et al. [38] proposed a strategy for distributing redundant computations. Carbunar and Sion [16] proposed a solution where workers are rated for the quality of their work by a predefined number of randomly chosen witnesses.

Golle and Mironov [27] first introduced the concept of ringers to elegantly solve the trust problem of verifying computation completion for the "inversion of one-way function" class of computations. Du et al.

[22] proposed a solution to this problem by requiring workers to commit to the computed values using Merkle trees. Carbunar and Tripunitara [18] firstly addressed the trust problem of retrieving payments after performing the computation and present a conditional payment scheme. However, the solution is impractical for realistic applications since it uses the inefficient secret sharing scheme and cut-and-choose protocol. Recently, Carbunar and Tripunitara [19] proposed a new fair payment for oursourced computations. Nevertheless, the solution also uses the inefficient cut-and-choose protocol. Both solutions can be viewed as using an instance of a conditional e-cash [36]. On the other hand, the solutions assume only the lazy-but-honest workers. Therefore, neither of them can solve the trust problem of obtaining the computation result from the dishonest workers.

B. Organization

The rest of the paper is organized as follows: Some preliminaries are given in Section II. The proposed efficient fair conditional payments scheme is given in Section III. The security analysis of the conditional payments scheme is given in Section IV. Finally, conclusions will be made in Section V.

II. Preliminaries

In this section, we first introduce the model for outsourcing computations and then give an overview for the concept of ringers. Besides, we present the proof of knowledge for the equality of discrete logarithms and the verifiable encryption for discrete logarithms, which play an important role in generating the payment token in our scheme.

A. Model for Outsourcing Computations

We consider an outsourcing computation in which an outsourcer assigns the task of a computation to different workers. Also, the outsourcer and the workers do not trust each other. Specifically, we assume the rational "lazy-and-partially-dishonest" workers in the outsourcing computation. Formally, such a computation is defined as follows [27]:

- A job $F_i =< f, D_i, M_i >$. The task of the job F_i is to evaluate a function f on the finite domain D, where $f : D \longrightarrow M$, $D_i \subseteq D$, and M_i is a set of values of interest for an outsourcer O. A worker W needs to compute $f(x)$ for all $x \in D_i$ and return those x values such that $f(x) \in M_i$.

- A payment scheme P_i. The payment P_i can be thought of an electronic payment token (for example, an electronic cash) that is issued by a bank B.
- A screener $S_i(x, f(x), P_i)$. The screener S_i is typically implemented as a program that takes as input a pair $(x, f(x))$ for $x \in D$, and a payment scheme P_i. The output of S_i is a string s. The intent behind S_i is to identify "valuable" output of f, either for O (solutions for the job F_i) or for W (values that aid W in retrieving the payment associated with the job).

There are the following four stages in the model of outsourcing computations.

- **Initialization**: The outsourcer O prepares the outsourcing instance F_i and the payment P_i, and then sends (F_i, P_i) to the worker W.
- **Verification**: W validates P_i to gain assurance that it will be paid once it completes the job.
- **Computation** and **Payment**: For each input $x \in D_i$, W computes $f(x)$ and uses the screener S_i to find x such that $f(x) \in M_i$, which is returned to O. Besides, W uses the screener S_i to derive its payment.
- **Cancellation/Spending**: If W cannot finish the job before the deadline, O can cancel the payment or spend it to others.

B. Ringers

The idea of "ringers", first introduced by Golle and Mironov [27], can be used to solve the problem of the trust in W. That is, O needs to be convinced that W does indeed perform all the computations that were outsourced to him. A ringer is a value chosen by O according to the target of f. There are two kinds of ringers, *true* ringers and *bogus* ringers. A true ringer is such that $x \in D_i$ (recall that D_i is the domain of the function f to be computed by W), and a bogus ringer is such that $x \notin D_i$. If W honestly does its work, then what it sends O at the end is the set of true ringers, and possibly the special pre-image for which O is looking. The ringers ensure that W does its entire work. The bogus ringers makes it more difficult for W to stop prematurely and still make O believe that it did the entire work.

- **Initialization**: O chooses an integer $2m$, the number of ringers. It picks a random integer $t \in [m + 1, \cdots, 2m]$ to be the number of true ringers, and $2m - t$ to be the number of bogus ringers. The distribution of t in $[m + 1, \cdots, 2m]$ is $d(t) = 2^{2m-t-1}$. O computes $f(x)$ for every true and bogus ringer x. These post-images are included in the screener S_i that is sent to W. The screener S_i is used by W to decide what it must store for transmission back to O once it completes the job. O uses this information to infer whether W indeed completed the entire job.

- **Verification**: W runs a protocol with O to ensure that it can retrieve the payment tokens if it completed the job.
- **Computation** and **Payment**: The screener S_i takes as input a pair $(x, f(x))$ and tests whether $f(x) \in \{y, y_1, y_2, \cdots, y_{2m}\}$, where y is the post-image whose pre-image O seeks, and each y_j is the post-image of a true or bogus ringer. If x is indeed in the set, then S_i outputs x; otherwise it outputs the empty string. W computes f for each element in D_i, processes each through S_i, collects all the outputs of S_i and sends them to O to receive its payment.

Golle and Mironov [27] proved the following theorem: The bogus ringers scheme ensures a coverage constant of $1 - \frac{1}{n2^{n+1}} - (\frac{4}{n})^n$, where coverage constant denotes the fraction of D_i on which W must evaluate f before submitting the computation for payment.

C. Knowledge Proof for the Equality of Discrete Logarithms

Let g and h be two generators of the group $\mathbb{G}$ with a prime order q. Let $H : \{0,1\}^* \rightarrow \{0,1\}^k$ be a collision-resistant hash function. A prover with possession a secret number $x \in \mathbb{Z}_q$ wants to show that $x = \log_g u = \log_h v$ without exposing x. Chaum and Pedersen [14] firstly proposed the proof as follows: The prover chooses a random number $r \in_R \mathbb{Z}_q$, and then sends the verifier $c = H(g, h, u, v, g^r, h^r)$, and $s = r - cx \mod q$. The verifier accepts the proof if and only if $c = H(g, h, u, v, g^s u^c, h^s v^c)$. Specially, if $c = H(\mathbf{m}, g, h, u, v, g^r, h^r)$, then (c, s) is a signature on message $\mathbf{m}$ due to the Fiat-Shamir heuristic.

Camenisch and Michels [15] first presented a protocol to prove the equality of discrete logarithms from *different* groups. Let $\mathbb{G}_1 = \langle g \rangle$ and $\mathbb{G}_2 = \langle h \rangle$ be two distinct groups with different prime orders q_1 and q_2, respectively. Let $H : \{0,1\}^* \rightarrow \{0,1\}^k$ be a collision-resistant hash function. Let l be a integer such that $2^{l+1} < \min\{q_1, q_2\}$ and $\epsilon > 1$ be a security parameter. Let $u = g^x$ and $v = h^x$. If x lies in an interval $[-2^{(l-2)/\epsilon - k}, 2^{(l-2)/\epsilon - k}]$, the prover can convince the verifier that $\log_g u = \log_h v$ in $\mathbb{Z}$ as follows. First the prover and the verifier engage in the (once and for all) set-up phase. The verifier randomly chooses two sufficiently large safe primes p_1 and p_2, and computes $n = p_1 p_2$. The verifier also chooses two random elements h_1 and h_2 from $\mathbb{Z}_n$, sends the prover n, h_1 and h_2, and proves that n indeed is the product of two safe primes. The prover checks whether $h_i \neq \pm 1 \mod n$ and $\gcd(h_i, n) = 1$ to convince that h_i has large order, where $i = 1, 2$. Then the prover randomly chooses $r \in_R \mathbb{Z}_n$, $r_1 \in \{-2^{l-2}, \cdots, 2^{l-2}\}$ and $r_2 \in \{-(n2^k)^\epsilon, \cdots, (n2^k)^\epsilon\}$, computes $\tilde{y} = h_1^r h_2^x \mod n$, $c = H(\mathbf{m}, g, h, u, v, \tilde{y}, g^{r_1}, h^{r_1}, h_1^{r_2} h_2^{r_1})$, $s_1 = r_1 - cx$ (in $\mathbb{Z}$), and $s_2 = r_2 - cr$ (in $\mathbb{Z}$). The prover sends the verifier $(\tilde{y}, \mathbf{m}, c, s_1, s_2)$ and the verifier accepts if and only if $c = H(\mathbf{m}, g, h, u, v, \tilde{y}, g^{s_1} u^c, h^{s_1} v^c, h_1^{s_2} h_2^{s_1} \tilde{y}^c)$ and $-2^{l-1} < s_1 < 2^{l-1}$.

D. Verifiable Encryption for Discrete Logarithms

The concept of verifiable encryption was first introduced by Stadler [35] in the context of publicly verifiable secret sharing schemes. Asokan et al. [4] extended the notion in a more general form for fair exchange and then Camenisch et al. [11] presented a formal definition for verifiable encryption.

Suppose Alice and Bob agree on a common value α^x, where α is a generator in a cyclic group $\mathbb{G}$. Alice wants to generate a verifiable encryption for x under the public key of a trusted third party T. Trivially, we assume that it is intractable to compute x from α^x in $\mathbb{G}$. Ateniese [1] presented a simple method of verifiable encryption for discrete logarithms as follows.

Consider the Naccache-Stern cryptosystem [32], let $n = pq$ be an RSA modulus which is generated by T along with a small integer B. Let σ be a square-free odd B-smooth integer such that it divides $\phi(n)$ and is prime to $\phi(n)/\sigma$ (a suggested size is $\sigma > 2^{160}$). Let g be an element whose multiplicative order modulo n is a large multiple of σ. A message $x < \sigma$ is encrypted by $g^x \mod n$. Decryption is performed using the prime factors of σ, getting x by the Chinese remainder theorem: Let p_i ($1 \leq i \leq k$) be the prime factors of σ. Given the ciphertext $c = g^x \mod n$, compute $c_i = c^{\frac{\phi(n)}{p_i}} = g^{\frac{x_i \phi(n)}{p_i}} \mod n$, where $x_i = x \mod p_i$ can be computed by the Pohlig-Hellman algorithm [33]. Using the Chinese remainder theorem, T can obtain the message x easily.

The verifiable encryption of x, given α^x, is performed by computing $g^x \mod n$ and showing that $\log_\alpha \alpha^x = \log_g g^x$ with the knowledge proof for the equality of discrete logarithms from different groups. For ease of notation, we denote by $\mathsf{VEDL}(x, \alpha^x, \mathbf{m}, T)$ the encryption of a discrete logarithm x for α^x under the public key of a third party T throughout the whole paper, where $\mathsf{VEDL}(x, \alpha^x, \mathbf{m}, T) = (g^x, \tilde{y}, \mathbf{m}, c, s_1, s_2)$ such that $-2^{l-1} < s_1 < 2^{l-1}$ and $c = H(\mathbf{m}, \alpha, g, \alpha^x, g^x, \tilde{y}, \alpha^{s_1}(\alpha^x)^c, g^{s_1}(g^x)^c, h_1^{s_2}h_2^{s_1}\tilde{y}^c)$. For more details, please refer to Section II-C.

III. Efficient Fair Conditional Payments

A. Security Model

In this section, we present an efficient fair conditional payment scheme. There are four participants in our scheme, an outsourcer O, a worker W, a bank $\mathcal{B}$, and a third party T. Without loss of generality, let O be a generic account-holder of $\mathcal{B}$, and $\mathcal{B}$ can issue O an electronic coin efficiently with a traditional electronic cash system ($\mathcal{B}$ requires neither the setting of conditional e-cash, nor the inefficient cut-and-choose protocol to generate the payment token in our solution). In the normal case, O and W fairly exchange the computation result and the payment token. In the case of disputes, T is involved in the protocol to ensure the fairness of the payments. We assume that T is not fully trusted, and may collude with one party to obtain profits at the expense of the other party. Actually, T just acts as a server in our proposed payments: it receives a request from O or W, updates its internal state and sends a response.

Similar to [3], [4], we assume that the communication channel between any two participants is *resilient*. The resilient channel assumption leads to an asynchronous communication model without global clocks, where messages can be delayed arbitrarily but with finite amount of time. In order to avoid the disputes, the deadline Time to finish the outsourcing computation task should include the time to be delayed in the communication channel.

The security properties of the proposed payments for outsourcing computations are defined in term of completeness, fairness, accountability.

- **Completeness:** It is infeasible for the adversary to prevent honest O and W from successfully obtaining the computation results and the payment token, respectively. The adversary can interact with T, but cannot interfere with the interaction of O and W, except insofar as the adversary still has the power to schedule the messages from O and W to T.
- **Fairness:** We consider a game between an adversary and an honest party. Generally, we let the adversary

play the role of the corrupt party, who completely controls the network, arbitrarily interacts with T, and arbitrarily delays the honest party's requests to T. In this sense, the fairness means that it is infeasible for the adversary O (or W) to obtain the computation results (or the payment token), while without allowing the honest party W (or O) to obtain the payment token (or the computation results).

- **Accountability:** T must be accountable for his dishonest actions, $i.e.$, it can be detected and proven if T misbehaves.

B. Main Idea

As mentioned earlier, the third trust problem in outsourcing computations is how O and W exchange the payment token and the computation results in a fair and efficient manner. We follow the paradigm of optimistic fair exchange and introduce a semi-trusted third party T in our proposed scheme. In the normal case, only O and W perform the Exchange protocol. In case of disputes, T is involved in the so-called Abort or Resolve protocols (either of them but not both). Compared with the traditional optimistic fair exchange protocols, the main difference is that the Abort (resp., Resolve) protocol can be executed only when the current time exceeds (resp., falls below) the deadline Time (not at any stage of the protocols).

On the other hand, a major difficulty in the fair payment scheme for outsourcing computations is how to generate the payment token efficiently. We use Brand's electronic cash system [8] for payment generation. The main trick is to encrypt partial information (e.g., r_2 in our construction) of the e-cash by using the verifiable encryption of discrete logarithms (VEDL). Note that VEDL consist of a (non-interactive) proof of knowledge for r_2, W can verify the validity of the payment token (while W cannot retrieve the token). Only when O obtains the computation results, W could retrieve the payment token from O.

Besides, neither the secret sharing/splitting scheme nor the cut-and-choose protocol is used for the generation or verification of the payment token. Hence, it is clear that our solution outperforms the existing schemes [18], [19] in terms of efficiency. The reasons are two folds: Firstly, the payment token is split into l shares to ensure their validity by using the cut-and-choose protocol in [18], [19], where l is the number of ringers. Therefore, the computation complexity for payment generation and verification is $O(l)$. However, the computation complexity of our scheme is only $O(1)$. Secondly, the cut-and-choose protocol is interactive and thus it requires at least 3 round of communications for payment verification, while VEDL is non-interactive and requires only 1 round of communications.

C. The Proposed Fair Conditional Payments

In this section, we present an efficient fair conditional payment scheme. We first present some notations before presenting our solution in detail. Denote by $\mathsf{Sig}(SK_X, M)$ the signature on message M with the secret key SK_X of the party $X \in \{O, W, T\}$; Denote by $\mathsf{VEDL}(x, \alpha^x, \mathbf{m}, T)$ the encryption of x for α^x under the public key of T as defined in section II-D.

- **Initialization**: The outsourcer O prepares the outsourcing instance F_i and the payment P_i, and then sends (F_i, P_i) to the worker W.
  - Job generation: O generates an instance of a job $F_i =< f, D_i, M_i >$, where $f : D \longrightarrow M$, and $D_i \subseteq D$. Assume that Time is the deadline to finish the job. O firstly chooses an integer $2m$, the total number of ringers. Moreover, O picks a random integer $t \in [m+1, \cdots, 2m]$ to be the number of true ringers. Trivially, $2m - t$ is the number of bogus ringers. The distribution of t in $[m+1, \cdots, 2m]$ is $d(t) = 2^{2m-t-1}$. O computes $y_j = f(x_j)$ for every true and bogus ringer x_j ($1 \leq j \leq 2m$). Let $M_i = \{y, y_1, y_2, \cdots, y_{2m}\}$, where $y = f(x)$ is the post-image whose pre-image O seeks. M_i is included in the screener S_i that is sent to W.
  - Payment generation: We will use Brand's electronic cash system [8] for payment generation. Let (g, g_1, g_2) be a random generator tuple of the group $\mathbb{G}$ with the prime order q. The secret/public key pair of $\mathcal{B}$ is $(\theta, \Theta = g^\theta)$. Define two collision-resistant hash functions $H : \mathbb{G}^5 \to \mathbb{Z}_q^*$, and $H_0 : \mathbb{G} \times \mathbb{G} \times ID \times time \to \mathbb{Z}_q$. When O opens an account at $\mathcal{B}$, $\mathcal{B}$ requests O to identify himself. O then generates at random a number $u_1 \in_R \mathbb{Z}_q$, and computes the unique account number $I = g_1^{u_1}$. If $g_1^{u_1} g_2 \neq 1$, then O transmits I to $\mathcal{B}$, and keeps u_1 secret. $\mathcal{B}$ computes and sends $z = (Ig_2)^\theta$ to O. Also, $\mathcal{B}$ stores the identifying information of O in the account database, together with I. The information I enables $\mathcal{B}$ to uniquely identify O in case he double-spends. When O wants to withdraw a coin, he first proves ownership of his account. To this end, the following withdrawal protocol between O and $\mathcal{B}$ is performed:

1) $\mathcal{B}$ generates a random number $\omega \in_R \mathbb{Z}_q$, and sends $a = g^{\omega}$, and $b = (Ig_2)^{\omega}$ to O.

2) O generates random numbers $s \in_R \mathbb{Z}_q^*$, $x_1, x_2, u, v \in_R \mathbb{Z}_q$ and computes $A = (Ig_2)^s, z' = z^s, a' = a^u g^v, b' = b^{su} A^v, B = g_1^{x_1} g_2^{x_2}$, and $c' = H(A, B, z', a', b')$. He then sends $c = u^{-1}c' \mod q$ to $\mathcal{B}$.

3) $\mathcal{B}$ responds with $r = c\theta + \omega \mod q$. If $g^r = \Theta^c a$ and $(Ig_2)^r = z^c b$, then $A, B, (z', c', r')$ is a valid coin (payment token) of which O knows a representation, where $r' = ru + v \mod q$.

- **Verification**: Let $d = H_0(A, B, ID_W, \mathsf{Time})$, O firstly computes $r_1 = d(u_1 s) + x_1 \mod q$, and $r_2 = ds + x_2 \mod q$. O then sends $A, B, (z', c', r')$, r_1, $g_2^{r_2}$, and $\mathsf{VEDL}(r_2, g_2^{r_2}, F_i, S_i, T)$ to W. If and only if $A \neq 1$, $c' = H(A, B, z', g^{r'} y^{-c'}, A^{r'} z'^{-c'})$, $g_1^{r_1} g_2^{r_2} = A^d B$, and $\mathsf{VEDL}(r_2, g_2^{r_2}, F_i, S_i, T)$ is a valid verifiable encryption of r_2, W accepts the payment token. In the following, we use the abbreviated notation $\mathsf{VEDL}(r_2)$ instead of $\mathsf{VEDL}(r_2, g_2^{r_2}, F_i, S_i, T)$ for simplicity.

- **Computation**: For each input $x \in D_i$, W computes $f(x)$ and then uses the screener S_i to output x if $f(x) \in M_i$. Let the set of all elements x be Sol.

- **Payment**: O and W are involved in a protocol to exchange Sol and the payment token in a fair manner. The protocol consists of three sub-protocols: $\mathsf{Exchange}$, Abort, and $\mathsf{Resolve}$. In the normal case, only the $\mathsf{Exchange}$ protocol is executed. The other two protocols are used only if O or W misbehaves. If W cannot present Sol to O before Time, the Abort protocol is executed in order to cancel or spend the payment token by O. If O rejects to pay W after he obtains Sol before the deadline Time, W can perform the $\mathsf{Resolve}$ protocol to retrieve the payment token with the help of T.

  - $\mathsf{Exchange}$: A protocol between O and W if both parties are honest.

    1) W sends Sol to O before Time.

    2) If Sol is the valid solution for F_i, O sends r_2 and the corresponding signature $\mathsf{Sig}(SK_O, r_2)$ to W.

    3) W sends $A, B, (z', c', r')$, d, r_1, and r_2 to $\mathcal{B}$. $\mathcal{B}$ first checks the validity of the payment token, and then searches its deposit database to find out whether A has stored before. If A has not stored

before, $\mathcal{B}$ stores (A, d, r_1, r_2) in its database and credits the account of W; Else, $\mathcal{B}$ can detect the double-depositing (the same d) or the double-spending (the different d).

- Abort: A protocol between O and T if O fails to obtain Sol after the deadline Time.

  1) O firstly computes the signature $\mathsf{Sig}(SK_O, \mathrm{abort}||\mathsf{VEDL}(r_2))$ and then sends

  $$(\mathsf{VEDL}(r_2), \mathsf{Sig}(SK_O, \mathrm{abort}||\mathsf{VEDL}(r_2)))$$

  to T.

  2) If the current time falls below Time, the protocol is terminated. If the signature is valid and W has not resolved, T then uses the secret key SK_T to issue an abort-token

  $$\mathcal{AT} = \mathsf{Sig}(SK_T, \mathsf{Sig}(SK_O, \mathrm{abort}||\mathsf{VEDL}(r_2)))$$

  to O and stores it. The abort token is not a proof that the exchange has been aborted, but a guarantee by T that it has not and will not execute the $\mathsf{Resolve}$ protocol.

- $\mathsf{Resolve}$: A protocol among W, T and O if O obtains Sol while rejects to pay W before the deadline Time.

  1) W firstly computes the signature $\mathsf{Sig}(SK_W, \mathrm{resolve}||\mathsf{VEDL}(r_2))$ and sends

  $$(\mathsf{VEDL}(r_2), \mathsf{Sig}(SK_W, \mathrm{resolve}||\mathsf{VEDL}(r_2)))$$

  and Sol to T.

  2) If the current time exceeds Time, the protocol is terminated. Otherwise, if and only if the signature and Sol are both valid, T computes r_2 and sends $(r_2, \mathsf{Sig}(SK_T, r_2))$ to W. W can verify the validity of r_2 and deposit the payment token.

  3) T sends Sol to O.

- **Cancellation/Spending**: After O has performed the Abort protocol with T, O can cancel or spend the payment token as follows: O sends $A, B, (z', c', r')$, $d' = H_0(A, B, ID_S, time)$, $r'_1 = d'(u_1 s) + x_1 \mod q$, and $r'_2 = d's + x_2 \mod q$ to an intended shop S (Specially, if O also acts as the role of S, then the payment token is canceled as [18]).

Remark 1. The proposed scheme is not based on some specific e-cash systems such as a conditional e-cash [17], [36], but a traditional e-cash system. The main distinctions between traditional and conditional e-cash are given in [6]: Firstly, a payer can anonymously transfer an e-coin to an *anonymous* payee in conditional payments. However, in traditional e-cash systems, the coin is normally bound to the identity of the merchant during the spending. Trivially, note that $d = H_0(A, B, ID_W, \mathsf{Time})$ in our scheme, hence the payment token is bound to the identity of W. Secondly, in conditional e-cash systems, a payer should have the ability to cash back the payment in case of an unfavorable outcome of the condition to the payee (the anonymity of the payer cannot be ensured since the bank credits the account of the payer), while in traditional e-cash systems, the only way to (anonymously) spend a coin is to through a merchant. Nevertheless, in our scheme, a payee can either spend the coin through a merchant or cash back the payment in an indistinguishable way.

Remark 2. If the double-spending is detected, $\mathcal{B}$ can compute $(r_1 - r_1')/(r_2 - r_2')$ to trace O. In this case, O provides

$$(\mathcal{AT}, r_2, \mathsf{VEDL}(r_2), \mathsf{Sig}(SK_O, \text{abort}||\mathsf{VEDL}(r_2)))$$

to $\mathcal{B}$. If the verifications hold, $\mathcal{B}$ summons W to present the proof. If W can provide the signature $\mathsf{Sig}(SK_O, r_2)$, $\mathcal{B}$ can deduce that O is the double-spender; Else if W can provide the signature $\mathsf{Sig}(SK_T, r_2)$, $\mathcal{B}$ can deduce that T misbehaves (This means that T performed both the Abort and $\mathsf{Resolve}$ protocols).

Remark 3. In the **Payment** procedure, we introduced a semi-trusted third-party T. This is different from [18], [19]. However, we argue that T is involved in the protocol only in the case of disputes and it is essential to solve the third trust problem. Besides, in the $\mathsf{Resolve}$ protocol, T needs to verify the validity of the solution Sol. Since W sends $\mathsf{VEDL}(r_2)$ to T and $\mathsf{VEDL}(r_2)$ is the abbreviated notation of $\mathsf{VEDL}(r_2, g_2^{r_2}, F_i, S_i, T)$, T clearly knows what the computation task F_i and the screener S_i are and O cannot provide a different M_i set to fool T. Besides, T (as same as O) can efficiently verify the validity of Sol.

Remark 4. Blanton [6] proposed an improved conditional e-cash based on CL-signature with protocols [12] and verifiable encryptions [11], which does not require the expensive cut-and-choose protocol and thus is more efficient than the scheme [36]. Besides, the scheme [6] assumes that the publisher (*i.e.,* the third party) is *trusted* to correctly publish the outcome of the event and any other information associated with it. Therefore, it does not consider the case that the dishonest publisher may collude with the payer or payee. On the other hand, our solution uses the e-cash system based on blind signatures, while the scheme [6] uses the e-cash system based on CL-signature with protocols and thus requires some additional proofs of knowledge in the conditional transfer stage.

Remark 5. The above solution cannot ensure the full anonymity of O since the signature $\mathsf{Sig}(SK_O, r_2)$ on r_2 is given to W in the $\mathsf{Exchange}$ protocol. We propose an improved $\mathsf{Exchange}$ protocol to achieve the anonymity as follows:

1) W sends Sol to O before Time.
2) If Sol is the valid solution for F_i, O sends $d^* = H_0(A, B, ID_W, time, \mathsf{Time})$, $r_1^* = d^*(u_1 s) + x_1 \mod q$, and $r_2^* = d^* s + x_2 \mod q$ to W, where $time$ is the number representing time of the transaction.
3) W sends $A, B, (z', c', r')$, d^*, r_1^*, and r_2^* to $\mathcal{B}$. $\mathcal{B}$ first checks the validity of the payment token, and then searches its deposit database. If A has not stored before, $\mathcal{B}$ stores $(A, ID_W, time, \mathsf{Time}, d^*, r_1^*, r_2^*)$ in its database and credits the account of W; Else, $\mathcal{B}$ can detect the double-depositing (the same ID_W and Time) or the double-spending (otherwise). More precisely, if both of the tuples $(A, B, (z', c', r'), ID_W, time, \mathsf{Time}, r_1^*, r_2^*)$ and $(A, B, (z', c', r'), ID_W, \mathsf{Time}, r_1, r_2, \mathsf{Sig}(SK_T, r_2))$ are presented to $\mathcal{B}$, then the double-depositing is detected. This prevent the dishonest W from obtaining (d^*, r_1^*, r_2^*) from O and $(d, r_1, r_2, \mathsf{Sig}(SK_T, r_2))$ from T simultaneously.

Besides, if W performs the $\mathsf{Resolve}$ protocol to obtain $(r_2, \mathsf{Sig}(SK_T, r_2))$ from T, then he can send $A, B, (z', c', r')$, d, r_1, r_2, and $\mathsf{Sig}(SK_T, r_2)$ to $\mathcal{B}$. If and only if $A \neq 1$, $c' = H(A, B, z', g^{r'} y^{-c'}, A^{r'} z'^{-c'})$, $g_1^{r_1} g_2^{r_2} = A^d B$, and $\mathsf{Sig}(SK_T, r_2)$ is valid, $\mathcal{B}$ accepts the payment token and then searches its deposit database to find out whether A has stored before. If A has not stored before, $\mathcal{B}$ stores $(A, ID_W, \mathsf{Time}, r_1, r_2, \mathsf{Sig}(SK_T, r_2))$ in its database and credits the account of W; Else, $\mathcal{B}$ can detect the double-depositing (the same ID_W and Time) or the double-spending (otherwise).

After the double-spending is detected, O provides $(\mathcal{AT}, r_2, \mathsf{VEDL}(r_2), \mathsf{Sig}(SK_O, \mathrm{abort}||\mathsf{VEDL}(r_2)))$ to $\mathcal{B}$. If the verifications hold, $\mathcal{B}$ can deduce that T misbehaves (This means that T performed both the Abort and $\mathsf{Resolve}$ protocols). Otherwise, $\mathcal{B}$ can deduce that O is the double-spender.

IV. Security Analysis

In this section, we present the security analysis of the proposed fair conditional payments. As mentioned before, we assume that the outsourcer O and the rational "lazy-and-partially-dishonest" worker W do not trust each other. Besides, we assume that the third party T is not fully trusted and may collude with O or W.

Theorem 4.1: The proposed conditional payments satisfies the property of completeness.

Proof: If both O and W are honest, they will successfully perform the $\mathsf{Exchange}$ protocol and then obtain Sol and the payment token (note that W actually obtains the value r_2 of the payment token), respectively. ■

Theorem 4.2: The proposed conditional payments satisfies the property of fairness.

Proof: We first prove the fairness for O. Let us consider the game that an honest O is playing against a dishonest W. We say that W wins the game if and only if W obtains the payment token while O does not obtain Sol before Time. Assume that O does not obtain Sol before Time, then W cannot obtain the payment token from O. Therefore, W must successfully run the $\mathsf{Resolve}$ protocol with T in order to obtain the payment token. However, O can also obtain Sol from T and this deduces a contradiction. Therefore, the successful probability for B to win the game is negligible.

We then prove the fairness for W. Consider an honest W playing against a dishonest O. We say that O wins the game if and only if O obtains Sol before Time while W does not obtain the payment token. Similarly, we assume that W does not obtain the payment token from O. If W does not complete the entire job before Time, then O cannot obtain Sol either. Otherwise, W can successfully run the $\mathsf{Resolve}$ protocol and obtain the payment token from T, which contradicts the assumption. Therefore, the successful probability for O to win the game is negligible. ■

Theorem 4.3: The proposed conditional payments satisfies the property of accountability.

Proof: Suppose that T performs the $\mathsf{Resolve}$ protocol with W and cannot sends the computation results Sol before Time. There are two cases for this event: 1. W does not complete the computation task before Time and colludes with T; 2. W sends Sol to T while T does not send it to O. In any case, O must perform the Abort protocol to obtain the abort token from T. This means that O can successfully cancel or spend the payment token. If the double-spending is detected, O can provide the abort token while W cannot provide the signature of O for r_2. Then, B can deduce that T misbehaves since both $\mathsf{Resolve}$ and Abort protocols are successfully performed by T. As a result, the equal amount of the e-cash is decreased in the account of T. ■

Theorem 4.4: The proposed conditional payments can solve all the three trust problems.

Proof: Firstly, due to the idea of ringers, the probability for W to obtain the Sol without completing the entire computation task is negligible. Secondly, W can undoubtedly obtain the payment token after he sends Sol to O or T. Finally, O can cancel or spend the payment token if he cannot obtain the Sol before Time. Therefore, all the three trust problems can be solved. ■

V. Conclusions

In this paper, we first assume the rational lazy-and-partially-dishonest workers in the distributed outsourcing computations, and introduce a new trust problem of obtaining the computation result from the dishonest workers. Moreover, we propose a new fair conditional payment scheme that can solve all the trust problems between the outsourcer and the workers. Compared with the existing solutions [18], [19], the proposed solution is much more efficient for real applications since neither the secret sharing/splitting scheme nor the cut-and-choose protocol is used for the generation or verification of the payment token.

Acknowledgement

We are grateful to the anonymous referees for their invaluable suggestions. This work is supported by the National Natural Science Foundation of China (Nos. 60970144 and 61100224), China 111 Project (No. B08038), and the Fundamental Research Funds for the Central Universities (Nos. K50510010003, K50511010001 and JY10000901034). The third author is supported by the ARC Future Fellowship (FT0991397).

References

[1] G. Ateniese, *Verifiable encryption of digital signatures and applications*, ACM Transaction on Information and System Security, 7(1), ACM Press, pp.1-20, 2004.

[2] N. Asokan, M. Schunter, M. Waidner, *Optimistic protocols for fair exchange*, ACM Conference on Computer and Communications Security, ACM press, pp.6-17, 1997.

[3] N. Asokan, V. Shoup, and M. Waidner, *Asynchronous protocols for optimistic fair exchange*, IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, pp.86-99, 1998.

[4] N. Asokan, V. Shoup, and M. Waidner, *Optimistic fair exchange of digital signatures*, Advances in Cryptology-Eurocrypt 1998, LNCS 1403, Springer, Berlin Heidelberg New York, pp.591-606, 1998.

[5] N. Asokan, V. Shoup, and M. Waidner, *Optimistic fair exchange of digital signatures*, IEEE Journal on Selected Areas in Communications, 18(4), pp.593-610, 2000.

[6] M. Blanton, *Improved conditional e-payments*, ACNS 2008, LNCS 5037, Springer-Verlag, pp.188-206, 2008.

[7] M. Blum, *How to exchange (secret) keys*, ACM Transactions on Computer Systems, 1(2), pp.175-193, 1983.

[8] S. Brands, *Untraceable Off-line cash in wallet with observers*, Advances in Cryptology-Crypto 1993, LNCS 773, Springer-Verlag, pp.302-318, 1993.

[9] B. Baum-Waidner, *Optimistic asynchronous multi-party contract signing with reduced number of rounds*, ICALP 2001, LNCS 2076, Springer, Berlin Heidelberg New York, pp.898-911, 2001.

[10] F. Bao, R. Deng, W. Mao, *Efficient and practical fair exchange protocols with off-Line TTP*, IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, pp.77-85, 1998.

[11] J. Camenisch, and I. Damgård, *Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes*, Advances in Cryptology-Asiacrypt 2000, LNCS 1976, Springer-Verlag, pp.331-345, 2000.

[12] J. Camenisch, A. Lysyanskaya, *A signature scheme with efficient protocols*, SCN 2002. LNCS 2576, Springer-Verlag, pp.268-289, 2003.

[13] J. Camenisch, A. Lysyanskaya, and Meyerovich, *Endorsed E-cash*, Proceeding of the IEEE symposuim on Secrity and Privacy, IEEE, pp.101-115, 2007.

[14] D. Chaum and T. Pedersen, *Wallet databases with observers*, Advances in Cryptology-Crypto 1992, LNCS 740, Springer-Verlag, pp.89-105, 1993.

[15] J. Camenisch, M. Michels, *Separability and efficiency for generic group signature schemes*, Advances in Cryptology-Crypto 1999, LNCS 1666, Springer-Verlag, pp.413-430, 1999.

[16] B. Carbunar and R. Sion, *Uncheatable reputation for distributed computation markets*, FC 2006, LNCS 4107, Springer-Verlag, pp.96-110, 2006.

[17] B. Carbunar, L. Shi, and R. Sion, *Conditional e-payments with transferability*, Journal of Parallel Distributed Computing, 71(1), Elsevier, pp.16-26, 2011.

[18] B. Carbunar, M. Tripunitara, *Conditioal Payments for Computing Markets*, CANS 2008, LNCS 5339, Springer-Verlag, pp.317-331, 2008.

[19] B. Carbunar, M. Tripunitara, *Fair Payments for Outsourced Computations*, Proceedings of the 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2010), IEEE, pp.529-537, 2010.

[20] X. Chen, F. Zhang, H. Tian, Q. Wu, Y. Mu, J. Kim, and K. Kim, *Three-round Abuse-free Optimistic Contract Signing With Everlasting Secrecy*, FC 2010, LNCS 6052, Springer-Verlag, pp.304-311, 2010.

[21] I. Damgård, *Practical and provably secure release of a secret and exchange of signatures*, Journal of Cryptology, 8(4), Springer, Berlin Heidelberg New York, pp.201-222, 1995.

[22] W. Du, J. Jia, M. Mangal, and M. Murugesan, *Uncheatable grid computing*, Proceedings of the 24th International Conference on Distributed Computing Systems, pp.4-11, 2004.

[23] R. H. Deng, L. Gong, A. Lazar, and W. Wang, *Practical protocol for certified electronic mail*, Journal of Network and Systems Management, 4(3), pp.279-297, 1996.

[24] S. Even, O. Goldreich, and A. Lempel, *A randomized protocol for signing contracts*, Communications of ACM, 28(6), 1985, pp.637-647.

[25] M. Franklin, M. Reiter, *Fair exchange with a semi-trusted third party*, ACM Conference on Computer and Communications Security, ACM press, pp.1-5, 1997.

[26] J.A. Garay, M. Jakobsson, and P. MacKenzie, *Abuse-free optimistic contract signing*, Advances in Cryptology-Crypto 1999, LNCS 1666, Springer, Berlin Heidelberg New York, pp.449-466, 1999.

[27] P. Golle, I. Mironov, *Uncheatable distributed computations*, CT-RSA 2001, LNCS 2020, Springer-Verlag, pp.425-440, 2001.

[28] P. Golle, S.G. Stubblebine, *Secure distributed computing in a commercial environment*, FC 2001, LNCS 2339, Springer-Verlag, pp.289-304, 2001.

[29] Q. Huang, G. Yang, D. Wong, and W. Susilo, *Ambiguous optimistic fair exchange*, Advances in Cryptology-ASIACRYPT 2008, LNCS 5350, Springer, Berlin Heidelberg New York, pp.74-89, 2008.

[30] T. Kleinjung, K. Aoki, J. Franke, A. Lenstra, E. Thomé, J. Bos, P. Gaudry, A. Kruppa, P. Montgomery, D. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann, *Factorization of a 768-bit RSA modulus*, Advances in Cryptology-Crypto 2010, LNCS 6223, Springer, Berlin Heidelberg New York, pp.333-350, 2010.

[31] F. Monrose, P. Wyckoff, and A. Rubin, *Distributed execution with remote audit*, In Proc. of the Network and Distributed System Security Symposium, pp.103-113, 1999.

[32] D. Naccache and J. Stern, *A new public key cryptosystem based on higher residues*, The Fifth ACM Conference on Computer and Communications Security, ACM Press, pp.59-66, 1998.

[33] S.C. Pohlig and M.E. Hellman, *An improved algorithm for computing logarithms over $GF(p)$ and its crytographic significance*, IEEE Transactions on Information Theory, vol.IT-24-1, pp.106-110, 1978.

[34] L.F.G. Sarmenta, *Sabotage-tolerance mechanisms for volunteer computing systems*, Future Generation Computer Systems: Special Issue on Cluster Computing and the Grid, Vol.18, pp.561-572, 2002.

[35] M. Stadler, *Publicly verifiable secret sharing*, Advances in Cryptology-Eurocrypt 1996, LNCS 1070, Springer-Verlag, pp.191-199, 1996.

[36] L. Shi, B. Carbunar, and R. Sion, *Conditional E-Cash*, FC 2007, LNCS 4886, Springer-Verlag, pp.15-28, 2007.

[37] D. Szajda, B. Lawson, J. Owen, *Hardening functions for large-scale distributed computations*, Proceedings of IEEE Symposium on Security and Privacy, pp.216-224, 2003.

[38] D. Szajda, B. Lawson, J. Owen, *Toward an optimal redundancy strategy for distributed computations*, Proceedings of the 2005 IEEE International Conference on Cluster Computing, pp.1-11, 2005.

[39] J. Zhou, and D. Gollmann, *A fair non-repudiation protocol*, IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, pp.55-61, 1996.