Reducing Computations in Chang *et al.*'s on-line Electronic Check

Zhili Zhou^{1*}, Ching-Nung Yang² and Chih-Fei Chen²

¹*Jiangsu Engineering Center of Network Monitoring & School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, Jiangsu, 210044 China*

²*Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, 974 Taiwan
zhou_zhili@163.com*

Abstract

With the recent popularity of shopping online, on-line electronic check (OLEC) as low-risk and low-cost way to perform payment has become an important component in e-commerce. Recently, Chang et al. propose an OLEC by using hash chain to represent the maximum face value and the face amount on an e-check. However, these values of Chang et al.'s OLEC cannot be arbitrary, and it is infeasible to compute hash operations for too large values. In this paper, we adopt a decimal based hash chain to design an enhanced OLEC (EOLEC). Our EOLEC significantly reduces the hash operations in Chang et al.'s OLEC to provide a much faster transaction, and meanwhile retains its original characteristics. Also, our EOLEC with arbitrary maximum face value and face amount on a check is more like the traditional paper check.

Keywords: *E-check; E-commerce; One-way hash function; Decimal representation; Hash chain*

1. Introduction

On-line electronic check (OLEC) has become a low-risk and low-cost way to perform payment in e-commerce, because of the rapid development of Internet and cloud technologies [1][2]. According to the amounts of money in an electronic payment, there are two categories of on-line payment schemes: micropayment scheme [3-5] and macropayment scheme. A micropayment scheme has a small amount of money and a high frequency in transaction. Thus, the security of micropayment scheme is often based on the low-overhead hashing function. Because of a larger amount of money, macropayment schemes require the heavy-weight encryption, *e.g.*, public key cryptosystem, digital signature and so on, to achieve the more rigorous security requirement. Generally, macropayment schemes are used by most e-commerce websites, and there are various macropayment schemes, *e.g.*, on-line credit card payment, electronic cash [6-8] and electronic check (e-check) [9-13]. The on-line credit card (OLCC) is the most popular way we pay in e-commerce transaction. For example, 3-D Secure is a mature technology, which offers customers a secure online payment by credit card.

For purchases of merchandise, sometimes it is necessary to write a paper check. If merchants can offer customers an alternate payment option paying by on-line electronic check (OLEC), it can enrich the cash flow and customers will like it. Moreover, OLEC provides a much faster, low-risk and low-cost way than the traditional paper checks. Therefore, OLEC can achieve promising performance in e-

commerce. Recently, Chang *et al.* [13] propose an OLEC by using hash chain to represent the maximum face value and the face amount on an e-check. Also, the information of a payee can be attached to an e-check in a transaction. The security of Chang *et al.*'s OLEC is based on one-way hash function, blind signature, and public key cryptosystem. However, the maximum face value and the face amount on an e-check cannot be arbitrary. Chang *et al.*'s OLEC is infeasible for computing hash operations for too large values. To avoid the heavy hash operations, in [15], Chen *et al.*, do not use hash chain for representing/verifying the face amount of e-check but their scheme needs a secure channel.

In this paper, we adopt a decimal based hash chain to design an enhanced OLEC (EOLEC). To provide a much faster transaction, our EOLEC significantly reduces hash operations in Chang *et al.*'s OLEC, and meanwhile retains its original characteristics. The rest of this paper is organized as follows. Section 2 describes Chang *et al.*' OLEC and Chen *et al.*' OLEC. Section 3 introduces our EOLEC and its design concept. Security analysis, discussion and comparison are given in Section 4. Conclusion is drawn in Section 5.

2. Previous Works

2.1. Chang *et al.*'s OLEC

Chang *et al.*, propose an OLEC with uniqueness, robustness, non-repudiation and mutual authentication [13, 14]. Here, we briefly describe the registration phase and the paying phase in Chang *et al.*'s OLEC. As shown in Figure 1, this scheme embraces the following entities: payer (**PR**), payee (**PE**) and bank (**B**). All entities have public and private key pairs (pk_{PR}, sk_{PR}) , (pk_{PE}, sk_{PE}) and (pk_B, sk_B) , respectively. Also, ID_{PR} , ID_{PE} and ID_B are the identities of **PR**, **PE** and **B**. The notation $H(\cdot)$ is a one-way hash function, and the notation $E(\cdot)_k$ denotes an encryption function with a key k shared by **PR** and **PE**.

Registration Phase:

In this phase, a payer **PR** makes a personal on-line checking account from the bank **B**, which defines the maximum face value of an e-check as " w " dollars and the number of times that **PR** can use this e-check book as " j ".

(Step 1) A payer **PR** chooses a random integer r and calculates $m = H(ID_{PR} \parallel H^w(r))$ and $\alpha = H(m)$, where " $\parallel$ " is the concatenation operation and $H^w(\cdot)$ means the hash function being performed for w times.

(Step 2) A payer **PR** sends the opening e-check request $\{ID_{PR}, \alpha\}$ to **B**, and tells **B** the identity of a payer to request an e-check book.

(Step 3) After receiving the opening e-check request, **B** verifies the identity of payer ID_{PR} and calculates $\alpha' = H^j(m)$. Also, computes a RSA signature $s = (\alpha')^{sk_B} \bmod n$ for α' . Then, **B** sends $\{s, j\}$ to **PR**.

(Step 4) After receiving $\{s, j\}$, **PR** verifies $s^{pk_B} \bmod n \stackrel{?}{=} H^j(m)$ and then stores this e-check book (m, s, j) .

Paying Phase:

In this phase, a payer **PR** purchases products or services, and makes a payment by e-check. A transaction how a payer **PR** writes an a -dollar e-check, where $a \leq w$, is shown below.

(Step 1) A payer **PR** chooses two random integers R and b , and calculates $k = R^{pk_{PE}} \cdot b \bmod n'$. Then sends the value $\{k\}$ to **PE**.

(Step 2) After receiving $\{k\}$, **PE** obtains $k' = k^{sk_{PE}} = R \cdot b^{sk_{PE}} \bmod n'$, and returns the value $\{k'\}$ to **PR**.

(Step 3) By receiving $\{k'\}$, **PR** calculates $M = k' \cdot R^{-1} = b^{sk_{PE}} \bmod n'$, $C_1 = H^{w-a}(r) \oplus M$ and $C_2 = E_k(i \| T)$, where i is the number of times that the e-check book is used and T is the current timestamp.

(Step 4) **PR** verifies $b \stackrel{?}{=} M^{pk_{PE}} \bmod n'$. If it is valid, **PR** sends $\{ID_{PR}, ID_B, a, b, j, s, C_1, C_2, T\}$ to **PE**.

(Step 5) **PE** verifies $s^{pk_B} \bmod n \stackrel{?}{=} H^j(H(ID_{PR} \| H^a(C_1 \oplus b^{sk_{PE}} \bmod n')))$. If it is valid, **PE** will send $\{ID_{PR}, ID_{PE}, a, b, j, s, C_1, C_2, T\}$ to **B** for double spending check. Otherwise, this e-check is rejected.

(Step 6) When receiving $\{ID_{PR}, ID_{PE}, a, b, j, s, C_1, C_2, T\}$ from **PE**, the bank **B** checks the following: (i) checking whether the e-check account in the bank or not, (ii) checking whether the spent money a is over than payer's deposit or not, (iii) checking whether $(T' - T)$ is within the valid time interval or not, where T' is the current time, and (iv) verifying $s^{pk_B} \bmod n \stackrel{?}{=} H^i(H^{j-i}(m))$.

(Step 7) If all the above verifications pass, **B** accepts this e-check and informs **PE**. Meantime, **B** deducts a dollars from the account of **PR** and deposits this payment to the account of **PE**.

(Step 8) **PE** receives the notice of "Accept/Reject" from **B**. If the message is "Accept", **PE** provides products/services to **PR**, or he rejects this e-check.

Figure 1. Registration and Paying Phases of Chang et al.'s OLEC.

2.2. Chen *et al.*'s OLEC

Chen *et al.*, [16] conquer some weaknesses in Chang *et al.*'s OLEC, the non-anonymity, the time synchronization and the large computation. They use the anonymous identity $CID_{PR}=H(ID_{PR}\oplus d)$ of a payer **PR** to achieve the anonymous paying. By adopting nonce instead of timestamp, Chen *et al.*'s OLEC can resist the replay attack and meanwhile avoid the problem of time synchronization. In this paper, we deal with the reduction of hash operations in Chang *et al.*'s OLEC. Therefore, we use a simple block diagram of Chen *et al.*'s OLEC (see Figure 2) to show how it modifies the operation $m = H(ID_{PR} \parallel H^w(r))$ in Step 1 during registration phase, and the operations $C_1 = H^{w-a}(r) \oplus M$ (Step 3) and $s^{pk_B} \bmod n \stackrel{?}{=} H^j(H(ID_{PR} \parallel H^a(C_1 \oplus b^{sk_{PE}})))$ (Step 5) in paying phase, to reduce computing hash operations in Chang *et al.*'s OLEC.

Registration Phase:

(Step 1) A payer **PR** sends the identity of a payer $\{ID_{PR}\}$ to **B** via a secure channel, and requests an e-check book.

(Step 2) After receiving the opening e-check request, **B** verifies the identity of payer ID_{PR} . Then, **B** selects a random integer d to calculate the anonymous identity of **PR** $CID_{PR}=H(ID_{PR}\oplus d)$ and its signature $r_i = (CID_{PR})^{sk_B} \bmod n$. Also, calculates $\alpha = H(CID_{PR} \oplus w \oplus j)$. Then, **B** sends $\{CID_{PR}, j, \alpha, r_i, w\}$ to **PR** via a secure channel.

(Step 3) After receiving $\{CID_{PR}, j, \alpha, r_i, w\}$, **PR** verifies $\alpha \stackrel{?}{=} H(CID_{PR} \oplus w \oplus j)$ and then stores this e-check book.

Partial Paying Phase:

Here, we only show a part of paying phase that **PR** writes an a -dollar e-check, and **PE** verifies the e-check. The checking processes of double spending and the times that e-check book is used will be omitted here.

(Step 1-Step2) Same as (Step1) and (Step 2) in Chang *et al.*'s OLEC, but use $(b \oplus N_1)$ instead of b , where N_1 is a random nonce.

(Step 3) By receiving $\{k'\}$, **PR** computes $M = k' \cdot R^{-1} = (b \oplus N_1)^{sk_{PE}} \bmod n'$, $C_1 = H(CID_{PR} \oplus a \oplus M)$ and $C_2 = E_{sk_{PR}}(j \parallel a \parallel N_2) \oplus r_i$, where N_2 is a random nonce.

(Step 4) **PR** verifies $(b \oplus N_1) \stackrel{?}{=} M^{pk_{PE}} \bmod n'$. If it is valid, **PR** sends $\{C_1, C_2, a, b, N_1, N_2, CID_{PR}, ID_B\}$ to **PE**.

(Step 5) **PE** verifies $C_1 \stackrel{?}{=} H(CID_{PR} \oplus a \oplus ((b \oplus N_1)^{sk_{PE}} \bmod n'))$. If it is valid, **PE** forwards $\{CID_{PR}, N_2, a, C_2\}$ to **B** for further checking process. Otherwise, this e-check is rejected.

(Other steps): omitted here.

Figure 2. A Simplified Block Diagram of Chen *et al.*'s OLEC.

3. The Enhanced On-line Electronic Check

For example, “1. Introduction”, should be Times New Roman 13-point boldface, initially capitalized, flush left, with one blank line before, and one blank line after. Use a period (“.”) after the heading number, not a colon.

3.1. Observation

Table 1 summarizes the hash operations in Chang *et al.*'s OLEC, which may have heavy computations. In fact, in the traditional paper check, there is no limit of the maximum face value “*w*” as long as there are sufficient funds in the bank account of a payer. Theoretically, the field used for “amount” has 10 digits in a check, so the money above 10^{10} cents (10 billion USD dollars), which would require 11 digits, cannot be processed.

Table 1. The Heavy Hash Operations in Chang *et al.*'s OLEC.

Step/Phase	Operation	Number of $H(\cdot)$ operation	Should operation be finished in real time?
Step 1 in registration phase	$m = H(ID_{PR} \parallel H^w(r))$	$(w+1)$	NO
Step 3 in registration phase	$\alpha' = H^j(m)$	j	NO
Step 4 in registration phase	$s^{pk_B} \bmod n \stackrel{?}{=} H^j(m)$	j	NO
Step 3 in paying phase	$C_1 = H^{w-a}(r) \oplus M$	$(w-a)$	YES
Step 5 in paying phase	$s^{pk_B} \bmod n \stackrel{?}{=} H^j(H(ID_{PR} \parallel H^a(C_1 \oplus b^{sk_{PE}} \bmod n')))$	$(a+j+1)$	YES
Step 6 in paying phase	verify $s^{pk_B} \bmod n \stackrel{?}{=} H^i(H^{j-i}(m))$	i	YES

Although the registration phase is not necessarily processed in real time, it is infeasible to perform a hash operation $m = H(ID_{PR} \parallel H^w(r))$ with $w=10^9$ (see Step 1 in registration phase). Thus, Chang *et al.*'s OLEC cannot achieve the maximum face value like a traditional paper check. However, considering a real scenario that checks also may be cashed at check cashing stores and retailers including Wal-Mart, 7-Eleven and Kmart. For example, Wal-Mart can cash a check with a maximum amount of

\$5,000 [17]. Next, we discuss “ j ” (the number of times that the e-check book can be used) and “ a ” (the face value on a check). How many checks in a traditional checkbook do a payer get when opening a new account? It depends entirely on what kind of account a payer opens. The standards usually are 100, 125 or 150 checks in a paper checkbook. Banks generally supply you with 150 checks per paper checkbook [17]. About the face value of a check, the average face value per check is about 500 US dollars [18, 19].

From the above description, suppose that OLEC has $w=5000$, $j=150$ and $a=500$, which has the similar features with the traditional paper check. Consider Step 1, Step 3 and Step 4 in registration phase. Computations of $\alpha' = H^j(m)$ (Step 3) and $s^{pk_B} \bmod n \stackrel{?}{=} H^j(m)$ (Step 4) with $j=150$ in non real-time are still reasonable. For $w=5000$, the computation of $m = H(ID_{PR} \parallel H^w(r))$ with $w=5000$ has heavy computations even though it does not have to be finished in real time.

During paying phase, banks in OLEC have to immediately verify whether the payer has enough money in his account to cover the face value in a check, and a payee also should check the face amount and authenticate the payer, so that the payee won't hesitate to receive the check from a payer. Thus, the operations in paying should be finished in real time. Consider Step 3, Step 5 and Step 6 in paying phase. The computation of hash operations in $C_1 = H^{w-a}(r) \oplus M$ (Step 3) with $(w-a)=4500$ and $s^{pk_B} \bmod n \stackrel{?}{=} H^j(H(ID_{PR} \parallel H^a(C_1 \oplus b^{sk_{PE}} \bmod n')))$ (Step 5) with $(a+j+1)=651$ in real time are hard to accomplished. For the hash operation of $s^{pk_B} \bmod n \stackrel{?}{=} H^i(H^{j-i}(m))$ (Step 6) with $j=150$, bank can store $H^{j-i}(m)$ and do the hash operation for i times to verify the i th e-check. The number of required hash operations ranges between 1 to 150 times, and this verification is feasible for a bank server.

Chen *et al.*'s OLEC discards the hash chain and puts the values of w and j into a hash function $\alpha = H(CID_{PR} \oplus w \oplus j)$, and thus there is no need to perform heavy hash operations. However, it sends the plaintext w and j to a prayer (see Step 2 in registration phase), and thus Chen *et al.*'s OLEC needs a secure channel. On the other hand, in Chang *et al.*'s OLEC, the authors sets w and j as the number of hash operations (*i.e.*, $H^w(\cdot)$ and $H^j(\cdot)$) in $m = H(ID_{PR} \parallel H^w(r))$ and $s = (H^j(m))^{sk_B} \bmod n$. This is the why Chang *et al.*'s OLEC does not need the secure channel.

3.1. Design Concept

By the observations from Table 1, we have to reduce the hash operations in three cases: (i) Step 1 in registration phase (although it is not necessary to finish in real time, the operations of $H^w(r)$ is infeasible for a very large w (say 10^9), (ii) Step 3 in paying phase (should be finished in real time), and (iii) Step 5 in paying phase (should be finished in real time).

Case 1 (Step 1 in registration phase):

Consider the hash operation $H^w(r)$ in $m = H(ID_{PR} \parallel H^w(r))$. Suppose that the value of w has D digits, and its decimal representation is $(w_{D-1}, w_{D-2}, \dots, w_0)$, where w_{D-1} and w_0 are the most and least significant digits. For the need of verification, in our EOLEC, we have to let $w_{D-2} = \dots = w_0 = 9$. For example, for $w=5000$ (*i.e.*, $w_3=5, w_2=w_1=w_0=0$), we should choose the maximum face value $w=5999$ (*i.e.*, $w_3=5, w_2=w_1=w_0=9$). This choice of $w=5999$ more than the original 5000 is reasonable (note: a large w more conforms to present situation of a traditional check). Also, let $h(\cdot)$ be a hash function

with $(|H|/D)$ -bit output, where $|H|$ is the output length of the hash function $H(\cdot)$. We modify $m = H(ID_{PR} \parallel H^w(r))$ as the following.

$$m = H(ID_{PR} \parallel h^{w_{D-1}}(r) \parallel h^{w_{D-2}}(r) \parallel \dots \parallel h^{w_0}(r)) \quad (1)$$

For $w=5999$, the number of hash operations of $H^w(r)$ is 5999, while $(h^{w_{D-1}}(r) \parallel h^{w_{D-2}}(r) \parallel \dots \parallel h^{w_0}(r))$ in Eq. (1) only needs $(w_3 + w_2 + w_1 + w_0) = (5+9+9+9)=32$ hash operations.

Case 2 (Step 3 in paying phase):

Consider the hash operation $H^{w-a}(r)$ in $C_1 = H^{w-a}(r) \oplus M$. Since $a \leq w$, the value of a also has D digits. Suppose that the decimal presentation of a is $(a_{D-1}, a_{D-2}, \dots, a_0)$. We can modify $C_1 = H^{w-a}(r) \oplus M$ as the following.

$$C_1 = (h^{w_{D-1}-a_{D-1}}(r) \parallel h^{w_{D-2}-a_{D-2}}(r) \parallel \dots \parallel h^{w_0-a_0}(r)) \oplus M \quad (2)$$

Because the output length of $h(\cdot)$ is $|H|/D$ bits, $h^{w_{D-1}-a_{D-1}}(r) \parallel h^{w_{D-2}-a_{D-2}}(r) \parallel \dots \parallel h^{w_0-a_0}(r)$ has the same length as $H^{w-a}(r)$. Thus, the modified C_1 has the same length as the previous one. For $w=5999$ and $a=500$ (note: $a_3=0, a_2=5, a_1=0, a_0=0$), the number of hash operations of $H^{w-a}(r)$ is 5499. The following equation shows that Eq. (2) only needs 27 hash operations.

$$\begin{cases} (w_3 - a_3) + (w_2 - a_2) + (w_1 - a_1) + (w_0 - a_0) \\ = (5 - 0) + (9 - 5) + (9 - 0) + (9 - 0) \\ = 5 + 4 + 9 + 9 = 27 \end{cases} \quad (3)$$

Case 3 (Step 5 in paying phase):

We first consider the hash operation $H^a(C_1 \oplus b^{sk_{PE}} \bmod n')$ in $s^{pk_B} \bmod n \stackrel{?}{=} H^j(H(ID_{PR} \parallel H^a(C_1 \oplus b^{sk_{PE}} \bmod n')))$. Because C_1 is modified as Eq. (2), we can recover $h^{w_{D-1}-a_{D-1}}(r) \parallel h^{w_{D-2}-a_{D-2}}(r) \parallel \dots \parallel h^{w_0-a_0}(r)$ from $C_1 \oplus b^{sk_{PE}} \bmod n'$. Therefore, we use Eq. (4) instead of $H^a(C_1 \oplus b^{sk_{PE}} \bmod n')$ to hold this verification.

$$h^{a_{D-1}}(h^{w_{D-1}-a_{D-1}}(r)) \parallel h^{a_{D-2}}(h^{w_{D-2}-a_{D-2}}(r)) \parallel \dots \parallel h^{a_0}(h^{w_0-a_0}(r)) \quad (4)$$

Lemma 1: The verification

$s^{pk_B} \bmod n = H^j(H(ID_{PR} \parallel h^{a_{D-1}}(h^{w_{D-1}-a_{D-1}}(r)) \parallel h^{a_{D-2}}(h^{w_{D-2}-a_{D-2}}(r)) \parallel \dots \parallel h^{a_0}(h^{w_0-a_0}(r))))$ is true.

Proof: Since

$h^{a_{D-1}}(h^{w_{D-1}-a_{D-1}}(r)) \parallel h^{a_{D-2}}(h^{w_{D-2}-a_{D-2}}(r)) \parallel \dots \parallel h^{a_0}(h^{w_0-a_0}(r)) = h^{w_{D-1}}(r) \parallel h^{w_{D-2}}(r) \parallel \dots \parallel h^{w_0}(r)$, we can derive the following.

$$\begin{cases} H^j(H(ID_{PR} \parallel h^{a_{D-1}}(h^{w_{D-1}-a_{D-1}}(r)) \parallel h^{a_{D-2}}(h^{w_{D-2}-a_{D-2}}(r)) \parallel \dots \parallel h^{a_0}(h^{w_0-a_0}(r)))) \\ = H^j(H(ID_{PR} \parallel h^{w_{D-1}}(r) \parallel h^{w_{D-2}}(r) \parallel \dots \parallel h^{w_0}(r))) \\ = H^j(m) \text{ (from Eq. (1))} \\ = \alpha' = s^{pk_B} \bmod n \text{ (} \because s = (\alpha')^{sk_B} \bmod n \text{)} \end{cases} \quad (5)$$

For $a=500$, the number of hash operations of $H^a(C_1 \oplus b^{sk_{PE}})$ is 500, while Eq. (4) only needs $(a_3 + a_2 + a_1 + a_0) = 5$ hash operations.

Secondly, we consider the hash operation $H^j(m)$ in Eq. (5). Compared with the value of a , the value j is not a critical issue. Either we can perform the hash operation for j times, or we can reduce the hash operation by the same argument. Suppose that j has J digits and the decimal representations is $(j_{J-1}, j_{J-2}, \dots, j_0)$. We can modify $H^j(m)$ as Eq. (6). For $j=150$, the number of hash operations of $H^j(m)$ is 150, while Eq. (7) only needs $(1 + j_2 + j_1 + j_0) = 7$ hash operations. Notice that this approach for $H^j(\cdot)$ can also be applied for Step 3 and Step 4 in registration phase.

$$H(H^{j_{J-1}}(m) \| H^{j_{J-2}}(m) \| \dots \| H^{j_0}(m)) \quad (6)$$

3.2. Our EOLEC

Based on the proposed decimal based hash chain, we can significantly reduce the hash operations of Chang's OLEC. Our EOLEC provides the fast computation even for the large w , a and j . We only modify some parts of hash operations, and thus our EOLEC does not need a secure channel like Chen *et al.*'s OLEC [14]. This new EOLEC with less hash operations can be implemented on mobile devices feasibly and let e-check be used ubiquitously. Our registration phase and paying phase are shown below.

Registration Phase:

In this phase, a payer PR applies a personal on-line checking account from the bank B, which defines the maximum face value of an e-check as "w" dollars and the number of times that PR can use this e-check book as "j".

(Step 1) A payer PR chooses a random integer r and calculates $m = H(ID_{PR} \| h^{w_{D-1}}(r) \| h^{w_{D-2}}(r) \| \dots \| h^{w_0}(r))$ and $\alpha = H(m)$.

(Step 2) Same as the step in Chang et al.'s OLEC.

(Step 3) After receiving the opening e-check request, B verifies the identity of payer ID_{PR} and calculates $\alpha' = H(H^{j_{J-1}}(m) \| H^{j_{J-2}}(m) \| \dots \| H^{j_0}(m))$. Also, gives a RSA signature $s = (\alpha')^{sk_B} \bmod n$ for α' . Then, B sends $\{s, j\}$ to PR.

(Step 4) After receiving $\{s, j\}$, PR can verify $s^{pk_B} \bmod n \stackrel{?}{=} H(H^{j_{J-1}}(m) \| H^{j_{J-2}}(m) \| \dots \| H^{j_0}(m))$ and then stores this e-check book (m, s, j) .

Paying Phase:

In this phase, a payer PR purchases products or services, and makes a-dollar payment by e-check. A transaction how a payer PR pays a dollars is shown below.

(Step 1-Step2) Same as (Step1) and (Step 2) in Chang et al.'s OLEC.

(Step 3) Same as (Step 3) in Chang et al.'s OLEC, but calculate $C_1 = (h^{w_{D-1}-a_{D-1}}(r) \| h^{w_{D-2}-a_{D-2}}(r) \| \dots \| h^{w_0-a_0}(r)) \oplus M$.

(Step 4) Same as (Step 4) in Chang et al.'s OLEC.

(Step 5) PE decrypts C_1 and recovers $h^{w_{D-1}-a_{D-1}}(r)$, $h^{w_{D-2}-a_{D-2}}(r)$, ..., and $h^{w_0-a_0}(r)$. Then, calculates m by $m = H(ID_{PR} \| h^{a_{D-1}}(h^{w_{D-1}-a_{D-1}}(r)) \| h^{a_{D-2}}(h^{w_{D-2}-a_{D-2}}(r)) \| \dots \| h^{a_0}(h^{w_0-a_0}(r))) = H(ID_{PR} \| h^{w_{D-1}}(r) \| h^{w_{D-2}}(r) \| \dots \| h^{w_0}(r))$. By verifying

$s^{pk_B} \bmod n \stackrel{?}{=} H(H^{j_{j-1}}(m) \| H^{j_{j-2}}(m) \| \dots \| H^{j_0}(m))$, if it is valid PE sends $\{ID_{PR}, ID_{PE}, a, b, j, s, C_1, C_2, T\}$ to B for double spending check. Otherwise, this e-check is rejected.

(Step 6-Step 8) Same as (Step 6-Step 8) in Chang *et al.*'s OLEC.

4. Security Analysis and Comparison

4.1. Security Analysis

Our main objective is to reduce hash operations in Chang *et al.*'s OLEC. In fact, we only change the hash chain in Chang's OLEC to the decimal based has chain. Thus, we give a precise assessment of the security for modified parts, which are summarized in the following equations. Eqs. (7-1), (8-1) and (9-1) are our modified parts: Step 1 and Step 3 in registration phase, and Step 3 in paying phase. Other phases are slightly modified according to these modifications. If we can show that these modifications are secure as the originals (Eqs. (7-2), (8-2) and (9-2)), our EOLEC would have the same security like Chang's OLEC.

Figure 3. The Hash Chains Micropayment Scheme and OLEC: (a) Rivest and Shamir's Micropayment (b) Chang's OLEC (c) our EOLEC.

$$\begin{cases} m = H(ID_{PR} || h^{w_{D-1}}(r) || h^{w_{D-2}}(r) || L || h^{w_0}(r)) \\ m = H(ID_{PR} || H^w(r)) \end{cases} \quad (7-1)$$

$$(7-2)$$

$$\begin{cases} \alpha' = H(H^{j_{j-1}}(m) || H^{j_{j-2}}(m) || L || H^{j_0}(m)) \\ \alpha' = H^j(m) \end{cases} \quad (8-1)$$

$$(8-2)$$

$$\begin{cases} C_1 = (h^{w_{D-1}-a_{D-1}}(r) || h^{w_{D-2}-a_{D-2}}(r) || L || h^{w_0-a_0}(r)) \oplus M \\ C_1 = H^{w-a}(r) \oplus M \end{cases} \quad (9-1)$$

$$(9-2)$$

Chang *et al.*'s OLEC adopts hash operation for verifying the e-check, which is very similar to the well-known Rivest and Shamir's micropayment [3] based on one-way hash chain. The hash chains of Rivest and Shamir's micropayment, Chang *et al.*'s OLEC and our EOLEC are illustrated in Figure 3.

All these hash chains are based on a cryptographically strong hash function, where calculation of $y=H(x)$ is easy whereas the calculation of the inverse $x=H^{-1}(y)$ is infeasible. Figure 3(a) shows that **B** generates a payword chain of length n (say n dollars), $\{p_0, p_1, \dots, p_n\}$, where $p_i=H(p_{i+1})$ for $i=n-1, n-2, \dots, 0$, and $p_n=r$. Here, an anchor p_0 is used to verify this payword chain by **PE**, and the chains $\{p_0, p_1, \dots, p_n\}$ are sent to **PR** as digital coins. In paying phase, **PR** sends a payment p_a for a dollars, and **PE** can verify $p_0 \stackrel{?}{=} H^a(p_a)$ for an a -dollar payment. Due to the one-way property, **PE** only has the anchor p_0 and he cannot counterfeit the paywords. After verifying the payment of a dollars from **PR**, **PE** will store p_a instead of p_0 as an anchor. Thus, **PR** cannot doubly spend the digital coin p_a . The hash chain of Chang *et al.*'s OLEC (see Figure 3(b)) is very similar to Rivest and Shamir's hash chain. The difference is that the payword $p_a = H^{n-a}(r)$ can only be used once, while the value of $H^{w-a}(r)$ in Chang *et al.*'s OLEC can be repeatedly used. The reasons are as follows: (i) $H^{w-a}(r)$ is used to denote the face amount of check but not a spent digital cash (ii) $H^{w-a}(r)$ is encrypted in C_1 by XOR-ing M , which is different in various paying phase (iii) even for the same face value a (i.e., using the same $H^{w-a}(r)$), the random integer b and the timestamp T are various for each e-check.

From the above description, although the payword $p_a = H^{n-a}(r)$ and the value of $H^{w-a}(r)$ are all used to represent a dollars, their intrinsic characters are different. Chang *et al.*'s hash chain is more secure than Rivest and Shamir's because it is encrypted in C_1 . As shown in Figure 3(c), our decimal based hash chain can be processed in parallel, which can significantly reduce hash operations. For the security, we need to evaluate the randomness of m , α' and C_1 between Chang *et al.*'s OLEC and our EOLEC.

Randomness of m generated from Eq. (7-1) and Eq. (7-2):

Suppose that the hash functions $H(\cdot)$ and $h(\cdot)$ are strong hash functions and could undergo the severe collision attack and the preimage attack [20]. For $x_1=3$ and $x_2=3000$, their outputs ($y_1 = h^{x_1}(r)$ and $y_2 = h^{x_2}(r)$) should have the same randomness. Although the values of w may be several thousands, and the values of $w_{D-1} (\leq 9)$ and $w_{D-2} = \dots = w_0 (=9)$ are small, the randomness of $m = H(ID_{PR} \parallel h^{w_{D-1}}(r) \parallel h^{w_{D-2}}(r) \parallel \dots \parallel h^{w_0}(r))$ and $m = H(ID_{PR} \parallel H^w(r))$ should be the same.

The possible problem is that our input of hash function ($ID_{PR} \parallel h^{w_{D-1}}(r) \parallel \overbrace{h^9(r) \text{ repeat } (D-1) \text{ times}}^{h^9(r) \text{ repeat } (D-1) \text{ times}} \parallel h^9(r)$) ($\because w_{D-2} = \dots = w_0 = 9$) is a little bit regular in the last $(D-1) \times |H|/D$ bits. Although this does not compromise the security, it can be easily solved by using the following modification.

$$m = H(ID_{PR} \parallel h^{w_{D-1}}(r + 2^{D-1}) \parallel h^{w_{D-2}}(r + 2^{D-2}) \parallel \dots \parallel h^{w_0}(r + 2^0)) \quad (10)$$

Randomness of α' generated from Eq. (8-1) and Eq. (8-2):

Same as the analysis of the randomness of m , although the values of j is larger than the values of j_i (≤ 9), where $0 \leq i \leq (J-1)$, $\alpha' = H(H^{j_{j-1}}(m) \| H^{j_{j-2}}(m) \| \dots \| H^{j_0}(m))$ and $\alpha' = H^j(m)$ should have the same randomness.

Randomness of C_1 generated from Eq. (9-1) and Eq. (9-2):

By the same argument, although the values of $(w-a)$ is larger than the values of $(w_i - a_i)$ (≤ 9), where $0 \leq i \leq (D-1)$, $C_1 = (h^{w_{D-1}-a_{D-1}}(r) \| h^{w_{D-2}-a_{D-2}}(r) \| \dots \| h^{w_0-a_0}(r)) \oplus M$ and $C_1 = H^{w-a}(r) \oplus M$ should have the same randomness. If the modification in Eq. (10) is used, the calculation of C_1 should be modified accordingly as follows.

$$C_1 = (h^{w_{D-1}-a_{D-1}}(r+2^{D-1}) \| h^{w_{D-2}-a_{D-2}}(r+2^{D-2}) \| \dots \| h^{w_0-a_0}(r+2^0)) \oplus M \quad (11)$$

Except the above modifications, our EOLEC and Chang et al.'s OLEC are the same. Thus, our EOLEC can resist the replay attack, the electronic check book loss, the forgery attack, the impersonating attack, DoS attack, and the double spending.

4.2. Discussion and Comparison

All these three OLECs (Chang et al.'s OLEC, Chen et al.'s OLEC and our EOLEC) provide electronic check services. Other important issues among these schemes are further discussed: (i) channel type (ii) anonymity (iii) replay attack (iv) maximum face value and face amount on a check (v) number of hash operations in hash chain.

Channel type:

In Chen et al.'s OLEC, secure channel between bank and payer (or payee) should be used. Obviously, using secure channel is a simple way to achieve security. However, for some reason, there is no secure channel available. Then, Chen et al.'s OLEC does not work. For example, in Step 2 of registration phase, **B** sends $\{CID_{PR}, j, \alpha, r_i, w\}$ to **PR** via a secure channel. If this message is sent via an insecure channel, attackers can intercept this information. Although attackers do not have the private key sk_{PR} to compute $C_2 = E_{sk_{PR}}(j \| a \| N_2) \oplus r_i$, they can generate a correct $C_1 = H(CID_{PR} \oplus a \oplus M)$. By impersonating **PR**, attackers send the $\{C_1, C_2, a, b, N_1, N_2, CID_{PR}, ID_B\}$ to **PE**. **PE** would verify $C_1 \stackrel{?}{=} H(CID_{PR} \oplus a \oplus ((b \oplus N_1)^{sk_{PE}} \bmod n'))$. If it is valid, **PE** will forward $\{CID_{PR}, N_2, a, C_2\}$ to **B** for further checking process. Because attackers can generate the correct C_1 , which can pass the verification of **PE**. Although the information $\{CID_{PR}, N_2, a, C_2\}$ to **B** cannot pass the further checking. However, the verification of **PE** passes, and **PE** needs to forward the information $\{CID_{PR}, N_2, a, C_2\}$ to **B** every time. Thus, attacker can perform the denial-of-service attack on the bank to exhaust bank computation resources in further checking process.

Anonymity:

Chen et al.'s OLEC uses the anonymous CID_{PR} for the identity of **PR**. However, in registration phase, **PR** sends the identity of a payer $\{ID_{PR}\}$ to **B**, and **B** sends $\{CID_{PR}, j, \alpha, r_i, w\}$ to **PR**. Both messages need to be sent relying on an additional secure channel. Suppose that there is no secure channel. Obviously, attackers can figure out the relationship between ID_{PR} and CID_{PR} , i.e., the anonymity is compromised. On the other hand, in most case the anonymity does not needs to be confirmed in an online e-check method. Therefore, Chen et al.'s OLEC propose a trivial approach, via a secure channel, to achieve the anonymity.

Replay attack:

Chen *et al.*'s OLEC adopts nonce to resist the replay attack, and meanwhile can avoid the time synchronization problem. Chang *et al.*'s OLEC and our EOLEC use timestamping against replay attack. Timestamp is another way of preventing a replay attack. In most cryptographic protocols, both timestamp (*i.e.*, lifetime) and nonce are often used to address the replay attack. When using the approach of nonce, schemes need to generate a random number by pseudo-random number generator (PRNG). The advantage of using timestamp is that we do not need PRNG, while time synchronization should be achieved by using a secure protocol. As we know, recently, network time protocol (NTP) is a mature technology for clock synchronization. Thus, it is obvious that we can use timestamp to handle relay attack.

Maximum Face value and face amount on a check:

An e-check should perform the same function as a traditional paper check. As described in sub Section 3.1, there is no limit of the maximum face value " w ", and the average face value per check is about 500 US dollars. Therefore, the hash chain in Chang *et al.*'s OLEC (see Figure 3(b)) is unsuitable for computations for an unlimited w . Our decimal based hash chain processes D (say w has D digits) hash chains in parallel, and the maximum hash length of each chain is "9". Therefore, our EOLEC can significantly reduce hash operations in Chang *et al.*'s OLEC. On the basis of this decimal based hash chain, the values of a , j and w of our EOLEC can be arbitrary. Chen *et al.*'s OLEC puts the values of a , j and w into a hash function, or encrypts them. So, it has the same feature (arbitrary a , j and w) like our scheme (note: it needs a secure channel). For this reason, our EOLEC and Chen *et al.*'s OLEC are more like traditional paper checks.

Number of hash operations in hash chain:

Chen *et al.*'s OLEC does not use hash chain to represent and verify the face amount of e-check. Thus, we only compare the hash operations of our EOLEC with those of Chang *et al.*'s OLEC. Here, we use an example that a payer writes an e-check with face amount $a=468$ USD, and this e-check has the maximum face value $w=2500$ USD (for Chang's OLEC, our w will be 2999) and has $j=100$ e-checks. All parameters in our EOLEC can be determined from the values of a , w and j , where $(a_3, a_2, a_1, a_0) = (0, 4, 6, 8)$, $(w_3, w_2, w_1, w_0) = (2, 9, 9, 9)$ and $(j_2, j_1, j_0) = (1, 0, 0)$. Let the number of hash operations for calculating m (Step 1 in registration phase), α' (Step 3 in registration phase), C_1 (Step 3 in paying phase), and verifications in registration phase (Step 4) and paying phase (Step 5) be N_m , $N_{\alpha'}$, N_{C_1} , N_{VR} and N_{VP} , respectively. For our EOLEC, these numbers are $N_m = (w_3 + w_2 + w_1 + w_0 + 1) = 30$, $N_{\alpha'} = N_{VR} = (j_2 + j_1 + j_0 + 1) = 2$, $N_{C_1} = (w_3 + w_2 + w_1 + w_0) - (a_3 + a_2 + a_1 + a_0) = 12$ and $N_{VP} = (a_3 + a_2 + a_1 + a_0) + (j_2 + j_1 + j_0) + 2 = 21$. Chnag et al.'s OLEC has $N_m = (w+1) = 2501$, $N_{\alpha'} = N_{VR} = j = 100$, $N_{C_1} = (w-a) = 2032$ and $N_{VP} = (a+j+1) = 569$. In fact, the most critical phase about computation loading is the calculation of C_1 by payer in paying phase and the verification by payee in paying phase. The following shows the upper bounds of N_{C_1} and N_{VP} for our EOLEC:

$$N_{C_1} = (w_{D-1} + w_{D-2} + \dots + w_0) - (a_{D-1} + a_{D-2} + \dots + a_0) \leq 9D \quad \text{and}$$

$$N_{VP} = (a_{D-1} + a_{D-2} + \dots + a_0) + (j_{J-1} + j_{J-2} + \dots + j_0) + 2 \leq 9D + 9J + 2.$$

Suppose that w is a four-digit decimal value. Our upper bound of N_{C_1} is 36, while Chang *et al.*'s N_{C_1} is

about several thousands. In the next lemma, we theoretically prove that the numbers of hash operations of our EOLEC are lesser than those of Chang et al.'s OLEC.

Lemma 2: The numbers N_m , $N_{a'}$, N_{VR} , N_{C_1} and N_{VP} of our EOLEC are lesser than those of Chang *et al.*'s OLEC.

Proof: We first summarize all equations we need to prove in Eq. (12).

$$\begin{cases} N_m : (w_{D-1} + w_{D-2} + \dots + w_0 + 1) \leq (w + 1) \\ N_{a'}, N_{VR} : (j_{J-1} + j_{J-2} + \dots + j_0 + 1) \leq j \\ N_{C_1} : (w_{D-1} + w_{D-2} + \dots + w_0) - (a_{D-1} + a_{D-2} + \dots + a_0) \leq (w - a) \\ N_{VP} : (a_{D-1} + a_{D-2} + \dots + a_0) + (j_{J-1} + j_{J-2} + \dots + j_0) + 2 \leq (a + j + 1) \end{cases} \quad (12)$$

For N_m , because $(w_{D-1}, w_{D-2}, \dots, w_0)$ is the decimal representation of w , we can derive the following.

$$\begin{cases} w + 1 = (w_{D-1} \times 10^{D-1} + w_{D-2} \times 10^{D-2} + \dots + w_0 \times 10^0) + 1 \\ \geq (w_{D-1} + w_{D-2} + \dots + w_0 + 1) \end{cases} \quad (13)$$

Secondly, we prove $(j_{J-1} + j_{J-2} + \dots + j_0 + 1) \leq j$ for the case $j > 9$. Note: the hypothesis of j larger than 9 is definitely reasonable (please see subsection 3.1). Because $(j_{J-1}, j_{J-2}, \dots, j_1, j_0)$ is the decimal representation of j (>9), it implies that there is at least one value in $\{j_1, j_2, \dots, j_{J-1}\}$ is nonzero. Then, we can derive the following.

$$\begin{cases} j = j_{J-1} \times 10^{J-1} + j_{J-2} \times 10^{J-2} + \dots + j_1 \times 10^1 + j_0 \times 10^0 \\ \geq (j_{J-1} + j_{J-2} + \dots + j_0) + (10^i - 1) \times j_i \text{ (suppose that } j_i \neq 0, i \in [1, J-1]) \\ \geq (j_{J-1} + j_{J-2} + \dots + j_0) + 1 \end{cases} \quad (14)$$

From Eq. (13), we have $w \geq (w_{D-1} + w_{D-2} + \dots + w_0)$. By the same argument we can derive $a \geq (a_{D-1} + a_{D-2} + \dots + a_0)$. Subtracting these two equations, we can derive $(w_{D-1} + w_{D-2} + \dots + w_0) - (a_{D-1} + a_{D-2} + \dots + a_0) \leq (w - a)$. From Eq. (14), we have $(j + 1) \geq (j_{J-1} + j_{J-2} + \dots + j_0 + 2)$; also $a \geq (a_{D-1} + a_{D-2} + \dots + a_0)$. Thus, we can derive $(a + j + 1) \geq (a_{D-1} + a_{D-2} + \dots + a_0) + (j_{J-1} + j_{J-2} + \dots + j_0) + 2$.

Our advantage comes from using decimal based hash chain. From the above, the numbers of hash operations by using the conventional hash chain ($H^x(\cdot)$) and by the decimal based hash chain ($H^{x_{D-1}}(\cdot) \parallel H^{x_{D-1}}(\cdot) \parallel \dots \parallel H^{x_0}(\cdot)$), can be represented as $y = x$ and $y = (x_{D-1}, x_{D-2}, \dots, x_0)$, respectively, where $(x_{D-1}, x_{D-2}, \dots, x_0)$ is the decimal representation of x . Figure 4 gives a quantitative view to show the numbers of hash operations by using the two hash chains from $x=150$ to $x=1000$. From this figure, it can be observed that using decimal hash chain has a huge improvement.

Figure 4. Numbers of Hash Operations using Conventional Hash Chain and Decimal Based Hash Chain from 150 to 1000

According to the above discussions, from Table 2, several comparisons among the three OLECs are summarized as follows. 1) Our EOLEC and Chang *et al.*'s OLEC can be used via insecure channel, while Chen *et al.*'s OLEC sends messages relying on an additional channel, which is a limitation in practical applications. 2) Our EOLEC and Chang *et al.*'s OLEC do not achieve anonymity, but Chen *et al.*'s OLEC can achieve anonymity. However, in most case the anonymity does not need to be confirmed in an online e-check method. 3) All of these OLECs can resist the replay attack by using timestamp or nonce. 4) Our EOLEC and Chen *et al.*'s OLEC have arbitrary values of a , j and w . However, Chang *et al.*'s OLEC is infeasible for too large values, and thus cannot have the arbitrary values. 5) The number of hash operations of EOLEC is smaller than that of Chang *et al.*'s OLEC, which is infeasible to compute for too large values.

Table 2. Comparisons among OLEC Schemes

	Our EOLEC	Chang <i>et al.</i> 's OLEC	Chen <i>et al.</i> 's OLEC
channel type	can be used via insecure channel	can be used via insecure channel	needs an additional secure channel
anonymity	NO	NO	YES (using CID_{PR} and secure channel)
replay attack	YES (using timestamp)	YES (using timestam p)	YES (using nonce)

maximum face value/face amount		arbitrary values	infeasible for too large values	arbitrary values
number of heavy hash operations (computation costs)	N_m	$(w_{D-1} + w_{D-2} + \dots + w_0 + 1)$	$(w+1)$	—
	$N_{\alpha'}$	$(j_{J-1} + j_{J-2} + \dots + j_0 + 1)$	j	
	N_{VR}	$(j_{J-1} + j_{J-2} + \dots + j_0 + 1)$	j	
	N_{c_1}	$(w_{D-1} + w_{D-2} + \dots + w_0) - (a_{D-1} + a_{D-2} + \dots + a_0)$	$(w-a)$	
	N_{VP}	$(a_{D-1} + a_{D-2} + \dots + a_0) + (j_{J-1} + j_{J-2} + \dots + j_0) + 2$	$(a+j+1)$	

Therefore, our EOLEC has better overall performance than the other two methods. Our EOLEC not only has a much fewer hash operations compared with Chang *et al.*'s OLEC, but also does not need an additional secure channel. Meanwhile, it can support arbitrary values of maximum face value and face amount.

5. Conclusion

We address the problem of heavy hash operations in Chang *et al.*'s OLEC. The previous Chen *et al.*'s OLEC does not use hash chain to avoid heavy computations, but it needs an additional secure channel. In this paper, we adopt the decimal based hash chain to successfully reduce hash operations in Chang *et al.*'s OLEC, and meanwhile retain the characteristics of the original scheme. Our EOLEC provides much faster transactions than Chang *et al.*'s OLEC, and is more like the traditional paper check with arbitrary maximum face value and face amount on a check.

Acknowledgements

This work was supported by Ministry of Science and Technology, Taiwan, under Grant 104-2918-I-259-001 and 104-2221-E-259-013, National Natural Science Foundation of China (NSFC) (61602253, U1536206, 61232016, U1405254, 61373133, 61502242, 61572258), Jiangsu Basic Research Programs-Natural Science Foundation (BK20150925, BK20151530), Startup Foundation for Introducing Talent of Nanjing University of Information Science and Technology (2014r024), Open Fund of Demonstration Base of Internet Application Innovative Open Platform of Department of Education (KJRP1406, KJRP1407), and Priority Academic Program Development of Jiangsu Higher Education Institutions (PADA) Fund, China.

References

- [1] Z. Fu, X. Sun, Q. Liu, L. Zhou and J. Shu, "Achieving Efficient Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing", J. IEICE Transactions on Communications, vol. E98-B, no. 1, (2015), pp. 190-200.
- [2] Y. Ren, J. Shen, J. Wang, J. Han and S. Lee, "Mutual Verifiable Provable Data Auditing in Public Cloud Storage", J. Journal of Internet Technology, vol. 16, no. 2, (2015), pp. 317-323.
- [3] R. Rivest and A. Shamir, "Password and micromint: two simple micropayment schemes", Proceedings of the 1996 International Workshop on Security Protocols, Cambridge, UK., (1996) April.
- [4] K. Chaudhary, X. Dai and J. Grundy, "Experiences in developing a micro-payment system for peer-to-peer networks", J. International Journal of Information Technology and Web Engineering, vol. 5, no. 1, (2010), pp. 23-42.
- [5] C. Yang and C. Wu, "MSRC: (M)icropayment (S)cheme with ability to (R)eturn (C)hanges", J. Mathematical and Computer Modelling vol. 58, no. 1-2, (2013), pp. 96-107.
- [6] D. Chaum, "Blind signature for untraceable payments", Proceedings of a Workshop on the Theory and Application of Cryptographic Techniques, Santa Barbara, California, USA, (1983) August 23-25.

- [7] J. Wang, F. Yang and I. Paik, "A novel e-cash payment protocol using trapdoor hash function on smart mobile devices", J. International Journal of Computer Science and Network Security, vol. 6, (2011), pp. 12-19.
- [8] A. Swe and K. Kyaw, "Improved E-cash Protocol", J. International of Science & Technology Research, vol. 2, (2013), pp. 28-31.
- [9] D. Chaum, B. Boer, E. Heyst, S. Mjolsne and A. Steenbeek, "Efficient offline electronic check", Advances in Eurocrypt'89, Cryptographic Techniques Houthalen, Belgium, (1989), vol. 434, pp. 294-301.
- [10] D. Chaum, A. Fiat and M. Naor, "Untraceable electronic cash", 8th Annual International Cryptology Conference, Santa Barbara, California, USA, (1990) August 21-25.
- [11] W. Chen. "Efficient on-line electronic checks", J. Applied Mathematics and Computation, vol. 162, no. 3, (2005), pp. 1259-1263.
- [12] W. Hsin and L. Harn, "Simple certified e-check with a partial PKI solution", The 43rd Annual Southeast Regional Conference, Kennesaw, Georgia, Alabama, USA, (2005) March 18-20.
- [13] C. Chang, S. Chang and J. Lee, "An on-line electronic check system with mutual authentication", J. Computers and Electrical Engineering, vol. 35, no. 5, (2009), pp. 757-763.
- [14] Z. Zhou, Y. Wang, Q. Wu, C. Yang and X. Sun, "Effective and Efficient Global Context Verification for Image Copy Detection", J. IEEE Transactions on Information Forensics and Security, online (DOI: 10.1109/TIFS.2016.2601065), (2016).
- [15] C. Chen, C. Wu and W. Lin, "Improving an on-line electronic check system with mutual authentication", International Conference on Advanced Information Technologies, (2010).
- [16] The eHow website: http://www.ehow.com/info_8403545_maximum-limit-cash-checks.html.
- [17] The answerbag website: http://www.answerbag.com/q_view/2198872.
- [18] R. Schneider and B. Longjohn, "Beyond check-cashing: an examination of consumer demand and business innovation for immediate access to check funds", Center for Financial Services Innovation, (2014) June.
- [19] The Federal Deposit Insurance Corporation website: https://www.fdic.gov/bank/analytical/quarterly/2009_vol3_1/AltFinServicesprimer.html.
- [20] J. Aumasson, W. Meier and F. Mendel, "Preimage attacks on 3-Pass HAVAL and step-reduced MD5", The 15th Workshop on Selected Areas in Cryptography, Sackville, NB, Canada, (2009) August 14-15.

Authors

Zhili Zhou, he received his BS degree Communication Engineering from Hubei 2007, and his MS and PhD degrees in computer Application at the School of Information Science and Engineering from Hunan University, in 2010 and 2014, respectively. He is an assistant professor at Nanjing University of Information Science and Technology. His current research interests include digital forensics, image/video copy detection, pattern recognition, and image processing.

Ching-Nung Yang, he received the BS and MS degrees in telecommunication engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1983 and 1985, respectively, and the Ph.D. degree in electrical engineering from National Cheng Kung University, Tainan City, Taiwan, in 1997. He is currently a Full Professor with the Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan. His research interests include coding theory, information security, and cryptography.

Chih-Fei Chen, she is currently pursuing her MS degree in the Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan. Her current research interests include theory, information security, and cryptography.