IEICE TRANS. INF. & SYST., VOL.E105–D, NO.12 DECEMBER 2022

PAPER

A Novel e-Cash Payment System with Divisibility Based on Proxy Blind Signature in Web of Things

Iuon-Chang LIN[†], ***Nonmember***, **Chin-Chen CHANG**[††a)], ***Member***, ***and*** **Hsiao-Chi CHIANG**[†], ***Nonmember***

SUMMARY The prosperous Internet communication technologies have led to e-commerce in mobile computing and made Web of Things become popular. Electronic payment is the most important part of e-commerce, so many electronic payment schemes have been proposed. However, most of proposed schemes cannot give change. Based on proxy blind signatures, an e-cash payment system is proposed in this paper to solve this problem. This system can not only provide change divisibility through Web of Things, but also provide anonymity, verifiability, unforgeability and double-spending owner track.

key words: *electronic cash, proxy blind signature, bilinear pairing, mutual authentication, divisibility, anonymity, double spending track*

1. Introduction

Due to rapid progress of Internet and Web of Things (WoT), e-commerce has been used widely. Since Chaum first proposed an e-cash system based on a blind signature [1], many e-cash systems based on a blind signature have been proposed. In general, e-cash systems can be classified into two categories: online e-cash schemes [2] and offline e-cash schemes [3]. In the real world, the offline e-cash schemes are more practical because of not requiring online payment, but they easily suffer from double spending; therefore, it is more important to design a secure offline e-cash system [4].

In addition, for a blind signature scheme, the signer cannot view the message content, but a third party can make the signature verification and know if the signature is valid. However, Mambo et al. [5] first proposed the concept of a proxy blind signature scheme, which can be used by an original signer to delegate his signing ability to a proxy signature. Then a proxy signature key is generated after the proxy signer combines with the original signer's delegation. Therefore, the verifier is convinced that the signature is verified and authorized by the original signer.

However, we find that many previous proposed e-cash payment schemes [6]–[12] cannot provide divisibility. For example, a customer firstly requests an e-cash of one hundred dollars from a bank, and then he wants to pay ninety dollars for merchandise to a merchant, but the merchant cannot give change for him. In order to solve this problem, Juang and Liaw [13] proposed a divisibility e-cash scheme same as traditional cash: e-cash is issued with various face values such as \$1, \$5, \$10 to achieve the function of divisibility, but inconvenient for customers.

It is also found by recent research trends [14]–[18] that consumers' privacy is becoming more and more important. Anonymity has been one of the important properties in researches related to e-cash systems. Therefore, motivated by the e-cash system with the trustee-based scheme [19] and the proxy blind signature [20], we design a scheme based on a proxy blind signature to provide anonymity, verifiability, and untraceability. Furthermore, in order to overcome the change problem, we use Web of Things to provide divisibility for customers in this paper. In addition, since the scheme embeds pairing-based mutual authentications and key agreement into each transaction, there is no need to set up an authenticated channel for each transaction. Besides, if double spending happens, the e-cash owner's identity can be revealed by the trustee who uses symmetric decryption and exclusive-or operation.

In this paper, the related works focusing on designing an e-cash payment system is presented in Sect. 2, followed by Sect. 3 which describes the architecture of an e-cash payment system. In Sect. 4, an an e-cash payment protocol based on a proxy blind signature is introduced. The security analysis is presented in Sect. 5. Finally, conclusions are presented in Sect. 6.

2. Related Work

In this section, some preliminaries used in this paper are reviewed, including bilinear pairing, application of ID-based cryptosystem using pairing, and Tan's proxy blind signature.

2.1 Bilinear Pairing

The basic definition and properties of bilinear paring are described in the following.

Let G_1 be a cyclic additive group generator by P over an elliptic curve with order q, and G_2 be a cyclic multiplicative group of the same order. A bilinear map e: G_1 × G_1 → G_2 is an admissible bilinear paring, which satisfies the following properties:

(1) Identity: For all $P \in G_1$, $e(P, P) = 1$.

Manuscript received May 20, 2022.
Manuscript revised July 2, 2022.
Manuscript publicized September 2, 2022.

[†]The authors are with the Department of Management Information Systems, National Chung Hsing University, Taichung, Taiwan, R.O.C.
[††]The author is with the Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan, R.O.C.
a) E-mail: alan3c@gmail.com (Corresponding author)

DOI: 10.1587/transinf.2022EDP7081

(2) Alternation: For all $P_1, P_2 \in G_1, e(P_1, P_2) = e(P_2, P_1)$.
(3) Bilinearity: For all $P_1, P_2 \in G_1$ and any $a, b \in Z^*_q$, $e(aP_1, bP_2) = e(P_1, P_2)^{ab}$.
(4) Non-degeneracy: For all $P_1, P_2 \in G_1$, $P_1 \neq P_2$, $e(P_1, P_2) \neq 1$.
(5) Computability: There exists an efficient algorithm to compute $e(P_1, P_2)$ for any $P_1, P_2 \in G_1$.

2.2 Applications of Bilinear Pairing ID-Based Cryptosystem

First, the ID-based cryptosystem using a user's identity as a public key was proposed by Shamir [21] in 1984. The practical ID-based cryptosystem first proposed by Boneh and Franklin applies bilinear pairing, so many ID-based pairing applications have been proposed.

The original purpose for ID-based encryption is to distribute a public key to a user. In such a scheme, a Key Generation Center (*KGC*) is responsible for setting system parameters and key distribution. A secret key $s \in Z^*_q$ is randomly chosen by *KGC*, and $P_{Pub} = sP$ is set. The $\{G_1, G_2, P, q, e\}$ is chosen by KGC as the system parameters, as described in Sect. 2.1, and two hash functions H_1: $\{0,1\}^* \rightarrow \{0,1\}^q$ and H_2: $\{0,1\}^* \rightarrow G_1$ are defined. When a user sends his identity *ID* to *KGC*, he needs to request a public/private key pair through a secure channel to distribute keys, so the user's public key is computed by KGC as $Y_{ID} = H_1(ID)$, and private key computed as $X_{ID} = sY_{ID}$.

In such a scheme, a default session key based on the properties of bilinear paring can be shared, so the message can be encrypted and decrypted. For example, assuming user A and user B have a public/private key pair $\{Y_A = H_1(ID_A), X_A = sY_A\}$, $\{Y_B = H_1(ID_B), X_B = sY_B\}$, respectively. Then users A and B can compute the default session key $K_{AB} = e(X_A, Y_B) = e(sY_A, Y_B) = e(Y_A, Y_B)^s$ and $K_{BA} = e(X_B, Y_A) = e(sY_B, Y_A) = e(Y_B, Y_A)^s$, respectively, clearly, $K_{AB} = K_{BA}$.

Next, we introduce an application of ID-based cryptosystem using pairing.

- Mutual authentication and session key agreement

Mutual authentication means that both the users are authenticated to each other within the same protocol. We briefly describe the procedure as follows:

(1) Firstly, *user A* chooses a random number $a \in Z^*_q$ and uses his private key X_A and Y_B, which is the public key of his communication target B, to compute the session key as $K_{AB} = e(X_A, Y_B)^a = e(sY_A, Y_B)^a = e(Y_A, Y_B)^{sa}$. Then A sends $\{ID_A, aY_A, E_{K_{AB}}(a)\}$ to B, where $E_{K_{AB}}(a)$ is asymmetric encryption on a with session key K_{AB}.
(2) After receiving the message from A, B uses his secret key X_B and received aY_A to compute the session key as $K_{BA} = e(aY_A, X_B) = e(aY_A, sY_B) = e(Y_A, Y_B)^{sa}$.

If B can successfully decrypt the $E_{K_{AB}}(a)$ with the session key computed, it is obvious that $K_{AB} = K_{BA}$. And if B can successfully authenticate the identity of A by checking whether aY_A equals $a \cdot H_1(ID_A)$, it means A is the man who B wants to communicate. Therefore, B can directly authenticate A; on the contrary, A has indirectly authenticated B, because only the right one can decrypt the message with the intended private key X_B to compute the right key K_{BA}. Similarly, if B sends his identity and an encrypted message to A, then A can successfully decrypt it and verify the identity of B, and A has directly authenticated B. Hence, this scheme can achieve mutual authentication and session key agreement.

2.3 Tan's Proxy Blind Signature

In 2010, the proxy blind signature proposed by Tan that is based on bilinear pairing to provide unforgeability of e-cash, anonymity for honest customers and efficient traceability of double spending [22]. In this section, Tan's proxy blind signature scheme is roughly reviewed. The blind signature algorithm proposed by Fan *et al.* [23] has motivated four participants in Tan's proxy blind signature: an original signer O, a proxy signer P, a signature requester U, and a verifier V. It includes seven algorithms: *Setup*, *KeyGen*, *DelegationGen*, *DelegationVerify*, *SignatureKeyGen*, *ProxyBlindSign*, and *ProxyBlindVerify*. The concept of Tan's is as follows: a delegation token on a warrant is generated by the original signer O, and his private key and the delegation token are used by the proxy signer to compute the proxy signature key. Finally, the proxy signature key is used by the proxy signer to produce the proxy blind signature on message m.

(1) *Setup*: Let (G_1, G_2) be two bilinear groups where $|G_1| = |G_2| = q$ for some prime q order. Let Q be a generator of G_1, and e denote an admissible pairing e: $G_1 \times G_1 \rightarrow G_2$, and select two collision-resistant hash function as H_1: $\{0,1\}^* \rightarrow Z^*_q$ and H_2: $\{0,1\}^* \rightarrow G_1$.
(2) *KeyGen*: The original signer O and the proxy signer P have their private/public key pairs (X_O, Y_O) and (X_P, Y_P), respectively, where $Y_O = X_O Q$ and $Y_P = X_P Q$, $X_O, X_P \in Z^*_q$.
(3) *DelegationGen*: The original signer O generates a warrant w, which records the delegation limitations, and valids period of delegation, the identity information of original signer and proxy signer, *etc*. The original signer O chooses a random number $r_O \in Z^*_q$ and computes $R_O = r_O Q$, $s_O = X_O H_1(w \,\|\, R_O) + r_O \, mod \, q$. Then O sends (w, s_O, R_O) to P through a secure channel.
(4) *DelegationVerify*: After receiving the delegation, P firstly verifies the validity of the warrant w by checking the following equality: $s_O Q = H_1(w \,\|\, R_O) Y_O + R_O$.
(5) *SignKeyGen*: If the delegation is not valid, P will reject it. Otherwise, P will accepts it, and the proxy signature key $s_P = s_O + X_P \, mod \, q$ is computed to get the corresponding proxy public key as $Y_{Pub} = s_P Q = H_1(w \,\|\, R_O) Y_O + R_O + Y_P$.
(6) *ProxyBlindSign*: Then P generates the proxy blind signatures which involve the following steps.

(i) *Initialization*: If a signature requester U wants to

obtain a proxy blind signature on message m, U sends a request to the proxy signer P. Then P responses (w, R_O) to U.

(ii) *Blinding*: U chooses two random integers $b_1, b_2 \in Z^*_q$, and computes $R = b_1 H_2(m \| R_O \| w) + b_2 Q$, and then sends R to P.

(iii) *BlindSign*: P computes the blind signature $R_P = s_P R$ and sends it to U.

(iv) *Unblinding*: After receiving the message, U computes $S = b_1^{-1}(R_P - b_2 Y_{Pub})$. Then the (w, R_O, S) is a proxy blind signature on message m.

ProxyBlindVerify: When a verifier V receives a proxy blind signature (m, w, R_O, S), he verifies whether it is valid by checking if the following equation holds or not: $e(S, Q) = (H_2(m \| R_O \| w), H_1(w \| R_O) Y_O + R_O + Y_P$.

3. Architecture of Proposed Scheme

The protocol of this paper is demonstrated in Fig. 1 and described in the following. There are eight processes, including Customer C, Trustee T, Bank B, Cloud L, Merchant M issuing license, withdrawal money with license, delegation, e-cash withdrawal, payment, deposit, and e-cash owner tracing. The payment scenario is that customer C takes cash and goes to bank B to exchange electronic cash with a license issued by Trustee T. After obtaining the e-cash, customer C can go to merchant M to spend. Finally, merchant M will take the e-cash for settlement with bank B. This scenario can be used in most physical consumption or e-commerce fields.

Before withdrawing money from bank B, customer C needs request trustee T to issue a license for him. Then customer C uses the license to withdraw the money from bank B. After receiving the license, bank B verifies it. If the license is valid, bank B will delegate his signing ability and the total money which customer C requested to Cloud L, and L combines the delegation of B and his private key to generate a proxy signature key. Therefore, the L can divide the total money into many small e-cashs and generate a proxy blind signature for every small e-cash. When customer C wants to spend e-cash for purchases, he just needs to withdraw it from Cloud L at any time. For example, customer C firstly withdraws \$1,000 from bank B, then bank B delegates the total money to Cloud L. When C wants to pay \$580 for purchases, he just needs to withdraw \$580 from L, and then Cloud L deducts it. Next time customer C wants to use the surplus \$420, he also withdraws it from Cloud L. Customer C would not worry that the B has closed, or the e-cash cannot give change. After that, customer C can pay the e-cash to Merchant M for purchases. After receiving the e-cash, Merchant M verifies the validation of e-cash. If it is valid, Merchant M deposits it to bank B, then he will get the real money in his account.

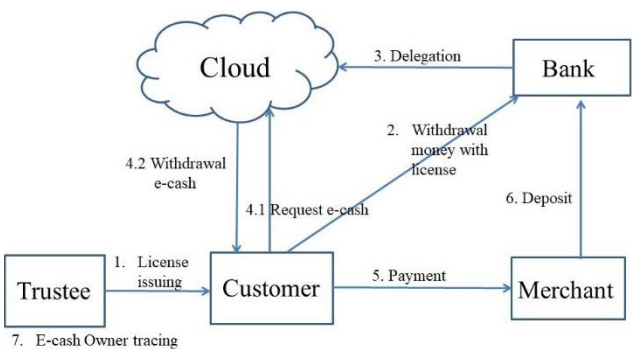

Fig. 1 The architecture of e-cash payment system.

4. Proposed Scheme

4.1 Notations

Before proceeding to our scheme, some notations used in Table 1 throughout the paper is first introduced to describe the e-cash payment system.

Table 1 Notations for protocol description.

Notations	Description
C	Customer
T	Trustee
B	Bank
L	Cloud
M	Merchant
KGC	Key Generation Center
LST	License secret token
LVT	License verifiable token
TNO	The transaction number
CNO	The serial number of e-coin
$ID_C/ID_T/ID_B/ID_L$	The identity of customer(C)/trustee(T)/bank(B)/cloud(L)
$(X_C, Y_C)/(X_T, Y_T)/(X_B, Y_B)/(X_L, Y_L)$	The private/public key pair of customer(C)/trustee(T)/bank(B)/cloud(L)
K_T	The secret key of trustee
K_{AB}	The session key between A and B
val	The total amount of customer request from Bank
a	The face value of e-coin
s_p	The proxy signature key
w	A warrant generated by bank
T_s	Timestamp
$\|\|$	The concatenation of two strings
$\oplus$	Exclusive-or

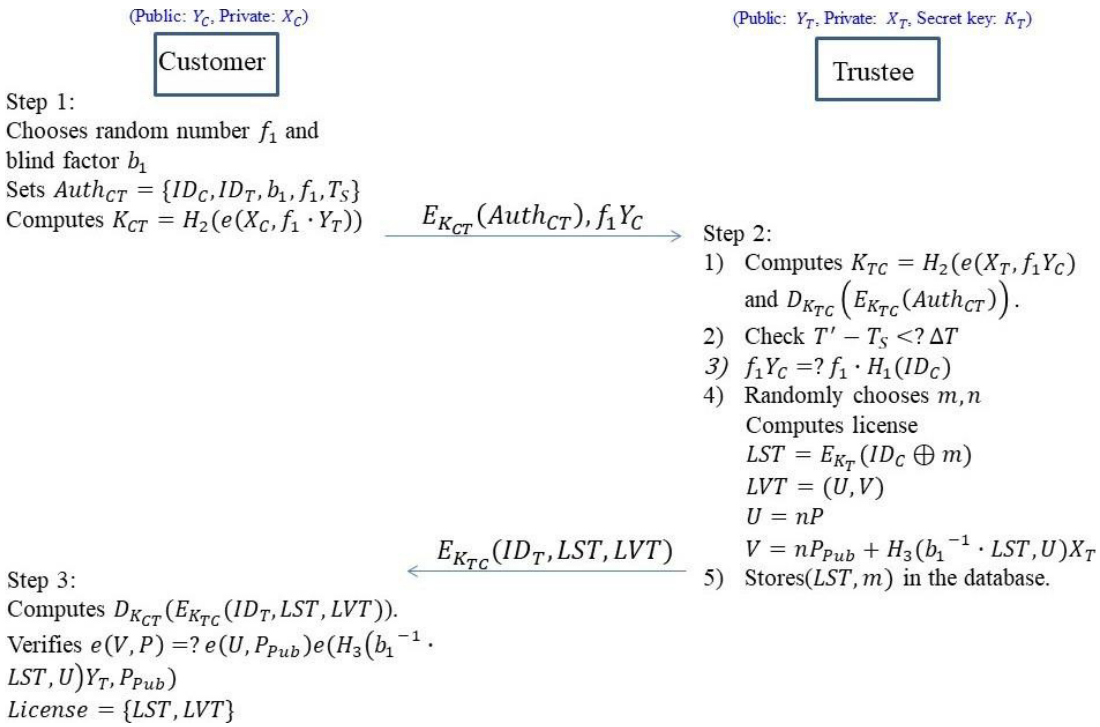

Fig. 2 License-issuing protocol.

4.2 System Set-Up

G_1 is selected by the key generation center (*KGC*) as a cyclic additive group with generator P and order q, G_2 as acyclic multiplicative group with the same order, and a bilinear paring map e: $G_1 \times G_1 \rightarrow G_2$. The *KGC* defines three hash functions, H_1: $\{0,1\}^* \rightarrow G_1$, H_2: $G_2 \rightarrow \{0,1\}^q$, H_3: $\{0,1\}^* \rightarrow \{0,1\}^q$, a symmetric encryption algorithm E, and the corresponding decryption algorithm D. A random number $s \in Z^*_q$ is chosen by the *KGC* as its private key, and he system public key $P_{Pub} = sP$ is set. Then the system parameters $\{G_1, G_2, q, P, P_{Pub}, e, H_1, H_2, H_3, E, D\}$ is published by the *KGC*.

As for key generation, a customer sends his identity ID_C to the *KGC*, and then the *KGC* computes the public key of the customer as $Y_C = H_1(ID_C)$ and his private key as $X_C = s \cdot Y_C$ and send the key pair to customer C through a secure channel. Likewise, the *KGC* generate the public/private key pairs $\{X_T, Y_T\}$, $\{X_B, Y_B\}$, $\{X_L, Y_L\}$, and $\{X_M, Y_M\}$ for Trustee T, Bank B, Cloud L, and Merchant M, respectively.

Additionally, a secret key K_T is selected by T to protect or reveal the identity of customer in the *LST*.

4.3 License Issuing

Before withdrawing money from a bank, a customer needs to request trustee T to issue a license for him. The steps of this phase are specified as follows (refer to Fig. 2):

Step 1: C chooses a random number $f_1 \in Z^*_q$ and a blind factor $b_1 \in Z^*_q$. And he computes the session key $K_{CT} = H_2(e(X_C, f_1 \cdot Y_T))$ with his private key X_C and T's public key Y_T and then sets $Auth_{CT}$ as $\{ID_C, ID_T, b_1, f_1, T_s\}$ where T_s is a timestamp. He then sends $\{E_{K_{CT}}(Auth_{CT}), f_1 Y_C\}$ to T, where $E_{K_{CT}}(Auth_{CT})$ is asymmetric encryption on $Auth_{CT}$ with session key K_{CT}.

Step 2: After receiving the message from C, T acts the following steps.

(1) T computes the session key $K_{TC} = H_2(e(X_T, f_1 \cdot Y_C))$ with his private key X_T and received $f_1 Y_C$ to decrypt $E_{K_{CT}}(Auth_{CT})$. If K_{TC} is equal to K_{CT} he can obtain the $Auth_{CT}$. After that, he checks whether ID_C and ID_T in $Auth_{CT}$ are correct, if they are not, T rejects the request.

(2) $T' - T_s$ is checked by T whether it is less than ΔT: T′ is the current system time of T, and ΔT is the amount tolerated for a transmission delay. If not, T rejects the reject.

(3) f_1 and ID_C in $Auth_{CT}$ are used by T to compute $f_1 \cdot H_1(ID_C)$, and T checks whether it is equal to the $f_1 Y_C$ received. If it is, T believes C is a valid party; on the other hand, T rejects the request.

(4) If above checks are correct, T randomly chooses a number m to compute the license secret token (*LST*) as $E_{K_T}(ID_C \oplus m)$.

(5) In order to sign on $b_1^{-1} \cdot LST$, T computes $U = nP$ and $V = nP_{Pub} + H_3(b_1^{-1} \cdot LST, U)X_T$, and sets (U, V) as the license verifiable token *(LVT)*.

(6) T sends $E_{K_{TC}}(ID_T, LST, LVT)$ to C, where $E_{K_{TC}}(ID_T, LST, LVT)$ is asymmetric encryption on (ID_T, LST, LVT) with the session key K_{TC}, and stores (LST, m) in the database.

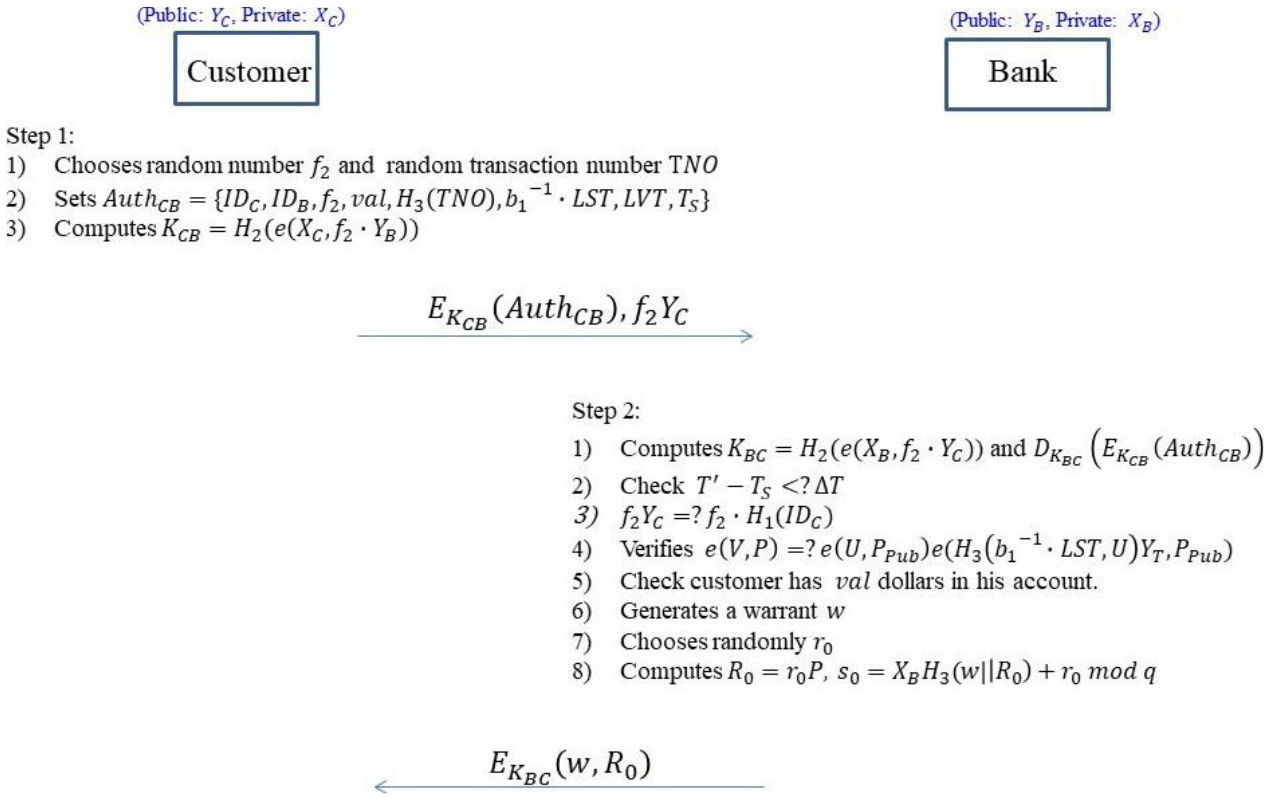

Fig. 3 Money withdrawal with license protocol.

Step 3: After receiving the message from T, C uses the session key K_{CT} to decrypt $E_{K_{TC}}(ID_T, LST, LVT)$. If ID_T is correct, C believes T is a valid party. And then C uses the public key of T to verify the license by checking whether $e(V,P)$ equals $e(U, P_{Pub}) e(H_3({b_1}^{-1} \cdot LST, U) Y_T, P_{Pub})$. If it is, the license for C is (LST, LVT).

4.4 Money Withdrawal with License

In this phase, customer C asks a withdraw money request to bank B. Bank B can confirm the license by checking the LST and LVT. If it is correct, bank B generates a warrant to customer C. The steps of this phase are specified as follows (refer to Fig. 3):

Step 1: customer C chooses a random number $f_2 \in {Z^*}_q$, a random serial number TNO, and determines an amount of money val he requests. And he computes the session key $K_{CB} = H_2(e(X_C, f_2 \cdot Y_B))$ with his private key X_C and B's public key Y_B. In addition, then he sets $Auth_{CB}$ as $\{ID_C, ID_B, f_2, val, H_3(TNO), {b_1}^{-1} \cdot LST, LVT, T_s\}$; b_1 is selected by C in the license-issuing protocol; LST blinded by ${b_1}^{-1}$; $\{{b_1}^{-1} \cdot LST, LVT\}$, the blind license; T_s, a timestamp. C then sends $\{E_{K_{CB}}(Auth_{CB}), f_2 Y_C\}$ to B, where $E_{K_{CB}}(Auth_{CB})$ is asymmetric encryption on $Auth_{CB}$ with session key K_{CB}.

Step 2: After receiving the message from customer C, bank B acts the following steps.

(1) bank B computes the session key $K_{BC} = H_2(e(X_B, f_2 \cdot Y_C))$ with his private key X_B and receives $f_2 Y_C$ to decrypt $E_{K_{CB}}(Auth_{CB})$. If K_{BC} is equal to K_{CB}, he can obtain the $Auth_{CB}$. After that, he checks whether ID_C and ID_B in $Auth_{CB}$ are correct, if they are not, bank B rejects the request.

(2) bank B checks whether $T' - T_s$ is less than ΔT. If not, he rejects the request.

(3) f_2 and ID_C in $Auth_{CB}$ are used by bank B to compute $f_2 \cdot H_1(ID_C)$, and to check whether it is equal to the $f_2 Y_C$ received. If it is, bank B believes customer C as a valid party; otherwise, bank B rejects the request.

(4) B verifies the license of customer C by checking whether $e(V,P)$ equals $e(U, P_{Pub}) e(H_3({b_1}^{-1} \cdot LST, U) Y_T, P_{Pub})$. If equal, bank B believes that the blind license of customer C was issued by T; otherwise, he rejects the request.

(5) bank B then checks customer C has val dollars in his account. If not, rejects the request.

(6) *If the above checks are correct, a warrant w is generated by B, recording the delegation restrictions of authority: valid periods of the delegation, the identity of original signer bank B and the proxy signer cloud L.* Then a random number $r_0 \in {Z^*}_q$ is selected to compute $R_0 = r_0 P$, $s_0 = X_B H_3(w \,\|\, R_0) + r_0 \ mod \ q$, where $\|$ is the concatenation of w and R_0.

(7) Finally, bank B sends $(w \,\|\, R_0)$ to customer C which is encrypted by session key K_{BC}. When customer C wants to request the e-cash for purchases, he must send $(w \,\|\, R_0)$ to L to obtain the e-coin.

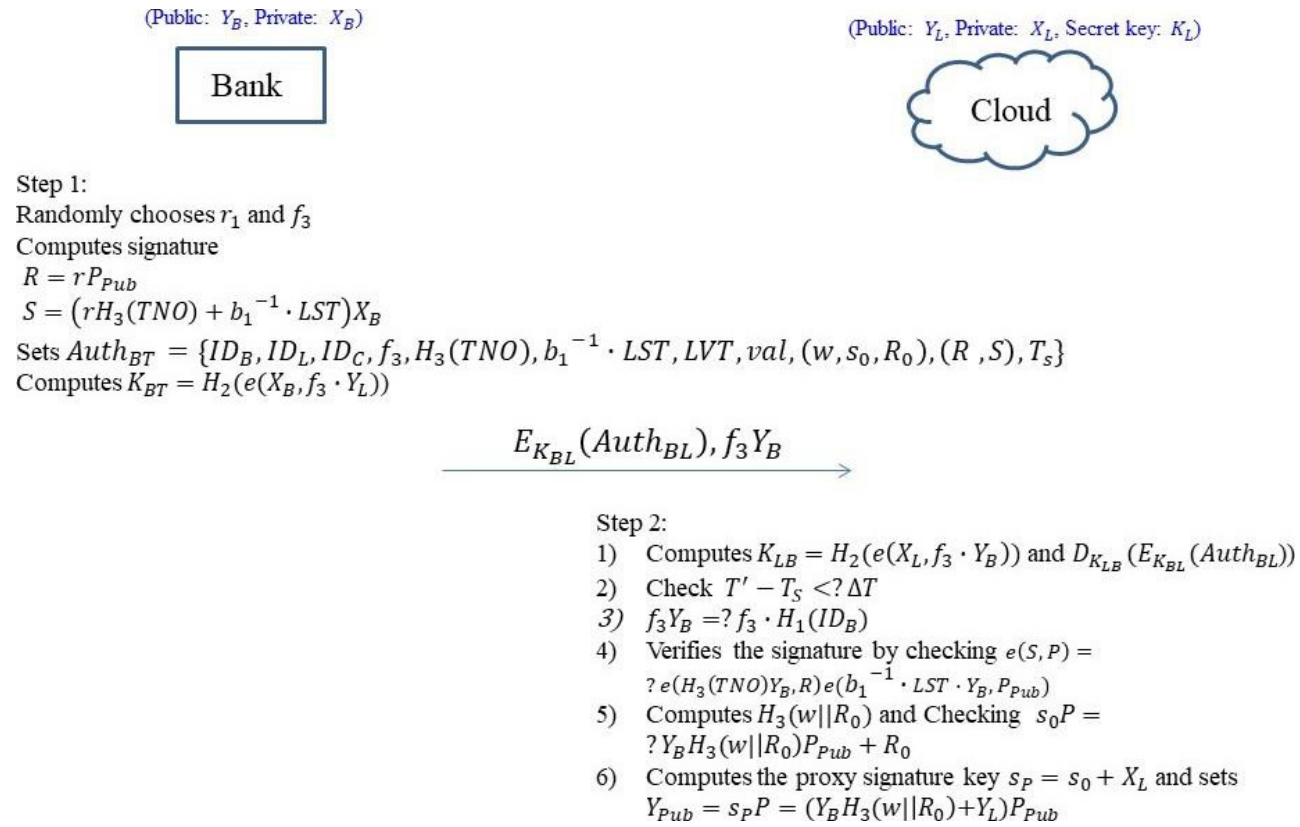

Fig. 4 Delegation protocol.

4.5 Delegation Protocol

In this phase, bank B delegates his signing ability to cloud L, and cloud L combines the delegation of bank B and his private key to generate a proxy signature key. By using the proxy signature, cloud L can produce the proxy signature on e-cash. The steps of this phase are specified as follows (refer to Fig. 4):

Step 1: bank B randomly chooses $r_1 \in Z^*_q$ and $f_3 \in Z^*_q$, and then he computes signature as $R = r_1 P_{Pub}$, $S = (r_1 H_3(TNO) + {b_1}^{-1} \cdot LST)X_B$. And he computes the session key $K_{BL} = H_2(e(X_B, f_3 \cdot Y_L))$ with his private key X_B and L's public key Y_L. In addition, then he sets $Auth_{BL}$ as $\{ID_B, ID_L, ID_C, f_3, H_3(TNO), {b_1}^{-1} \cdot LST, LVT, val, (w, s_0, R_0), (R, S), T_s\}$, where b_1 is the blind factor chosen by C in the license-issuing protocol, LST is blinded by ${b_1}^{-1}$, $\{{b_1}^{-1} \cdot LST, LVT\}$ is the blind license, val is the amount of money which bank B is delegated to cloud L, (w, s_0, R_0) is generated by bank B in the money withdrawal with license protocol, and T_s is a timestamp. bank B then sends $\{E_{K_{BL}}(Auth_{BL}), f_3 Y_B\}$ to cloud L, where $E_{K_{BL}}(Auth_{BL})$ is the result of $Auth_{BL}$ encrypted by session key K_{BL}.

Step 2: After receiving the message from bank B, cloud L acts the following steps.

(1) cloud L computes the session key $K_{LB} = H_2(e(X_L, f_3 \cdot Y_B))$ with his private key X_L and receives $f_3 Y_B$ to decrypt $E_{K_{BL}}(Auth_{BL})$. If K_{LB} is equal to K_{BL}, he can obtain the $Auth_{BL}$. After that, he checks whether ID_B and ID_L in $Auth_{BL}$ are correct, if they are not, L stops the phase.

(2) cloud L checks whether $T' - T_s$ is less than ΔT. If not, L stops the phase.

(3) cloud L checks bank B's identity by useing f_3 and ID_B in $Auth_{BL}$ to compute $f_3 \cdot H_1(ID_B)$ and checks whether it is equal to the $f_3 Y_L$ which he received. If not, cloud L stops the phase.

(4) cloud L verifies the signature of B by checking whether $e(S, P)$ equals $e(H_3(TNO)Y_B, R)e({b_1}^{-1} \cdot LST \cdot Y_B, P_{Pub})$. If equal, cloud L believes that the signature was signed by bank B; otherwise, he rejects the request.

(5) In order to verify the delegation, L computes $H_3(w||R_0)$ and checking whether $s_0 P$ equals $Y_B H_3(w \,||\, R_0)P_{Pub} + R_0$. If equal, cloud L believes the delegation is valid; otherwise, L stops the phase.

(6) Finally, cloud L computes the proxy signature key $s_P = s_0 + X_L$. And he sets $Y_{Pub} = s_P P = (Y_B H_3(w \,||\, R_0) + Y_L)P_{Pub}$.

4.6 e-Cash Withdrawal

In this phase, customer C withdraws the e-cash from cloud L so that he can pay the e-cash to Merchant M for purchasing the goods. The steps of this phase are specified as follows (refer to Fig. 5):

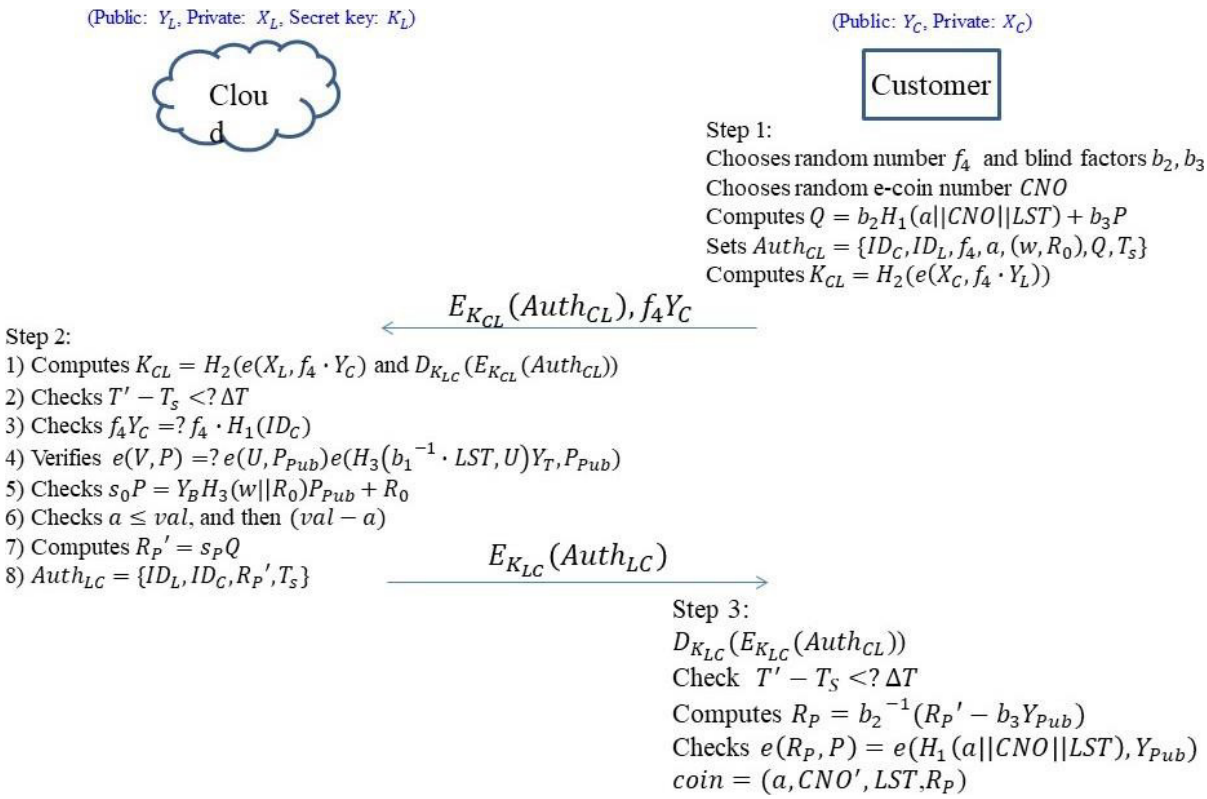

Fig. 5 e-Cash withdrawal protocol.

Step 1: C chooses a random number f_4, two blind factors b_2 and b_3, and a random e-cash number CNO, and then he determines the face value of e-cash a that he wants to withdraw this time. He then computes Q = $b_2H_1(a \parallel CNO \parallel LST) + b_3P$. And he computes the session key $K_{CL} = H_2(e(X_C, f_4 \cdot Y_L))$ with his private key X_C and cloud L's public key Y_L. In addition, then he sets $Auth_{CL}$ as $\{ID_C, ID_L, f_4, a, (w, R_0), Q, T_s\}$ where (w, R_0) is generated by bank B in the money withdrawal with license protocol, and T_s is a timestamp. Customer C then sends $\{E_{K_{BL}}(Auth_{BL}), f_4Y_C\}$ to cloud L, where $E_{K_{CL}}(Auth_{CL})$ is the result of $Auth_{CL}$ encrypted by session key K_{CL}.

Step 2: After receiving the message from bank B, cloud L acts the following steps.

(1) cloud L computes the session key $K_{LC} = H_2(e(X_L, f_4 \cdot Y_C))$ with his private key X_L and receives f_4Y_C to decrypt $E_{K_{CL}}(Auth_{CL})$. If K_{LC} is equal to K_{CL}, he can obtain the $Auth_{CL}$. After that, he checks whether ID_C and ID_L in $Auth_{CL}$ are correct; if they are not, cloud L stops the phase.
(2) L checks whether $T' - T_s$ is less than ΔT. If not, L stops the phase.
(3) cloud L uses f_4 and ID_C in $Auth_{CL}$ to compute $f_4 \cdot H_1(ID_C)$ and checks whether it is equal to the f_4Y_C which he received. If equal, cloud L believes customer C is a valid party; otherwise, cloud L stops the phase.
(4) cloud L verifies the license of customer C, which is stored in cloud L's database in the delegation protocol, by checking whether $e(V, P)$ equals $e(U, P_{Pub})$ $e(H_1(b_1{}^{-1} \cdot LST, U)Y_T, P_{Pub})$. If equal, cloud L believes that the blind license of customer C is issued by trustee T; otherwise, he rejects the request.
(5) In order to verify the delegation, cloud L computes $H_3(w \parallel R_0)$ and checks whether s_0P equals $Y_BH_3(w \parallel R_0)P_{Pub} + R_0$. If equal, cloud L believes the delegation is valid; otherwise, cloud L stops the phase.
(6) cloud L checks whether the face value of e-cash a is over the amount of val which C deposit in the bank, if it holds, cloud L deducts a value form val; otherwise, the phase has to be rejected.
(7) If above checks are correct, cloud L computes the proxy blind signature as $R'_P = s_PQ$.
(8) cloud L sends $Auth_{LC}$ as (ID_L, ID_C, R'_P, T_S) to customer C which is encrypted by session key K_{LC}.

Step 3: After receiving the message from cloud L, customer C can decrypt the message with the session key K_{CL}, if C is the corresponding receiver. After that, he checks whether $T' - T_s$ is less than ΔT. If not, cloud L stops the phase. He then unblinds the proxy blind signature by computing $R_P = b_2{}^{-1}(R'_P - b_3Y_{Pub})$ and verifies the proxy blind signature by checking whether $e(R_P, P)$ equals $e(H_1(a \parallel CNO \parallel LST), Y_{Pub})$. If the verification is correct, customer C can believe cloud L is a valid party which is delegated by bank B. Finally, the e-cash is (a, CNO, LST, R_P).

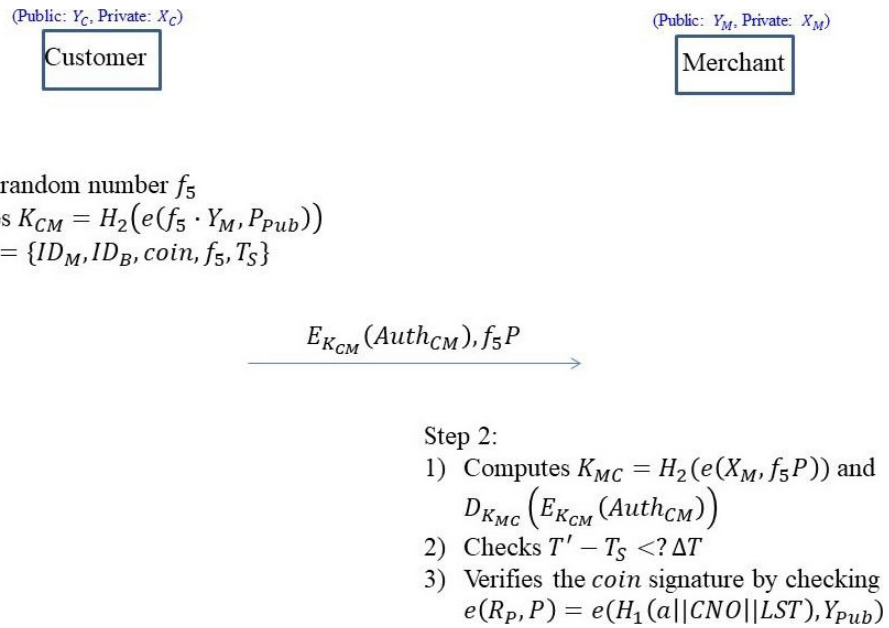

Fig. 6 Payment protocol.

4.7 Payment

In this protocol, when customer C decides to purchase a product from a specific merchant M, he can pay the e-cash to merchant M without disclosing his identity. The steps of this phase are specified as follows (refer to Fig. 6):

Step 1: customer C chooses a random number f_5, then he uses merchant M's public key Y_M to compute the session key K_{CM} as $H_2(e(f_5 \cdot Y_M, P_{Pub})$, and sets $Auth_{CM}$ as $\{ID_M, ID_B, coin, f_5, T_S\}$. He then sends $E_{K_{CM}}(Auth_{CM})$ and f_5P to merchant M.

Step 2: After receiving the message from customer C, merchant M acts the following steps.

(1) merchant M computes the session key $K_{MC} = H_2(e(X_M, f_5P))$ with his private key X_M and receives f_5P to decrypt $E_{K_{CM}}(Auth_{CM})$. If K_{MC} is equal to K_{CM}, he can obtain the $Auth_{CM}$. After that, he checks whether ID_M in $Auth_{CM}$ is correct; if not correct, merchant M stops the phase.
(2) merchant M checks whether $\mathrm{T}' - T_s$ is within the valid time interval ΔT. If it does not hold, merchant M stops the phase; otherwise, this procedure continues.
(3) Finally, merchant M verifies the e-cash signature by checking whether $e(R_P, P)$ equals $e(H_1(a \parallel CNO \parallel LST), Y_{Pub})$. If the verification is correct, merchant M can believe the e-cash received from customer C is valid, then transfers and deposits to bank B.

4.8 Deposit

In this protocol, merchant M deposits the received e-cash to bank B. When receiving the deposit requirement from the merchant M, bank B verifies the signature of the e-cash. If verified, he accepts the e-cash and credits the account of the merchant M; otherwise, bank B requests trustee T to disclose the identity of the dishonest customer. The steps of this phase are specified as follows (refer to Fig. 7):

Step 1: Firstly, merchant M chooses a random number f_6 and sets $Auth_{MB}$ as $\{ID_M, ID_B, coin, f_6, T_S\}$ and computes session key K_{MB} as $H_2(e(X_M, f_6 \cdot Y_B))$ by using his private key X_M and B's public key Y_B. Then he sends $\{E_{K_{MB}}(Auth_{MB}), f_6Y_M\}$ to bank B.

Step 2: After receiving the message from merchant M, bank B acts following steps.

(1) bank B computes the session key $K_{BM} = H_2(e(X_B, f_6 \cdot Y_M))$ with his private key X_B and receives f_6Y_M to decrypt $E_{K_{MB}}(Auth_{MB})$. If K_{BM} is equal to K_{MB} he can obtain the $Auth_{MB}$. After that, he checks whether ID_M and ID_B in $Auth_{MB}$ are correct, if they are not, B stops the phase.
(2) merchant M checks whether $\mathrm{T}' - T_s$ is within the valid time interval ΔT. If it does not hold, merchant M stops the phase; otherwise, this procedure continues.
(3) bank B uses f_6 and ID_M in $Auth_{MB}$ to compute $f_6 \cdot H_1(ID_M)$ and checks whether it is equal to the f_6Y_M which he received. If it is, bank B believes merchant M is a valid party; otherwise, B stops the phase.
(4) If the above checks are correct, bank B verifies the signature of e-cash by checking whether $e(R_P, P)$ equals $e(H_1(a||CNO||LST), Y_{Pub})$. If the verification is correct, bank B can believe the e-cash received from merchant M is valid.
(5) After that, B checks whether the e-cash has double spent before; if it is fresh, he credits amount a into merchant M's account.

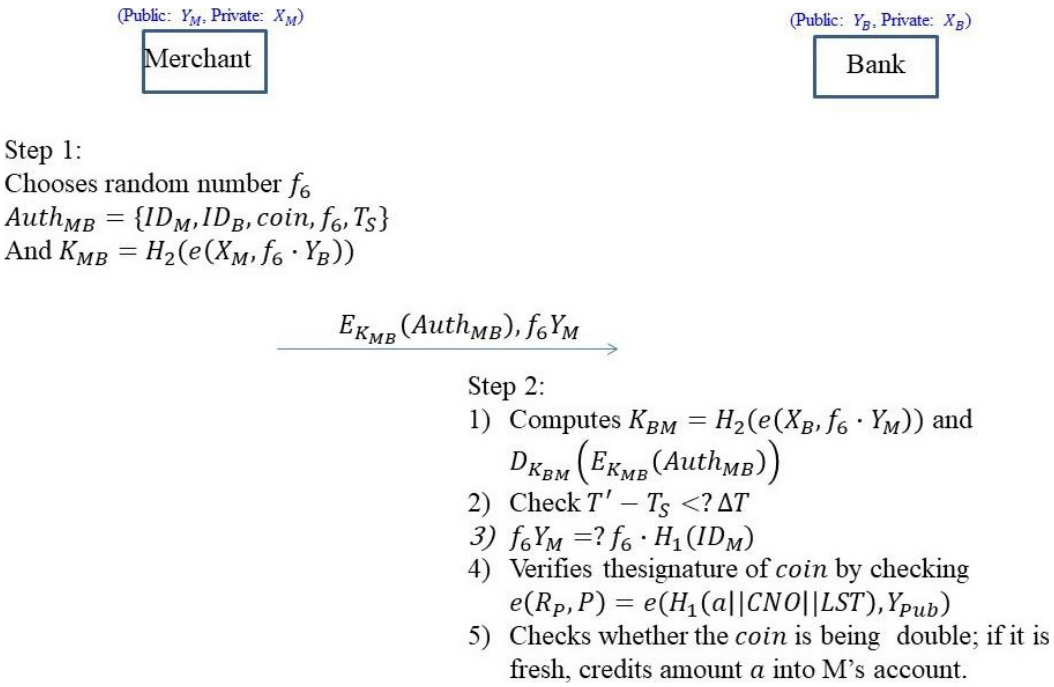

Fig. 7 Deposit protocol.

4.9 e-Cash Owner Tracing

In the proposed scheme, if the e-cash is spent twice (double spending) or abused by a criminal, the bank can ask trustee T to revoke the anonymity of the e-cash by providing the LST.

$$LST = E_{K_T}(ID_C \oplus m)$$

After receiving this request, trustee T retrieves (LST, m) from its database and uses his secret key K_T, to execute the following expression to reveal the e-cash owner's identity

$$ID = D_{KT}(LST) \oplus m.$$

Thus, it can disclose the owner's identity of the e-cash.

5. Security Analysis

In this section, it will demonstrate the security of the proposed e-cash payment protocol. Firstly, we easily confirm the following properties; (1) Identifiablity: the identity of the original signer bank B and the identity of the proxy signer cloud L can be easily confirmed by a verifier through the warrant w. (2) Non-repudiation: A proxy blind signature contains the warrant w. During the signature verification, the public key of original signer bank B and the public key of proxy signer cloud L must be used by the verifier. Hence, for a valid proxy blind signature, the delegation delegated to cloud L cannot be denied by the original signer bank B, and the proxy signer cloud L also cannot deny the signature signed. Other security analysis will be confirmed in the following; it involves five aspects: mutual authentication, verifiability, untraceability, unforgeability, and e-cash tracing.

5.1 Mutual Authentication

As defined in Sect. 2.2, for every transaction made between two parties, the message has to be encrypted by the session key. After the encryption is completed, one party will transfer it to another. Only the intended one can compute the corresponding session key to decrypt the request message. Hence, this scheme can achieve mutual authentication and session key agreement.

5.2 Verifiability

(1) In the money withdrawal with a license protocol and e-cash withdrawal protocol, bank B and cloud L can verify the license using trustee T's public key Y_T. The license is valid only if the following equation holds:

$$\begin{aligned} e(V, P) &= e(nP_{Pub} + H_3({b_1}^{-1} \cdot LST, U)X_T, P) \\ &= e(nP, P_{Pub})e(H_3({b_1}^{-1} \cdot LST, U)Y_T, P_{Pub}) \\ &= e(U, P_{Pub})e(H_3({b_1}^{-1} \cdot LST, U)Y_T, P_{Pub}) \end{aligned}$$

(2) In the delegation protocol, the cloud can verify the bank's signature using bank B's pubic key Y_B and the blind LST. The bank's signature is valid if the following equation holds:

$$\begin{aligned} e(S, P) &= e((rH_3(TNO) + {b_1}^{-1} \cdot LST)X_B, P) \\ &= e(H_3(TNO)Y_B, rP_{Pub})e({b_1}^{-1} \cdot LST \cdot Y_B, P_{Pub}) \end{aligned}$$

Table 2 Temperature and wildlife count in the three areas covered by the study.

	Anonymity/ untraceability	Verifiability	Unforgeability	Double-spending owner tracing	On/off-line system	Divisibility	Anonymity revocation	Built-in mutual authentication
Wang and Zhang [6]	Fail	Yes	Yes	Yes	Offline	No	No	No
Wang *et al.* [7]	Yes	Yes	Yes	Fail	Offline	No	No	No
Juang [8]	Yes	Yes	Yes	-	Online	No	No	No
Popescu and Oros [9]	Yes	Yes	Yes	Yes	Offline	No	Yes	No
Chen *et al.* [10]	Yes	Yes	Fail	Fail	Offline	No	No	No
Wang *et al.* [11]	Yes	Yes	Fail	Yes	Offline	No	Yes	No
Eslami and Talebi [12]	Yes	Yes	Yes	Fail	Offline	No	No	No
Juang and Liaw [13]	Yes	Yes	Fail	-	Offline	Yes	No	No
Chen *et al.* [19]	Yes	Yes	Yes	Yes	Offline	No	Yes	Yes
Tan [20]	Yes	Yes	Yes	Yes	Offline	No	Yes	No
Ours	Yes	Yes	Yes	Yes	Offline	Yes	Yes	Yes

$$= e(H_3(TNO)Y_B, R)e({b_1}^{-1} \cdot LST \cdot Y_B, P_{Pub})$$

(3) In the e-cash withdrawal, payment, and deposit protocols, anyone who obtains the e-cash can verify the signature of e-cash using the information of e-coin. The e-cash is valid only if the following equation holds:

$$\begin{aligned} &e(R_P, P) = e({b_2}^{-1}({R_P}' - b_3 Y_{Pub}), P) \\ &= e({b_2}^{-1}(s_P Q - b_3 Y_{Pub}), P) \\ &= e({b_2}^{-1} s_P (Q - b_3 P), P) \\ &= e({b_2}^{-1}(Q - b_3 P), Y_{Pub}) \\ &= e({b_2}^{-1}(b_2 H_1(a \,\|\, CNO \,\|\, LST) + b_3 P \\ &\quad - b_3 P), Y_{Pub}) \\ &= e(H_1(a \,\|\, CNO \,\|\, LST), Y_{Pub}) \end{aligned}$$

5.3 Untraceability

(1) During the money withdrawal with license protocol, when customer C wants to request money with his license, he must provide his identity ID_C and the blind license $\{{b_1}^{-1} \cdot LST, LVT\}$ to the bank. Although the bank knows the identity of a customer, he has no information about the LST, because it is blinded with ${b_1}^{-1}$. Therefore, it can avoid the bank B recording the LST with customer C's identity in the money withdrawal with license protocol, so that when bank B receives the LST in the deposit protocol, he cannot know the C' identity within it.

(2) In the e-cash withdrawal protocol, when C wants to request e-cash, he must provide his identity ID_C and blind hash value Q, which includes the face value of e-cash value a, the serial number of e-cash CNO, and LST. Although the cloud knows the identity of customer C, he has no information about either CNO or LST, because they are hashed and blind with b_2, b_3. After authenticating the identity of customer C, the clod L signs on Q by using his proxy signature key s_P, and output the proxy blind signature ${R_P}'$ to customer C. Then customer C unblinds it and makes the signature R_P, which is unknown to cloud L. Therefore, cloud L cannot trace the identity of customer C by $\{CNO, LST\}$ or R_P.

(3) In the payment protocol, when merchant M receives the e-cash from customer C, he cannot know the identity of customer ID_C by the e-cash. Due to customer ID_C is embedded in the LST, and it is the result of customer ID_C exclusive-or with a random number m and encrypted by trustee T's secret key K_T, which is only known by trustee T. Therefore, only when the e-cash is double spent or abused by the criminal, the bank can ask trustee T to disclose the identity of customer C. In addition, the message and session key between customer C and merchant M does not include any information about customer C, so that this protocol can keep the anonymity of customer C.

5.4 Unforgeability

Every legal customer C has a valid license issued from trustee T. However, in the payment protocol, the LST might be obtained by an adversary from a compromised merchant M. Then the adversary wants to embed the LST in a withdrawal request and pass to bank B's verification. However, without information about KGC's private key s, the adversary cannot successfully pass the verification because of the ECDLP problem. The following steps show how this attack fails.

Step 1: The adversary A obtains the LST of a legal customer C from a compromised merchant M.

Step 2: Assume that the adversary A chooses two random numbers ${b_1}^* \in {Z^*}_q$, $W^* \in G_1$ and computes the following equation:

$$\begin{aligned} &e(U^*, P_{Pub})e(H_3(({b_1}^*)^{-1} \cdot LST, U^*)Y_T, P_{Pub}) \\ &= e(U^* + H_3(({b_1}^*)^{-1} \cdot LST, U^*)Y_T, P_{Pub}) \\ &= e(s(U^* + H_3(({b_1}^*)^{-1} \cdot LST, U^*))Y_T, P) = e(V^*, P) \end{aligned}$$

5.5 e-Cash Tracing

As explained in Sect. 4.9, if the e-cash has double spent or abused by a criminal, the bank can ask trustee T to disclose the identity of the dishonest customer and revoke the anonymous e-cash by using the trustee T's secret key.

Trustee T retrieves (LST, m) from the database and uses the secret key K_T to compute ID_C as $D_{K_T}(LST) \oplus m$ so to disclose the owner's identity of the e-coin. Once T confirm the identity of customer C, trustee T can cancel the corresponding LST from database and keep the cancellation record. And the cancellation record might have bad influence for customer C's credit.

In the other hand, bank B can cooperate with trustee T to review customer C's credit periodically. If customer C has bad credit, trustee T and bank B can reject the request.

6. Comparisons

The comparisons between the proposed scheme and the related payment schemes are shown in Table 2 where it shows if these schemes can provide the anonymity (or untraceability), verifiability, unforgeability, double-spending owner tracing, on/off-line system, divisibility, and built-in mutual authentication or not. Anonymity, Verifiability, Unforgeability, and double-spending owner tracing are essential properties, which are related to privacy and security, divisibility, anonymity revocation, and built-in authentication are optional properties. For divisibility, many banks adopt a strategy similar to traditional cash to achieve the function of divisibility; in our scheme, we adopt the Cloud and Web of Things to give change to cover the function. For mutual authentication, it is used to protect the user from being cheated. In many schemes, the bank should pre-establish an authentication channel to authenticate a customer to perform the withdrawal service. However, in our proposed scheme, any party can authenticate the one who communicates with him in every transaction. Therefore, our proposed scheme is more efficient than others.

7. Conclusions

In this paper, we propose an e-cash payment system based on proxy blind signatures to overcome the problem of changes in Web of Things. This system not only provides unforgeability and verifiability, but also protects the anonymity of honest customers with an effective double-spending traceability function. Since pair-based mutual authentication and key agreement technology are adopted, each party has built-in mutual authentication. Compared with the related work, the scheme proposed by this paper can provide the properties of mutual authentication, verifiability, untraceability, unforgeability, and e-cash tracing when double spending occurs. The scheme proposed in this paper is safer, more efficient, and practical. However, if the e-cash doesn't meet the price of the merchandise or extra e-cash is needed to complete the transaction, it needs more computational cost. We'll figure out the more efficient protocol for customers and banks in the future.

Funding Statement

The research of this article was funded Ministry of Science and Technology, Taiwan, 110-2218-E-005-008-MBK and 110-2218-E-005-018.

References

[1] D. Chaum, "Blind signature for untraceable payments," Proc. Advances in Cryptology, pp.199–203, Springer-Verlag, New York, 1983.
[2] D. Chaum, "Online Cash Checks," Proc. Advances in Cryptology-Eurocrypt'89, LNCS#434, pp.288–293, Springer, 1990.
[3] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," Proc. Advances in Cryptology-Crypto'88, LNCS#403, pp.319–327, Springer, 1990.
[4] S. Solat, "Security of electronic payment systems: A comprehensive survey," arXiv, abs/1701.04556, 2017.
[5] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signature: delegation of the power to sign messages," IEICE Trans. Fundamentals, vol.E79-A, no.9, pp.1338–1354.
[6] H. Wang and Y. Zhang, "Untraceable off-line electronic cash flow in e-commerce," Proc. 24th Australasian Computer Science Conference (ACSC), Gold Coast, Qld., pp.191–198, 2001.
[7] H. Wang, J. Cao, and Y. Zhang, "A flexible payment scheme and its role-based access control," IEEE Trans. Knowl. Data Eng., vol.17, no.3, pp.425–436, 2005.
[8] W.-S. Juang, "D-cash: a flexible pre-paid e-cash scheme for date-attachment," Electronic Commerce Research and Applications, vol.6, no.1, pp.74–80, 2007.
[9] C. Popescu and H. Oros, "An off-line electronic cash system based on bilinear pairings," Proc. 14th Interneational Workshop on Systems, Signals and Image Processing, Maribor, pp.438–440, 2007.
[10] X. Chen, F. Zhang, and S. Liu, "ID-based restrictive partially blind signatures and applications," Journal of System and Software, vol.80, no.2, pp.164–171, 2007.
[11] S. Wang, Z. Chen, and X. Wang, "A new certificateless electronic cash scheme with multiple banks based on group signatures," Proc. 2008 IEEE International Symposium on Electronic Commerce and Security, Guangzhou City, pp.362–366, 2008.
[12] Z. Eslami and M. Talebi, "A new untraceable off-line electronic cash system," Electronic Commerce Research and Applications, vol.10, no.1, pp.59–66, 2011.
[13] W.-S. Juang and H.-T. Liaw, "A practical anonymous multi-authority e-cash scheme," Applied Mathematics and Computation, vol.147, no.3, pp.699–711, Jan. 2004.
[14] H. Tewari and A. Hughes, "Fully anonymous transferable E-cash," Cryptology ePrint archive, Report 2016/107.
[15] M. Scheir, J. Balasch, A. Rial, B. Preneel, and I. Verbauwhede, "Anonymous Split E-Cash—Toward Mobile Anonymous Payments," ACM Transactions on Embedded Computing Systems, vol.14, no.4, pp.1–25, 2015.
[16] F. Zhou, Y. Li, Q. Zhou, J. Miao, and J. Xu, "The electronic cash system based on non-interactive zero-knowledge proofs," International Journal of Computer Mathematics, vol.93, no.2, pp.239–257, 2016.
[17] V. Vadgama, B. Tanti, C. Modi, and N. Doshi, "A novel approach for E-payment using virtual password system," International Journal on Cryptography and Information Security (IJCIS), vol.1, no.1, pp.1–11, Dec. 2011.
[18] C. Patil, K. Hittinhalli, P. Manjunath, K.M. Praveen, and B.B.S. Kumar, "E-CASH: A novel approach of virtual money transaction using smart cards," International Journal of Advance Research and Innovative Ideas in Education, vol.2, no.5, pp.157–161, 2017.
[19] Y. Chen, J.-S. Chou, H.-M. Sun, and M.-H. Cho, "A novel electronic cash system with trustee-based anonymity revocation from pairing," Electronic Commerce Research and Application, vol.10, no.6, pp.673–682, 2011.
[20] Z. Tan, "An off-line electronic cash scheme based on proxy blind signature," The Computer Journal, vol.54, no.4, pp.505–512, 2011.

[21] A. Shamir, "Identity-based cryptosystems and signature schemes," Proc. Advances in Cryptology-Crypto'84, LNCS#196, pp.47–53, Springer, 1984.
[22] Z. Tan, "An off-line electronic cash scheme based on proxy blind signature," The Computer Journal, vol.54, no.4, pp.505–512, 2011.
[23] C.I. Fan, W.Z. Sun, and S.M. Huang, "Provably secure randomized blind signature scheme based on bilinear pairing," Computers and Mathematics with Applications, vol.60, no.2, pp.285–293, 2010.

Iuon-Chang Lin received the Ph.D. in Computer Science and Information Engineering in March 2004 from National Chung Cheng University, Chiayi, Taiwan. He is currently a professor and chair of the Department of Management Information Systems, National Chung Hsing University, Taichung, Taiwan. His current research interests include electronic commerce, information security, blockchain security, and cloud computing.

Chin-Chen Chang received his Ph.D. degree in computer engineering from National Chiao Tung University. His first degree is Bachelor of Science in Applied Mathematics and master degree is Master of Science in computer and decision sciences. Both were awarded in National Tsing Hua University. Dr. Chang served in National Chung Cheng University from 1989 to 2005. His current title is Chair Professor in Department of Information Engineering and Computer Science, Feng Chia University, from Feb. 2005. Prior to joining Feng Chia University, Professor Chang was an associate professor in Chiao Tung University, professor in National Chung Hsing University, chair professor in National Chung Cheng University. He had also been Visiting Researcher and Visiting Scientist to Tokyo University and Kyoto University, Japan. During his service in Chung Cheng, Professor Chang served as Chairman of the Institute of Computer Science and Information Engineering, Dean of College of Engineering, Provost and then Acting President of Chung Cheng University and Director of Advisory Office in Ministry of Education, Taiwan. Professor Chang has won many research awards and honorary positions by and in prestigious organizations both nationally and internationally. He is currently a Fellow of IEEE and a Fellow of IEE, UK. And since his early years of career development, he consecutively won Outstanding Talent in Information Sciences of the R. O. C., AceR Dragon Award of the Ten Most Outstanding Talents, Outstanding Scholar Award of the R. O. C., Outstanding Engineering Professor Award of the R. O. C., Distinguished Research Awards of National Science Council of the R. O.C., Top Fifteen Scholars in Systems and Software Engineering of the Journal of Systems and Software, and so on. On numerous occasions, he was invited to serve as Visiting Professor, Chair Professor, Honorary Professor, Honorary Director, Honorary Chairman, Distinguished Alumnus, Distinguished Researcher, Research Fellow by universities and research institutes. His current research interests include database design, computer cryptography, image compression and data structures.

Hsiao-Chi Chiang received the B.S. degree in Information Management from Yuan Ze University, Chung-Li, Taiwan, Republic of China, in 2010; the M.S. in Management Information System from National Chung Hsing University, Taichung, Twain, in 2012. She is currently a Project Manager of ACOM Networks, Taipei, Taiwan. Her current projects are focused on Telecommunication and Machine Learning.