4P_VES: A Collusion-Resistant Accountable Virtual Economy System

Hong Zhang[1], Xiaolei Dong[2(✉)], Zhenfu Cao[2(✉)], and Jiachen Shen[1]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
{zefan,jiachenshen}@sjtu.edu.cn

[2] Shanghai Key Lab for Trustworthy Computing, East China Normal University, Shanghai 200062, China
{dongxiaolei,zfcao}@sei.ecnu.edu.cn

Abstract. Virtual economy develops rapidly and accounts for quite a large proportion in the entire economy. Markets of virtual goods, such as games, apps and cloud services, are quite active and contribute a lot to the revenue of platforms and providers. The existing virtual economy systems mainly rely on the trustworthiness of the platforms who maintain the accounts of all participants, which is short of transparency. Another concern is how to maintain the market order when conspiracy between participants happens. In this paper, we extend the Verito scheme (NDSS 2013) and introduce a new participant Payment to obtain our 4P_VES scheme, in which collusion between two parties can be detected while satisfying required properties of transparency and accountability at the same time. We also analyze the properties and prove the security of our scheme. Finally, we evaluate the additional cost compared with Verito and find that our scheme is a cost affordable and practical one which enhances the system's independency and security.

Keywords: Virtual economy · E-coin · Homomorphic commitment · Dynamic accumulator

1 Introduction

With the rapid development of Internet, network economy expands to nearly all kinds of market segments and accounts for a large proportion in the entire economy. The recent years have seen the boom of transactions of virtual goods and services, including online games, apps and cloud services. According to Face book annual report 2013, payments revenue, generated almost exclusively from game applications, in the fourth quarter is $241 million. It is the largest source of revenue after advertising [1]. For cloud services, it is estimated that the Amazon Web Service hits $3.8 billion revenue in 2013. In China, Ten cent, one of the leading providers of internet value-added services, declares that the revenue of virtual goods and services, mainly contributed by online games, reached 11,972 million Yuan, or about 2 billion dollars, in the fourth quarter of 2013, accounting for 70 percent in the total revenue [2].

L.C.K. Hui et al. (Eds.): ICICS 2014, LNCS 8958, pp. 61–73, 2015.
DOI: 10.1007/978-3-319-21966-0_5

Usually these virtual economies are carried out on platforms using virtual currencies rather than real ones. Due to the high frequency and small-valued size of virtual transactions, it is inconvenient for both sides to use a real currency. Thus the platforms issue virtual currencies, which can be sold in bulk, to perform the function of circulation. There are a number of virtual currencies now, including Face book credits, Q-coins (issued by Ten cent), U-coins (issued by Sina, a top portal in China), etc. Due to the lack of strong supervision and effective regulation in virtual economy, platforms are able to issue virtual currency without any limit. This makes the exchange rate from virtual coins to real currency unpredictable. Additionally, the virtual economy is globally oriented, making the issue more complicated. Because of the very divergent pricing policies and floating exchange rates from region to region, the possibility of arbitrage should also be taken into consideration.

Raghav Bhaskar *et al.* proposed a solution Verito [8] in NDSS 2013 which provides four properties viz., transparency, fairness, non-repudiation and scalability to address the issue mentioned above. The main idea of their scheme is to make a commitment which binds a virtual coin with a price in real currency. [8] assumes that no two of the three parties (i.e. Platform, Merchant and User) will collude to compromise the third. However, according to the property of fairness they mentioned, the merchants cannot distinguish between any two coins even if they are of different real values, so the User does not have incentive to check the coin's real value. This leaves room to the conspiracy between Platform and User. Thus it is a necessity to propose an anti-collusion scheme.

Our Contribution: We modify the Verito to obtain our own solution, in which collusion can be detected while supporting the essential properties at the same time. As the majority of transactions take place online and third-party payments such as Paypal, Alipay are widely adopted, we introduce the third-party payment into our system to defend against collusion. We propose 4P_VES, a virtual economy system composed of four parties (Platform, Provider, Player and Payment) which provides transparency and accountability even allowing the existence of collusion. We also analyze the security as well as the feasibility in our paper.

In summary, this paper makes three main contributions:

- First, we identify the possibility of collusion between Platform and Player in virtual economy system as a realistic threat;
- Second, we modify Verito to obtain an anti-collusion system 4P_VES with the combination of cryptographic primitives: commitment and accumulator.
- Finally, we discuss the properties of our system and the security under concrete assumptions.

The rest of the paper is organized as follows: We describe the system and define the model in Sect. 2. Section 3 gives the preliminaries including cryptographic primitives and assumptions. Section 4 presents the concrete construction of our scheme. Properties and security are analyzed in Sect. 5. We evaluate its performance and compare it with different schemes in Sect. 6. We review related work in Sect. 7 and finally conclude in Sect. 8.

2 System Model

4P_VES consists of four parties: Platform, Provider, Payment and Player. (4P refers to these four parties whose names all start with P). The Platform (Face book, Ten cent etc.) provides an ecosystem for the transaction. It maintains the resources of system and deals with both the Provider and the Player. The Provider is the company or individual who is registered with Platform and provides virtual goods or services. Online games and application vendors are examples of the Provider, such as King, Wooga (registered in Face book). Players are the customers who buy virtual goods and services with virtual currency. And Payment (Paypal, Alipay, etc.) is the third party in charge of pay and encashment.

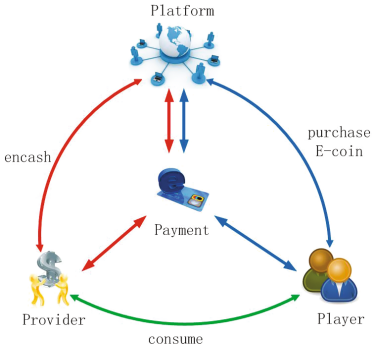

Fig. 1. Participants and interactions

Virtual currency, referred as e-coin in this paper, is adopted during the transaction. Platform issues e-coins and maintains the virtual accounts of both Provider and Player. Player purchases e-coins in bulk from Platform and pay with real currency through Payment. Player spends e-coins to buy virtual goods and services, such as apps and in-game props supplied by Provider. And Provider offers virtual goods or services in exchange for e-coins which can be encashed in bulk from Platform through Payment later. Payment plays a vital role in our system and is responsible for payment and encashment. The additional duty of Payment is to supervise the behavior of Platform and Player, checking whether there is conspiracy. It is reasonable to introduce the Payment party in our scheme. First, third party payment such as Paypal and Alipay are widely adopted in real world. Second, Payment is much more reliable than common users. Considering the large number of users registered, it is infeasible to verify and regulate all the users' behavior. Third party payment is under the public and government's supervision and is much easier for regulation. Furthermore, third party payment reduces the computation burden of clients, which makes the system more practical since the clients tend to use terminals with limited computational capacity. Therefore, the party, Payment, is introduced in our system.

The main interactions of the virtual economy (Fig. 1) are listed as follows.

(1) Purchase e-coins (Participants: Player, Platform, Payment):

- Player chooses the quantity and type of e-coins to buy from Platform.
- Platform generates the e-coins requested by Player.
- Payment verifies whether the real value bound to the e-coins is equal to the posted price. If the verification succeeds, Payment authenticates the e-coins by a signature scheme and transfers real currency from Player's account to Platform's. Otherwise, it rejects the Purchase action.
- Player receives e-coins together with the signature of Payment.

(2) Consume (Participants: Player, Provider, Platform):

- Player chooses the products (including goods and services) to buy and sends the e-coins as posted price to Provider.
- Provider checks the e-coins whether they are signed by Payment. Provider then generates a Transaction No. (TNO) and sends it back to Player.
- Player sends the TNO together with the e-coins to Platform.
- Platform verifies the e-coins, transfers the e-coins from Player's virtual account to Provider's and sends back a receipt to Player.
- Player forwards the receipt to Provider to get the products.

(3) Encash (Participants: Provider, Platform, Payment):

- Provider sends the e-coins to Platform to exchange for real currency.
- Platform checks the authenticity of the e-coins. If it is valid, Platform produces a proof that convinces the Provider that the aggregate value of the cashed e-coins is v without revealing the value of any individual e-coin.
- Provider reviews the proof.
- Platform removes the e-coins from Provider's virtual account.
- Payment transfers real currency from Platform's account to the Provider's.

3 Preliminaries

As our scheme follows [8] closely, the same modern cryptographic primitives, commitment and dynamic accumulator, are employed. In addition, to obtain a collusion-free system, we adopt another much more familiar primitive, digital signature, which is omitted here to save space.

3.1 Commitment

A commitment scheme includes two main phases: committing and revealing. Vividly, it can be thought of as analogous to an envelope. When a user Alice wants to form a commitment to a certain value, she puts the value into the envelope and seals it. The envelope is sealed in a special way that no one except Alice can open it. Then the envelope can be sent as a commitment to another user Bob. When it is time to reveal the value, Alice opens the envelope. There

are two properties of commitment: hiding and binding. The hiding property: As the value is sealed in the envelope, Bob cannot learn any information about the value before Alice reveals it. The binding property: Because the envelope is sealed, Alice cannot change the value in the envelope without Bob's notice. We denote the commitment of value m as $c = Com(m)$ and the revealing phase as $OpenCom(c, r, m)$, r is the randomness used to form the commitment.

One more property required in our commitment scheme is additive homomorphism, i.e. $Com(m_1 + m_2) = Com(m_1) \bigodot Com(m_2)$. There are several homomorphic commitment schemes such as [3,12,14,15]. To best suit our purpose, we adopt the Pedersen commitment [15], which relies on the security of the Discrete Logorithm assumption. Following is the main construction of [15].

$ComSetup()$: generate large primes p and q such that $q|(p-1)$. Let $\mathbb{G}_q \preceq \mathbb{Z}_p^*$, $|\mathbb{G}_q| = q$. Randomly choose a generator $g \in \mathbb{G}_q$ and an element $h \in \mathbb{G}_q$ such that $log_g h$ is unknown.

$Com(m)$: to commit an element $m \in \mathbb{Z}_q$, choose $r \in \mathbb{Z}_q$ at random and compute $c = g^m h^r$.

$OpenCom(c, r, m)$: output 1 if $g^m h^r \equiv c$, otherwise 0.

3.2 Dynamic Accumulator

An accumulator scheme, introduced by Benaloh and de Mare [7], allows aggregation of a large set of values into one constant size value. Camenisch and Lysyanskaya extended the accumulator to dynamic accumulator (DA), where the cost of adding or deleting elements from the accumulator and witness updating do not depend on the size of the accumulated set [5,10]. Here is the construction of [4,10] adopted in our implementation.

$\mathcal{F}_k$ is a family of functions that correspond to exponentiating modulo safe-prime products with length k. $f_n \in \mathcal{F}_k$, $f_n(u, x) = u^x \mod n$, $n = pq$, $p = 2p' + 1$, $q = 2q' + 1$, where p, p', q, q' are all prime, $x \in \mathbb{X} = \{a \mid a\ is\ prime,\ a \neq p', q'\}$.

The accumulator value will be updated when an element is added or deleted. Adding an element $\tilde{x} \in \mathbb{X}$ to the accumulator v:

$$v' = AccAdd(v, \tilde{x}) = f_n(v, \tilde{x}) = v^{\tilde{x}} \mod n;$$

Deleting an element $\tilde{x}$ from the accumulator v:

$$v' = AccDel(v, \tilde{x}) = D((p, q), v, \tilde{x}) = v^{\tilde{x}^{-1} \mod (p-1)(q-1)} \mod n.$$

For each element x accumulated in v, there is a corresponding witness w such that $w^x \equiv v$.

$$AccVerify(v, x) = 1,\ iff\ w^x \equiv v \mod n.$$

Updating the witness w of x after $\tilde{x}$ has been added:

$$w' = f_n(w, \tilde{x}) = w^{\tilde{x}} \mod n;$$

Updating the witness w of x after $\tilde{x}(\tilde{x} \neq x)$ has been deleted:

$$w' = w^b v'^a \mod n,\ ax + b\tilde{x} = 1.$$

And multi-elements can be added or deleted at once. Let π_a be the product of elements to be added while π_d be the product of ones to be deleted. Then

$$v' = v^{\pi_a \pi_d^{-1} \mod (p-1)(q-1)} \mod n;$$

$$w' = (w^{\pi_a})^b v'^a \mod n,\ ax + b\pi_d = 1.$$

3.3 Cryptographic Assumptions

The construction of our scheme is based on the following assumptions.

Discrete Logarithm (DL) Assumption: given a group $\mathbb{G}$ with generator g, $|\mathbb{G}| = p$, for all non-uniform probabilistic polynomial-time (PPT) algorithms $\mathcal{A}$,

$$\Pr_{h \leftarrow \mathbb{G}}[\mathcal{A}(h) = \log_g h] \text{ is negligible}$$

As the Pohlig-Hellman algorithm can solve the DL problem efficiently in the case when $p-1$ is a product of small primes, usually p is required to be a safe prime ($p = 2q + 1$, q is prime).

Strong RSA Assumption: given a modulus n of unknown factorization and a ciphertext c, it is infeasible to find a pair (m, e) such that $c = m^e \mod n$.

4 Scheme Construction

This section details the key parts of our system using the cryptographic primitives mentioned above. Other familiar primitives such as symmetric encryption and signature are also used. To best adapt our scheme, we adopt DES as $E_k(r, m)$ while RSA signature is used in function $sign()$.

(1) Setup
Platform sets the system parameters, generates a signing $keypair(vk, sk)$ and a symmetric key k to encrypt the commitment's open key which should be hidden from the Provider. It also runs the $ComSetup()$ to set the parameters used in commitment: p, q, g, h and $Accgen()$ to initiate the registered Players' and Providers' accumulator values.
(2) E-coin construction
There is a list mapping the e-coin's nominal value to its real value (including the currency type and amount) published by the Platform (Shown in Table 1). $E\text{-}coin = (Com(m), E_k(r, m))$ where $Com(m) = g^m h^r$, and r is chosen randomly from $\mathbb{Z}_q$.
(3) Purchase e-coins
Player chooses the type and quantity of e-coins to buy. Platform searches the corresponding Identity Number m_i of the chosen e-coin type and calls the

Table 1. Example of E-coins list

ID NO	E-coin Type	Real Price
m_1	Type1	\$0.1
m_2	Type2	£0.6
m_3	Type3	¥10
...	...	...

Table 2. Number of rounds of interactions

rounds \ schemes / phases	Verito	4P_VES	Variant
Purchase	2	4	3
Consume	5	7	6
Encash	2	4	3

construction function to generate the required e-coins. Then the e-coins are sent together with the open keys to Payment. Payment opens the commitments (using the homomorphic property here) to check the commited value and sends them to Player after signing. It also transfers corresponding amount of real currency from Player's account to Platform's. The Platform updates the accumulator of the Player's e-coins by *AccAdd* and sends the accumulator value to Player. Player updates all of his e-coins' witness, signs the accumulator value and sends back the signed value to Platform as a receipt. The detail is showed in Algorithm 1.

Algorithm 1. Purchase e-coins

Require: The type $type_i$ and number $\#num$ of e-coins to buy.

Platform

$m_i \leftarrow search(type_i)$ // search the ID of the $type_i$ e-coin
$ECOINS \leftarrow \emptyset$
for $k = 1$; $k <= \#num$; $k++$ **do**
  $Com(m_i) = g^{m_i} h^{r_i}$;
  $E_k(m_i, r_i)$;
  $coin_k \leftarrow (Com(m_i), E_k(m_i, r_i))$
  $\sigma(coin_k) = sign_{sk}(coin_k)$
  $ECOINS = ECOINS \cup \{coin_k\}$
end for
$C = \prod_{i=1}^{\#num} Com(m_i)$
$Openkey = \sum_{i=1}^{\#num} r_i$
$v_{id} = AccAdd(v_{id}, C)$ //v_{id} is the accumulator value of $Player'_{id}s$ e-coins

Payment

$M = \sum m_i$
if $OpenCom(C, Openkey, M) == 1$ **then**
  for $k = 1$; $k <= \#numn$; $k++$ **do**
    $\sigma(coin_k) = sign_{psk}(coin_k)$ //psk is the private key of Payment
  end for
  transfer() // transfer money from real cash account
end if

Player

for each $coin_i$ accumulated in v_{id} **do**
  $wit_i = WitUpdate(wit_i, C) = f_n(wit_i, C)$.
end for
$\sigma(v_{id}) = sign_{psk_{id}}(v_{id})$ //psk_{id} is the private key of $Player_{id}$

(4) Consume

Player initiates a transaction by sending the e-coins as posted price to the Provider to purchase virtual products. The Provider verifies the received e-coins by checking the signatures of Payment. If the verification succeeds, it accepts the

transaction and sends back a Transaction No. (TNO) to the Player. Then the Player sends e-coins to Platform together with the TNO and witnesses. Platform checks whether these e-coins are in the whitelist of the Player by $AccVerify$. If it is valid, it removes the e-coins from the Player's accumulator by *AccDel* and adds them to the Provider's by *AccAdd*. It sends back the updated accumulator values of the Player and the Provider to the Player. The Player updates the remaining e-coins' witnesses and forwards the Provider's accumulator value to the Provider. The Provider updates its e-coins' witnesses and provides the Player with the chosen products. At last, the Player and Provider sign their accumulator values and send back to the Platform. Algorithm 2 shows the detail.

Algorithm 2. Consume

Require: The products to buy and the required $\{ecoin_i\}$, $ecoin_i = (Com(m_i), E_k(r_i, m_i))$

Provider(prid)

1: **for** each $ecoin_i \in \{ecoin_i\}$ **do**
2: $VerifySign_{pvk}(ecoin_i)$//$pvk$ is the public key of Payment
3: **end for**
4: $TNO \leftarrow TransactionNO.$

Platform

5: **for** each $ecoin_i \in \{ecoin_i\}$ **do**
6: $AccVerify(v_{id}, Com(m_i))$//check the $ecoin_i$ is in the Player's accumulated value v_{id}
7: **end for**
8: $v_{id} = AccDel(v_{id}, \prod Com(m_i))$//$v_{id}$ is the accumulator of $Player_{id}$
9: $v_{prid} = AccAdd(v_{prid}, \prod Com(m_i))$//$v_{prid}$ is the accumulator of $Provider_{id}$

Player(id)

10: **for** each $ecoin_i$ accumulated in v_{id} **do**
11: $wit_i = WitUpdate(wit_i, \prod Com(m_i))$
12: **end for**
13: $\sigma(v_{id}) = sign_{psk_{id}}(v_{id})$ //psk_{id} is the private key of $Player_{id}$

Provider(prid)

14: **for** each $ecoin_i$ accumulated in v_{prid} **do**
15: $wit_i = WitUpdate(wit_i, \prod Com(m_i))$
16: **end for**
17: $\sigma(v_{prid}) = sign_{prsk_{id}}(v_{prid})$ //$prsk_{id}$ is the private key of $Provider_{id}$
18: provide products

(5) Encash

Provider sends a batch of e-coins to the Platform in exchange for real currency. The Platform verifies the e-coins and checks whether they are accumulated in the Provider's whitelist. If the verification succeeds, it computes the total value of the e-coins, decrypts the keys and computes the *Openkey* to open the product of commitments $\pi_{commitments}$ according to the homomorphism. The Provider opens $\pi_{commitments}$ with *Openkey* and checks whether the sum of money equals to the one Platform announced. The Platform also removes the e-coins from the Provider's accumulator by *AccDel* and sends back the updated accumulator value. The Provider updates the remaining e-coins' witnesses, signs the accumulator value and sends back to the Platform. At last, the Payment transfers the money (after deducting the Platform's profit) from Platform's real currency account to the Provider's. See the detail in Algorithm 3.

Algorithm 3. Encash

Require: The set of e-coins to cash $\{ecoin_i\}$, $ecoin_i = (Com(m_i), E_k(r_i, m_i))$

Platform

1: **for** each $ecoin_i \in \{ecoin_i\}$ **do**
2: $AccVerify(v_{prid}, Com(m_i))$//check the $ecoin_i$ is in the Provider's accumulated value v_{prid}
3: **end for**
4: $C = \prod Com(m_i)$
5: $Openkey = \sum r_i$
6: $M = \sum m_i$
7: $v_{prid} = AccDel(v_{prid}, \prod Com(m_i))$//$v_{prid}$ is the accumulator of $Provider_{id}$

Provider(prid)

8: **if** $OpenCom(C, Openkey, M) == 1$ **then**
9: **for** each $ecoin_i$ remained in v_{prid} **do**
10: $wit_i = WitUpdate(wit_i, C)$
11: **end for**
12: $\sigma(v_{prid}) = sign_{prsk_{id}}(v_{prid})$ //$prsk_{id}$ is the private key of $Provider_{id}$
13: **end if**

Payment

14: $transfer()$

5 Property Analysis

The properties of e-coin and virtual economy are given in this section. We define the properties formally as well as the attack model. We also analyze the security and discuss the cryptographic assumptions on which it is based.

Consistency: Once an e-coin is generated, the Platform should not be able to change its real value. The adversary is the Platform in this game.

- The adversary generates an e-coin with real value m;
- The adversary changes the value m to m' without other party's notice.

Formally, the adversary wins if it can find $m' \neq m$ that $Com(m') = Com(m)$. Analysis:

$$Pr[Adv\ wins] = Pr[Com(m') = Com(m) \wedge m' \neq m] = Pr[g^{m'}h^{r'} = g^m h^r \wedge m' \neq m].$$

$r' = (m - m')(log_g h)^{-1} + r$, according to the DL Assumption, it is with negligible probability to compute r'.

Indistinguishability: The Provider cannot distinguish between different e-coins with the same nominal value. This is a necessity to ensure the fairness of the virtual market as the Provider cannot give priority to the Player whose e-coins with higher real value. In this attack game, the adversary is the Provider and the challenger is the Player. The model is:

- Challenger randomly picks $t \in \{0, 1\}$ and sends $ecoin_t$ to adversary. $ecoin_0$ and $ecoin_1$ are two coins with the same face value but different real values.
- The adversary responds $t' \in \{0, 1\}$

The adversary wins if it guess $t' = t$ with probability $1/2 + \epsilon$, ϵ is non negligible.

Analysis: $c_0 = g^{m_0}h^{r_0} \mod p$, $c_1 = g^{m_1}h^{r_1} \mod p$. c_0 and c_1 are information-theoretically indistinguishable since the randomness of r and r'.

Unreusability: The Player cannot re-spend an e-coin and the Provider should not be able to encash an e-coin more than once.

In this game, the challenger is the Platform. The adversary is the Player in re-spending and the Provider in re-encashment.

- The adversary generates two sets of e-coins: S_1, S_2, $S_1 \cap S_2 \neq \emptyset$.
- The challenger removes the e-coins $\in S_1$ from the adversary's accumulator.
- The challenger verifies the e-coins $\in S_2$.

The adversary wins if the verification succeeds.

Analysis: When S_1 is removed from S(accumulator value v), v is updated to v', v' is the accumulator of $S'(S' = S - S_1)$. If the verification in the third step succeeds, for each e-coin $\in S_2$, it needs to generate a valid witness. Let $x \in S_1 \cap S_2$, then $x \notin S'$, we need to find the witness w' of x such that $w'^x = v' = w^{\prod \{x_i | x_i \in S'\}}$, according to [10], it can be reduced to the strong RSA problem.

Transparency: All the real cash inflow should be accountable without relying on the assumption of the Platform's honesty. Let $EcoinS$ denote the set of e-coins that have been sold. Let $Cashed$ and $Uncashed$ denote the sets of all e-coins which have been encashed and not cashed respectively. $Value(S)$ denotes the function computing the aggregate real value of the e-coins set S. Then $Value(Ecoins) = Value(Cashed) + Value(Uncashed)$ holds;

In this game the challenger is the Provider and the adversary is the Platform.

- The challenger sends the e-coins set $\{(Com(m_i), E_k(r_i, m_i))\}$ to the Platform together with the witnesses.
- The adversary responds to the challenger with the $Openkey$ to reveal the aggregated commitment $\pi_{Com(m_i)}$ as a proof of the e-coins' total value M.
- The challenger opens the $\pi_{Com(m_i)}$ with $Openkey$ to check the value M.

The adversary wins if $M = Open(\pi_{Com(m_i)}, Openkey)$ while $M \neq \sum m_i$.

$$\text{Analysis:} \quad Pr[Adv\ wins] = Pr[\pi_{Com(m_i)} = g^M h^{Openkey} \wedge M \neq \sum m_i]$$

$$= Pr[g^{\sum m_i} h^{\sum r_i} = g^M h^{Openkey} \wedge M \neq \sum m_i] = Pr[h^{Openkey} = g^{\sum m_i - M} h^{\sum r_i}].$$

$Openkey = (\sum m_i - M)(log_g h)^{-1} + \sum r_i$, finding such an Openkey can be reduced to the DL problem.

Collusion-Resistance: As discussed in the previous sections, it is possible for the Platform to collude with the Player. The Platform creates an e-coin with committed value m while selling it at a price m'. $4P\_VES$ introduces Payment to defend this conspiracy. The Payment will open the commitment before signing the e-coins. Provider will verify the Payment's signature whenever it receives an e-coin. As no one can forge the signature of Payment, there is no possibility of collusion between Platform and Player without the Provider's notice.

6 Performance Evaluation

In this section, we evaluate the performance of 4P_VES and compare it with Verito. As a new party Payment is introduced, additional cost of computation

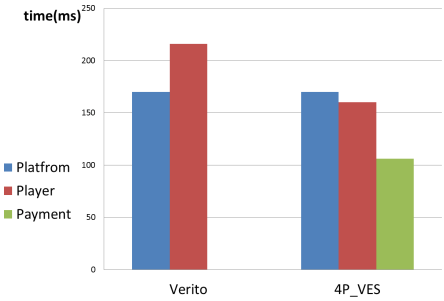

Fig. 2. Each party's cost in Purchase

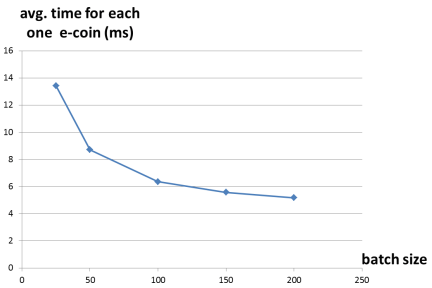

Fig. 3. Avg. cost of Purchase with diff. sizes

and communication is inevitable. We will show the extra cost is acceptable. We also try to optimize the performance by proposing a variant of 4P_VES.

During the purchase phase, as the commitment is opened by the Payment rather by the Player in Verito to detect conspiracy in our solution, the cost of this step is transferred from the Player to the Payment. And the Payment is also in charge of signing the e-coins which brings about extra cost. Figure 2 describes the cost of each party in purchase phase. As the cost of generating e-coins depends on the number of e-coins, we assume the Player buys a batch of 50 e-coins in our evaluation. The total time cost of Player and Payment is about 23 % higher than the cost of Player in Verito.

As the commitment is homomorphic and the accumulator is dynamic, the cost of opening commitment and updating accumulator can be amortized. Therefore the more e-coins in a batch, the less average cost it takes. Figure 3 shows the performance of purchase phase with different batch sizes.

Our solution adds interactions between the participants to make the system less rely on the Platform's honesty. Every time when the accumulator value changes, the Platform needs to require the corresponding signature. Table 2 compares the interaction rounds of different solutions. The variant version of 4P_VES is a tradeoff between security and communication efficiency. In the variant, the Player or Provider sends the accumulator value's signature together with the transaction request messages which will save one interaction. The expense is that the Platform merely holds the last accumulator value rather than the current one's signature which leaves a little room for the Platform's trick.

As evaluated above, although extra cost and delay are introduced in our solution, it is still within acceptable tolerance. That is to say, our solution is practical that enhances the Verito with little overhead.

7 Related Work

Our work is closely related to [8] in which the possibility of collusion between participants is not considered and the account's consistency relies on the honesty of platform. Our work is based on [8] and tries to avoid some of its shortcomings.

Other electronic cash systems focus on the anonymous and traceable payment [16]. Schemes [11,17] introduce the trusted third party (TTP) to trace double-spending. And only when some vital crime happens will the tracing be performed. Double-spending tracing schemes without TTP are also proposed in [6,9]. In [13], a provably secure E-cash scheme with tracing is proposed.

8 Conclusion and Future Work

In this paper, an accountable and transparent virtual economy system with four participant parties is proposed based on the scheme of Verito. Our system retains the properties of Verito while further study the anti-collusion problem. We introduce a new participant Payment to detect possible collusion. Our solution also enhances the security and reduces the dependency on Platform's honesty.

Although our solution solves the problem of collusion, it introduces extra cost. How to reduce computation and communication overhead and make the system more efficient and feasible is an interesting problem. In future, we will also extend our scheme to other application situations.

Acknowledgement. This work is supported in part by the National Natural Science Foundation of China under Grant 61321064, Grant 61371083, Grant 61373154, and Grant 61411146001, and in part by the Specialized Research Fund for the Doctoral Program of Higher Education of China through the Prioritized Development Projects under Grant 20130073130004.

References

1. Face book annual report (2013). http://investor.fb.com/annuals.cfm
2. Ten cent financial reports (2013). http://tencent.com/en-us/content/ir/rp/2013/attachments/201302.pdf
3. Abe, M., Cramer, R., Fehr, S.: Non-interactive distributed-verifier proofs and proving relations among commitments. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 206–223. Springer, Heidelberg (2002)
4. Au, M.H., Tsang, P.P., Susilo, W., Mu, Y.: Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 295–308. Springer, Heidelberg (2009)
5. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997)
6. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: Compact E-cash and simulatable VRFs revisited. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 114–131. Springer, Heidelberg (2009)
7. Benaloh, J.C., de Mare, M.: One-way accumulators: a decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
8. Bhaskar, R., Guha, S., Laxman, S., Naldurg, P.: Verito: A practical system for transparency and accountability in virtual economies. In: NDSS (2013)

9. Camenisch, J.L., Hohenberger, S., Lysyanskaya, A.: Compact E-cash. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
10. Camenisch, J.L., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 61. Springer, Heidelberg (2002)
11. Canard, S., Delerablée, C., Gouget, A., Hufschmitt, E., Laguillaumie, F., Sibert, H., Traoré, J., Vergnaud, D.: Fair E-cash: be compact, spend faster. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 294–309. Springer, Heidelberg (2009)
12. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. J. Cryptology **23**(4), 546–579 (2010)
13. Lian, B., Chen, G., Li, J.: Provably secure e-cash system with practical and efficient complete tracing. Int. J. Inf. Secur. **13**(3), 271–289 (2014)
14. Lipmaa, H.: Verifiable homomorphic oblivious transfer and private equality test. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 416–433. Springer, Heidelberg (2003)
15. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
16. Stadler, M.A., Piveteau, J.-M., Camenisch, J.L.: Fair blind signatures. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 209–219. Springer, Heidelberg (1995)
17. Zhang, J., Ma, L., Wang, Y.: Fair e-cash system without trustees for multiple banks. In: International Conference on Computational Intelligence and Security Workshops, CISW 2007, pp. 585–587. IEEE (2007)