High-Confidence Computing 1 (2021) 100031

Contents lists available at ScienceDirect

High-Confidence Computing

homepage: www.elsevier.com/locate/hcc

A simplified scheme for secure offline electronic payment systems

Md. Abdullah Al Rahat Kutubi [a,b], Kazi Md. Rokibul Alam [a,*], Yasuhiko Morimoto [c]

[a] Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna 9203, Bangladesh
[b] Department of Computer Science and Engineering, Bangladesh Army University of Engineering and Technology, Bangladesh
[c] Graduate School of Advanced Science and Engineering, Hiroshima University, Higashi-Hiroshima 739-8521, Japan

ARTICLE INFO

Keywords:
Offline electronic payment
Schnorr's blind signature
RSA algorithm
Zero-knowledge proof
Untraceability

ABSTRACT

This paper proposes a secure offline electronic (e-) payment scheme by adopting Schnorr's untraceable blind signature (BS). Thereby, to satisfy the essential security requirements of e-payment systems, it requires much more simple computations and becomes more practical than many existing schemes. Other considerations are: to prevent the forgery of e-coin, the Bank is only the lawful entity to produce the valid e-coin; and others can verify its correctness. To confirm no swindling, the e-coin owner also sticks her private signing key with the e-coin before spending it as the payment. Hence, through the commitment with challenge-response of Schnorr's BS, the merchant can verify the spent e-coin, and the trusted authority can identify the dishonest spender if multiple spending occurs. Moreover, it embeds three distinct information of date, namely expiration, deposit, and transaction dates with every e-coin. Thereby, it minimizes the size of the Bank's database, correctly calculates the interest of the e-coin, and helps in arbiter if multiple spending, respectively. Finally, it evaluates the performance and analyzes essential security requirements of the proposed scheme, plus studies a comparison with existing ones.

1. Introduction

The advancement of computing technologies is targeting to substitute paper cash by electronic (e-) cash, hard wallet by e-wallet [1], etc. By deploying cryptographic protocols along with distributed computing systems, the e-payment system arranges the trading between a customer and a merchant through e-coin. As involved parties do not need to communicate face to face, it saves their time and conveyances. Recently, various computer apps are also aiming to ensure the convenient and faster transfer of funds based on near field communication (NFC) technique to facilitate the Bank by reducing its costs and usage of cheques [2]. However, vulnerabilities identified in these apps, *e.g.*, forgery of the cash, unauthorized access, illegitimate handling, attempt for multiple spending, etc., have compromised their privacy issues. In contrast, cryptography-based e-payment systems, capable enough to maintain the privacy and confidentiality issues of involved entities, have drawn the attention of researchers.

Based on the mode of connectivity of the 3rd party (*e.g.*, the Bank, the trusted central authority (CA), etc.) within the system, broadly, the category of e-payment system is either online or offline. In an online system [3–7], usually, the Bank can check transactions between a customer and a merchant instantaneously. Thereby, the Bank can identify any illegal trade and can control it easily. However, it turns the system to the bottleneck situation owing to vast real-time security checking for transactions and sacrifices the anonymity of customers at some level. Conversely, the offline system [8–11] does not allow the simultaneous presence of any 3rd party during the trading. Additionally, it enables the merchant to deposit his earned e-coin safely to his corresponding Bank account (A/C) at a preferable time. Thereby, an offline system leads to less congestion. Furthermore, the offline mode remains in two categories, *i.e.*, identified and anonymous. The stunning feature of the anonymous system is maintaining the privacy of every honest customer. Here, the customer is anonymous against the merchant, the Bank, the CA, or any other 3rd party during payment as long as she acts legitimately, *i.e.*, she does not do any fraud transaction. Hence, research on the anonymous offline system has drawn a great attention. An e-payment scheme suitable for practical trading must satisfy security requirements such as anonymity, unlinkability, unforgeability, multiple spending check, no-swindling, no impersonation, no framing, fraudster control, conditional traceability, offline payment, date attachability, etc. [8,12,13,14,15].

A Schnorr's blind signature (BS) [16] based offline e-payment scheme proposed in this paper outperforms the efficiency and security of many schemes, *e.g.*, Hwang et al.'s RSA BS based offline one proposed in [14]. The reason is, Schnorr's BS uses much more simple computations to ensure untraceability. Thus, the proposed scheme becomes efficient by providing anonymity of the customer via reduced calculations. It also ensures unlinkability between the e-coin and its owner, *i.e.*, no

* Corresponding author.
E-mail addresses: rokib@cse.kuet.ac.bd (K.Md.R. Alam), morimo@hiroshima-u.ac.jp (Y. Morimoto).

https://doi.org/10.1016/j.hcc.2021.100031
Received 21 March 2021; Accepted 19 May 2021

one except the e-coin owner can know this link. Although the number of existing BS protocols in the territory of cryptography is large, most of them cannot maintain untraceability fairly and inexpensively. However, Schnorr's BS is provably secure, efficient, and it possesses characteristics of a standard BS protocol [16]. Moreover, its' strength relies on the hardness of discrete logarithm problem (DLP), and the developed scheme uses the RSA algorithm to ensure confidential communication. Additionally, every e-coin holds three distinct information of date, *i.e.*, the expiration, the deposit, and the transaction dates. Among them, the expiration date assists in minimizing the size of the Bank's database, the deposit date aids in estimating the interest of the e-coin precisely, and the transaction date helps to arbitrate if any multiple spending occurs.

The structure of the remaining portion of this paper is as follows: Section 2 studies related works. Section 3 explains the required security components. Section 4 illustrates the configuration and the individual protocols of the developed scheme. Next, Section 5 presents the experimental and security analyses. Finally, Section 6 concludes the paper.

2. Related works

In the domain of e-commerce research, numerous e-payment schemes are available. Cryptanalysis of most of these schemes also is done. The scheme proposed in [17] was the first initiative as an offline e-payment system, and it was developed based on the notion of a BS along with the cut-and-choose protocol. However, cryptanalysis of it [18] revealed that during the payment, the involved entities, *i.e.*, the Bank, the customer, the merchant, etc. need to exchange a vast amount of data which degrades its efficiency. Later on, several schemes, *e.g.*, [19,20,21], etc. attempted to overcome the flaws of the cut-and-choose protocol. Some other proposed schemes, *e.g.*, [8–11,22], etc. also aimed to satisfy the requirements of an ideal offline e-payment system. However, cryptanalysis of them also proved their inefficiency and susceptibility to certain significant security flaws, *e.g.*, schemes [8,10,23] that introduced divisibility and transferability of e-coin became practically useless. The reason is, the transaction history glued to e-coin for detecting the scam payment [14] will increase the size of the e-coin and it may restrict the number of transactions to control the size of the e-coin.

The offline e-cash scheme known as Mondex proposed in [21] exploited the public key cryptosystem, implemented for payment card's microchip, and still, MasterCard Inc. exercises it. However, the customer needs to disclose her identity while purchasing a Mondex card, which demolishes her anonymity. The scheme proposed in [19] introduced the presence of a trusted 3rd party. Additionally, to manage the congestion overhead during payment, it refrains the Bank from a continuous connection within the system, but cryptanalysis identified its weakness, *e.g.*, erroneous representation of the customer's identity. Later on, several schemes proposed in [22,24], etc. suggested some solutions to the flaw. The method proposed in [25] engaged the Bank rather than the trusted 3rd party for revealing the identity of a dishonest customer who committed a double spend of its e-coin. But the involvement of the Bank creates new security flaws such as the possibility of knowing the link between the e-cash and its owner by an issuer Bank or by an attacker [26]. Herein, the attacker intrudes within the issuer Bank's system by collecting information about withdrawal and deposit transactions. Employing ElGamal digital signature (DS) and RSA BS, the scheme proposed in [12] aimed to maintain security against various possible frauds. However, the cryptanalysis also identified several flaws such as it cannot prevent multiple spending and forging the expiration date. Although the scheme developed in [8] attempted to solve those flaws, it failed because the Bank is unable to trace the real identity of the attacker when double spending is detected [27].

Utilizing the automorphic BS, the group BS, and the Groth-Sahai zero-knowledge proof (ZKP), the scheme proposed in [28] attained optimal anonymity. It also developed a different structure of the e-coin, *i.e.*, divided the transferred e-coin into two parts, and targeted to overcome the limitations of most existing transferable optimal sized e-payment systems. The scheme developed in [29] was the first efficient divisible e-coin system. It exercised a fixed period for the withdrawal and the payment protocols and appointed the Bank to identify the double-spending quickly. The scheme proposed in [18] also targeted to fulfill transferability, divisibility, etc. security requirements. It was developed for mobile devices and based on ElGamal DS along with Schnorr's identification protocol. The scheme proposed in [30] adopted a new security requirement, namely, the change-giving problem (*i.e.*, permitting an online shop to give a change of e-cash to the customer) alongside the usual ones. Here to settle the change-giving issue and to conduct the withdrawal protocol, it utilized an existing group BS protocol. Another scheme proposed in [42] exploited the simple partially BS and did not involve any trusted third party. Due to low computation, it is suitable for mobile client and smart-card implementation.

The scheme proposed in [31] introduced the Bitcoin and its blockchain technology, which was the first decentralized digital currency. Here online transactions occur without the involvement of the Bank or a trusted 3rd party or any single administrator. Thereby it reduces the Bank processing charges [32]. Besides, these transactions are publicly transmitted and recorded in a publicly readable blockchain ledger [33]. However, Bitcoin can usually support about seven transactions per second, which specifies the inability of blockchain as a high-capacity and high-frequency payment system [34]. This low transaction rates in the blockchain systems due to increased transaction costs and vast network congestion further inspired researchers to apply off-chain and offline e-payment schemes. The scheme proposed in [2] initiated the BS to preserve the privacy of users in blockchain. The method proposed in [33] exploited blockchain to track the relevant transaction constraints that are used to lessen the chance to be unfair. But it was unable to provide a real-time guard against a multiple-spending that creates the possibility of an immense scam before any inspection. Moreover, the scheme proposed in [34] presented a secure, versatile light payment based on blockchain, which exploited the off-chain and offline payments. It had inspired the creation of many other new e-cash such as schemes proposed in [36,37], etc.

The analyses of existing BS based offline e-payment schemes show that lots of them sacrifice untraceability due to multiple spending control and conditional traceability. Although the scheme proposed in [14] achieved essential security requirements of e-payment systems, still it is inefficient. It is based on Hwang et al.'s BS [38], where the underlying computations are significantly expensive. This paper proposes a simplified e-payment scheme by adopting a customized form of Schnorr's BS [16]. Thus, it maintains the anonymity of the customer and unlinkability between the e-coin and its owner via more simple computations. It ensures the anonymity of the customer as long as she spends legally. If an unfair e-coin transaction is detected, only with the assistance of the CA, the Bank discloses the identity of the deceitful customer. Among existing BS protocols, most are unable to satisfy untraceability completely and cheaply. However, Schnorr's BS is well-known for its simplicity, amply robust and efficient as well as capable to generate short signatures [43]. The analyses in [44] also prove its security both in the algebraic group model and the random oracle model. Thus, it maintains the requirements of an ideal BS scheme. Additionally, the proposed scheme exploits the RSA algorithm and embeds the information of date within the e-coin which enable the scheme to satisfy the security requirements of ideal e-payment systems efficiently.

3. Security components

This section explains the security components required to develop the proposed e-payment scheme. These are the RSA algorithm and Schnorr's BS protocol.

3.1. RSA algorithm

The framework of both RSA DS and RSA public key cryptosystem depend on the RSA algorithm. Hence, their working procedure is almost the same. Usually, in RSA DS, the private key and the public key are used for signing and verification purposes, respectively. Whereas in RSA public key cryptosystem, the public key and the private key are used for encryption and decryption purposes, respectively. Besides, while RSA DS and RSA public key cryptosystem need to use simultaneously, their public-private key pairs must be distinct.

3.1.1. RSA digital signature

1 An involved entity A (*i.e.*, the merchant, the customer, the Bank, or the CA) picks two large-sized primes p_A and q_A. Now A computes $n_A = p_A.q_A$ and $S_A V_A \equiv 1 \ mod \ (p_A - 1)(q_A - 1)$. Then, A keeps (p_A, q_A, S_A) as the secret signing key and discloses (n_A, V_A) as the public verification key.

2 A selects a message m to be signed on it. Now using its' secret signing key S_A, entity A itself signs on m as $m_S = m^{S_A} \ mod \ n_A$. Then, m_S is known as the signed message.

3 Afterward, using A's public verification key V_A, anyone (having communication with A) can verify the signed message as $m = {m_S}^{V_A} \ mod \ n_A$.

3.1.2. RSA public key cryptosystem

1 An involved entity A (*i.e.*, the merchant, the customer, the Bank, or the CA) picks two large-sized primes $\underline{p}_A$ and $\underline{q}_A$. Now A computes $\underline{n}_A = \underline{p}_A \cdot \underline{q}_A$ and $E_A D_A \equiv 1 \ mod \ (\underline{p}_A - 1)(\underline{q}_A - 1)$. Then, A keeps ($\underline{p}_A$, $\underline{q}_A$, D_A) as the secret decryption key and discloses ($\underline{n}_A$, E_A) as the public encryption key.

2 Someone (having communication with A) selects a plaintext m to be encrypted. Now using A's public encryption key E_A, it encrypts m by calculating $m_e = m^{E_A} \ mod \ \underline{n}_A$. Then, m_e is known as the ciphertext.

3 Afterward, using its secret decryption key D_A, A retrieves the plaintext by calculating $m = {m_e}^{D_A} \ mod \ \underline{n}_A$.

3.2. Schnorr's blind signature

A BS protocol usually maintains the anonymity of the customer in an e-commerce scheme. Typically, it allows the signer to sign a blinded message of the message owner. Usually, a BS is a specialized form of the DS protocol. Schnorr's BS is developed by transforming the Schnorr's DS [39] comprises of challenge-response between a challenger (*i.e.*, the message owner) and the signer. Although this BS already exists in [16], the developed e-payment scheme again customizes it. Hereby, it lets the customer giving her signature on the merchant's message m without revealing her identity. It consists of four phases, *i.e.*, initialization, blinding and signing, verification, and revelation, and they are described below.

1 *Initialization:* To run the protocol, initially, it chooses two large primes p and q, such that $q|(p–1)$ (*i.e.*, q divides (p–1)). Further, it chooses a group G known as Schnorr group of prime order q with generator g. Now it publishes (p, q, g) for involved entities. Besides, it selects a one-way hash function H, which is also known by associated entities. The Bank then picks a private signing key s as $s \in Z/qZ$ (it is a multiplicative group of integers modulo q) and publishes the public verification key y by calculating $y = g^{-s} \ mod \ p$. Moreover, the customer chooses her secret RSA signing key as S_C.

2 *Blinding and signing:* When the customer needs a valid e-coin (*i.e.*, message m), she asks a commitment from the Bank. Before issuing this commitment and signature, the Bank chooses a random $t \in Z/qZ$, computes $a = g^t \ mod \ p$, and sends a to the customer as the challenge. The customer then blinds a by calculating $\alpha = a g^{\beta} \ mod \ p$ using secret random element β where $\beta \in Z/qZ$. Next, the Bank sends Y_C (where Y_C is already calculated by the customer as $Y_C = g^{S_C} \ mod \ N_{CS}$ and N_{CS} belongs to the customer's RSA public key) of the customer to the CA. The CA then chooses a secret random element $r \in Z/qZ$, blinds Y_C of the customer with r and encrypts r with the customer's public RSA encryption key (E_C, N_C) along with the CA's public RSA encryption key (E_{CA}, N_{CA}) as follows: $Y_r = {Y_C}^r \ mod \ p$, $r_C = E_C(r) \ mod \ N_C$, $r_{CA} = E_{CA}(r) \ mod \ N_{CA}$. Afterward, the CA sends (Y_r, r_{CA}, r_C) to the Bank. Subsequently, the Bank computes $m = (C||Y_r||r_{CA}||r_C)$, $e = H(\mathrm{m}, \alpha) \ mod \ \mathrm{p}$, and $\mathrm{R} = \mathrm{t} + \mathrm{es} \ mod \ q$ where C is the e-coin amount. After receiving (m, R, e) from the Bank, the customer computes $\rho = R + \beta \ mod \ q$ and $\sigma = \mathrm{g}^{-\beta} \ mod \ q$. Later on, for the payment, the merchant sends T (*i.e.*, transaction date-time) to the customer. Finally, the customer sends the five-tuple, *i.e.*, (m, ρ, σ, e, w) to the merchant where $w = \mathrm{S_C}.r.T - \beta$. Thus, the merchant gets the BS of the customer on his e-coin message.

3 *Verification:* Any entity can easily verify the validity of Schnorr's BS of the Bank on the e-coin since it satisfies $\mathrm{e} = H(\mathrm{m}, \mathrm{g}^{\rho}\mathrm{h}^{\mathrm{e}} \ mod \ \mathrm{p})$ easily. The merchant verifies the customer's BS on the e-coin if $\sigma \stackrel{?}{=} \mathrm{g}^{\mathrm{w}}\mathrm{Y_r}^{\mathrm{T}} mod \ q$ is yes.

4 *Revelation:* Here, conditionally unblinding the identity of the customer is only possible. The Bank with the CA can unblind the fraudulent customer's identity (*i.e.*, her secret RSA signing key$\mathrm{S_C}$) only when she performs double–spending. Let the two pair of (T, w) be (T_1, w_1) and (T_2, w_2). Two linear equations are formed from each of these two pairs as $w_1 = \mathrm{S_C}.r.T_1 - \beta$ and $w_2 = \mathrm{S_C}.r.T_2 - \beta$. After solving these two equations, the Bank can find out the value of $\mathrm{S_C}.r$ that contains the customer's $\mathrm{S_C}$. It is solved as $Q = \mathrm{S_C}.r = \frac{\mathrm{w_1} - \mathrm{w_2}}{\mathrm{T_1} - \mathrm{T_2}}$. Now, the Bank sends Q along with the e-coin (m) and the verification credentials (ρ, e, σ) to the CA to disclose the multiple spender's S_C. The CA then computes $S_C = \frac{Q}{r}$ to detect the multiple spender, and provides the information of this customer to the Bank.

Table 1
The CA's registration database.

IDs of customers and merchants	Information of customers and merchants
ID_{C1}	Name, address, etc.
ID_{M1}	Name, address, etc.
ID_{C2}	Name, address, etc.
ID_{M2}	Name, address, etc.
. . .	. . .
ID_{Cn}	Name, address, etc.
ID_{Mn}	Name, address, etc.

4. Development of an offline e-payment scheme

This section develops an offline e-payment scheme by exploiting Schnorr's BS and RSA algorithm while embedding three distinct information of date with every e-coin.

4.1. Configuration

The developed scheme involves the customer, the merchant, a trusted central authority (CA), and the Bank as entities. Their roles are explained below and Fig. 1 illustrates major interactions among them.

The central authority (CA)

(1) The CA acts as the trusted arbitrator, performs registration of other participating entities, and verifies the identity of the customer and the merchant. The CA also helps the Bank to open a Bank account (A/C) for any valid customer or merchant. Besides, the CA maintains a database (as shown in Table 1) that stores the information corresponding to the ID of the customer, and the merchant formed by signing with the CA's private key.

(2) Using the RSA algorithm, the CA keeps two separate key pairs. One pair is for public encryption–private decryption {*i.e.*, (E_{CA}, N_{CA}), (D_{CA}, p_{CA}, q_{CA})} and the other pair is for private signing–public verification {*i.e.*, (S_{CA}, p_{CAS}, q_{CAS}), (V_{CA}, N_{CAS})}.

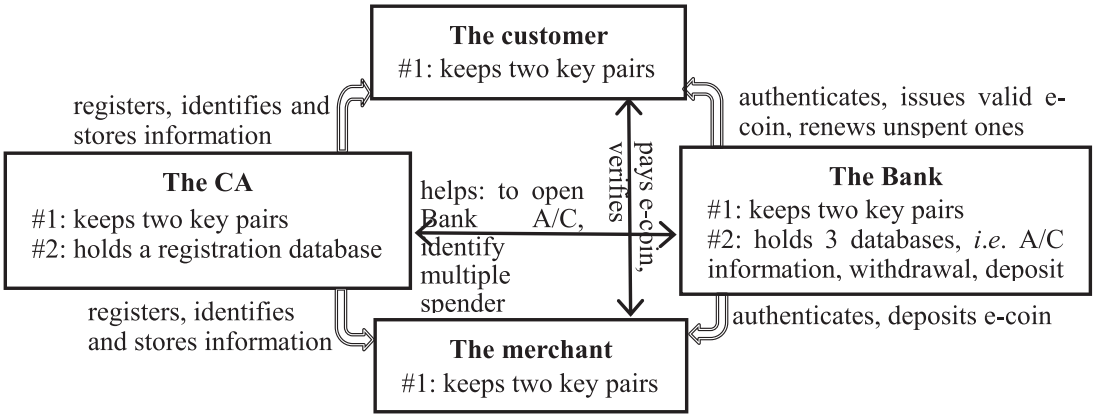

Fig. 1. The dataflow diagram of interactions among entities.

Table 2
The bank's account (A/C) infomation database.

A/C number (customers and merchants)	PIN hash code	A/C's balance ($)
AID_{C1}	PHC_{C1}	2000
AID_{M1}	PHC_{M1}	5000
AID_{C2}	PHC_{C2}	4000
AID_{M2}	PHC_{M2}	3000
. . .		. . .
AID_{Cn}	PHC_{Cn}	2000
AID_{Mn}	PHC_{Mn}	1000

Table 3
The bank's withdrawal database.

E-coin*	E-coin unique id	Additional info
$m_1, (R_1, e_1)$	C_{id1}	α_1
$m_2, (R_2, e_2)$	C_{id2}	α_2
. . .	. . .	. . .
$m_n, (R_n, e_n)$	C_{idn}	α_n

* e-coin with verification credentials.

Table 4
The bank's deposit database.

E-coin*	Date-time of	
	Deposit	Transaction
$m_1, (\rho_1, e_1, \sigma_1, w_1, T_1)$	D_{DT1}	T_{DT1}
$m_2, (\rho_2, e_2, \sigma_2, w_2, T_2)$	D_{DT2}	T_{DT2}
. . .	. . .	. . .
$(m_n, (\rho_n, e_n, \sigma_n, w_n, T_n)$	D_{DTn}	T_{DTn}

* e-coin with verification credentials.

(3) The CA also plays a role in creating valid e-coin in Withdrawal protocol and helps the Bank to identify the identity of the multiple spender.

The bank

(1) The Bank authenticates every customer and merchant with their own Bank A/C number. Additionally, it controls several databases, namely, A/C information database, withdrawal database, and deposit database, as presented in Table 2, Table 3, and Table 4, respectively.
(2) To manage the information of the customer and the merchant, it maintains the A/C information database (as shown in Table 2). Also, the Bank has to attach an expiration date to the e-coin asked for a withdrawal, and the withdrawn e-coin is stored in the withdrawal database (as shown in Table 3) for further usage.
(3) The Bank verifies the validity of an e-coin requested to deposit by the merchant before depositing it in the deposit database (as shown in Table 4). At deposit time, the Bank also detects multiple time spender to restrain multiple spending and discloses the liable one's identity with the help of the CA.
(4) The Bank further renews any customer's unspent but expired e-coin with a new expiration date and modifies the withdrawal database accordingly.
(5) Using the RSA algorithm, the Bank generates and maintains two separate key pairs. One pair is for public encryption–private decryption {*i.e.*, (E_B, N_{BE}), (D_B, p_{BE}, q_{BE})} and the other pair is for private signing–public verification {*i.e.*, (S_B, p_{BS}, q_{BS}), (V_B, N_{BS})}.

The customer

1 The customer can withdraw e-coins from her Bank A/C after being authenticated by the Bank as a valid Bank A/C holder. She also renews her unused but outdated e-coin from the Bank with a new expiration date.
2 Using the RSA algorithm, she generates and maintains two separate key pairs. One pair is for public encryption–private decryption {*i.e.*, (E_C, N_C), (D_C, p_C, q_C)} and the other pair is for private signing–public verification {*i.e.*, (S_C, p_{CS}, q_{CS}),(V_C, N_{CS})}.

The merchant

1 The merchant verifies the validity and ownership of e-coins received from the customer before transferring his commodities to the customer in exchange for e-coins. He also sticks transaction date T_{DT} with the valid e-coin and deposits it to the Bank.
2 Using the RSA algorithm, he generates and maintains two separate key pairs. One pair is for public encryption–private decryption {*i.e.*, (E_M, N_M), (D_M, p_M, q_M)} and the other pair is for private signing–public verification {*i.e.*, (S_M, p_{MS}, q_{MS}), (V_M, N_{MS})}.

4.2. Individual protocols of the scheme

Initially, all associated entities agree to generate their required own private parameters and announce the public parameters, as described in Sections 3 and 4.1. The developed scheme consists of six protocols, *i.e.*, Registration, Withdrawal, Payment, Deposit, Renewal, and Tracing. They proceed as follows.

4.2.1. Registration

The CA conducts the registration of the participating entities, *i.e.*, the customer and the merchant, along with the Bank. Hence, these entities provide their required information to the CA when they register themselves. The protocol proceeds as follows, and Fig. 2 depicts its dataflow (for the customer).

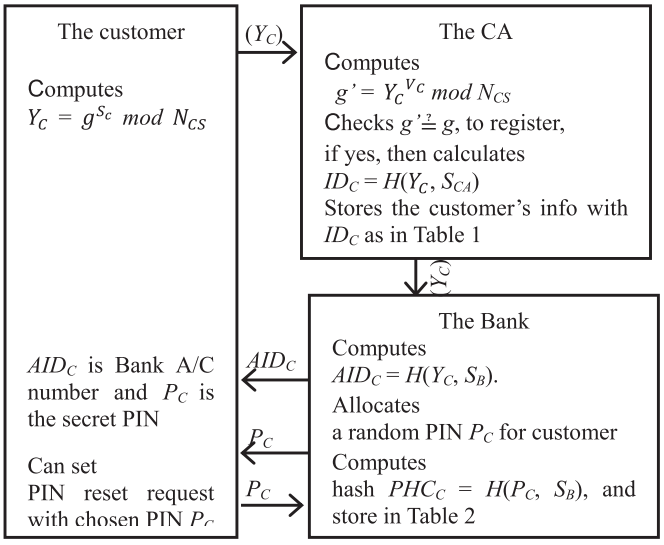

Fig. 2. The dataflow diagram of the Registration protocol (for the customer).

1 Using the RSA DS, the customer generates her secret signing key and public verification key as S_C and V_C, respectively where S_C is considered as the customer's secret ID number (CSIDN).
2 After that, the customer blinds her S_C by discrete logarithmic operation (*i.e.*, calculates $Y_C = g^{S_C} \ mod \ N_{CS}$) on the public generator g of the common group G. This output Y_C is used as the customer's public ID number (CPIDN).
3 Next, the customer sends the CPIDN (Y_C) to the CA to register herself. Later on, the CA can verify this CSIDN through the customer's V_C, where a valid verification always retrieves g.
4 Subsequently, the CA computes $g' = Y_C^{V_C} \ mod \ N_{CS}$ using the customer's V_C and checks $g' \stackrel{?}{=} g$. If yes, then it registers the customer. It generates the customer's ID_C as $ID_C = H(Y_C, S_{CA})$, where S_{CA} is the CA's secret signing key. It stores the identity information of the customer along with ID_C, as shown in Table 1.

Opening the bank A/C

To open a Bank A/C for the registered customer, the CA interacts with the Bank as follows.

(1) The CA → The Bank (Y_C): The CA sends CPIDN (*i.e.*, Y_C) to the Bank to open a Bank A/C corresponding to Y_C.
(2) The Bank → The customer (AID_C): After receiving the A/C opening request from the CA, the Bank opens a Bank A/C corresponding to Y_C. For this purpose, the Bank creates the customer's unique Bank A/C number, *i.e.*, $AID_C = H(Y_C, S_B)$, where S_B is the Bank's RSA secret signing key. The Bank manages a Bank A/C for the A/C number AID_C as shown in Table 2. The Bank chooses a random personal identification number (PIN) P_C, computes PIN hash code (PHC) as: $PHC_C = H(P_C, S_B)$, and stores in the Bank A/C information database as in Table 2. The Bank sends AID_C as her Bank A/C number (CBAN) and P_C as a secret PIN to the customer. However, the customer resets her PIN by choosing own one and send it to the Bank. If the Bank recieves PIN reset request, it regenerates PHC with the new PIN and update the corresponding PHC in Bank's A/C information database shown in Table 2. The customer uses AID_C and PIN to be authenticated by the Bank while withdrawing the e-coin.

Similarly, the registration process of the merchant is carried out as same as the customer.

4.2.2. Withdrawal

This protocol allows a legal customer to withdraw the e-coin from her Bank A/C. For this purpose, at first, the customer gets authenticated by the Bank through her Bank A/C number, *i.e.*, AID_C and PIN P_C. This protocol proceeds as follows, and Fig. 3 depicts the dataflow of different steps of the protocol.

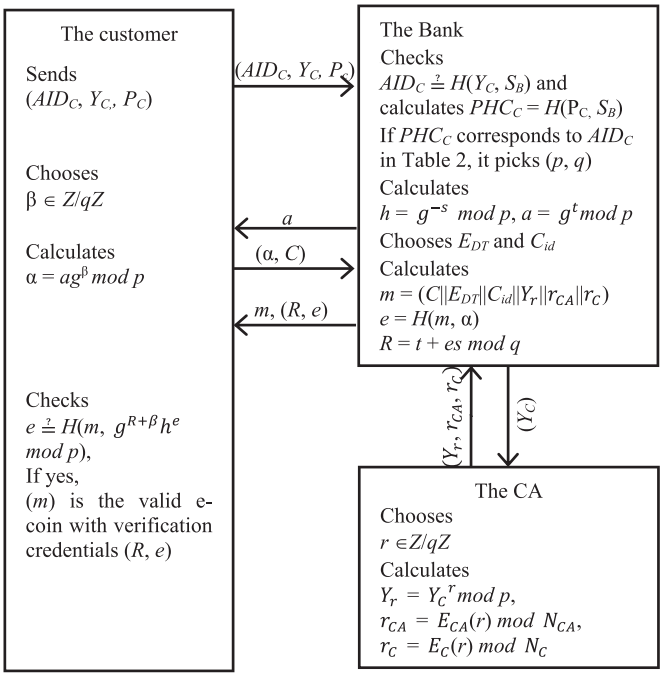

Fig. 3. The dataflow diagram of the Withdrawal protocol.

1 The customer → The Bank (AID_C, Y_C, P_C): To begin with, the customer sends her CBAN (*i.e.*, AID_C), CPIDN (*i.e.*, Y_C), and PIN (P_C) to the Bank to be authenticated. The Bank then checks $AID_C \stackrel{?}{=} H(Y_C, S_B)$. If yes, it computes $PHC_C = H(P_C, S_B)$. If PHC_C exists corresponding to AID_C in Bank A/C information database, the customer is authenticated correctly, and the Bank generates the e-coin for the customer as follows.
2 The Bank → The customer (a): Next, the Bank chooses public prime integers q and p such that $q|(p-1)$; g is an element of (Z/qZ) of order q. Then, the Bank selects the required parameters as follows (as already discussed in Section 3.2): secret $s \in Z/qZ$, public $h = g^{-s} \ mod \ p$, secret $t \in Z/qZ$, and $a = g^t \ mod \ p$. Now, the Bank sends a as the challenge to the customer to notify the withdrawing. Here, g is the public generator, and h is the Bank's public verification key.
3 The customer → The Bank (α, C): After receiving the withdrawal notification from the Bank, the customer blinds the challenge a. Hence, she chooses a secret random element $\beta \in Z/qZ$ and computes $\alpha = ag^{\beta} mod \ p$. She then sends α along with the e-coin amount C to the Bank.
4 The Bank → The CA (Y_C): When the Bank gets the final withdrawal notification by receiving (α, C) from the customer, it starts to create a valid e-coin for the corresponding customer. For this purpose, it gets help from the CA. Therefore, the Bank sends Y_C of the customer to the CA.
5 The CA → The Bank (Y_r, r_{CA}, r_C): The CA chooses a secret random element r as $r \in Z/qZ$. Now, it blinds Y_C of the customer with r as $Y_r = Y_C^{\ r} mod \ p$. Also, it encrypts r with the customer's public RSA encryption key as $r_C = E_C(r) \ mod \ N_C$. Again, the CA encrypts r using its own public RSA encryption key as $r_{CA} = E_{CA}(r) \ mod \ N_{CA}$. Then, it sends ($Y_r$, r_{CA}, r_C) to the Bank.
6 The Bank → The customer (m, R, e): Subsequently, the Bank chooses the e-coin expiration date (as date||time, *i.e.*, E_{DT}) and the unique e-coin ID (*i.e.*, C_{id}). Now, it computes $m = (C||E_{DT}||C_{id}||Y_r||r_{CA}||r_C)$, $e = H(m, \alpha)$, and $R = t + es \ mod \ q$. It then sends m as the created new valid e-coin with verification credentials (R, e) to the customer.

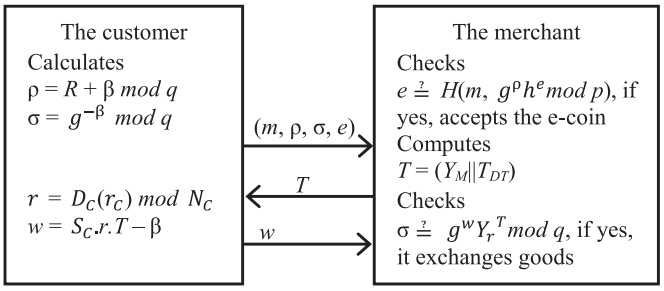

Fig. 4. The dataflow diagram of the Payment protocol.

It also debits the amount C from the Bank A/C of the customer. The Bank also stores the new e-coin in the withdrawal database, as shown in Table 3.

7 Finally, the customer checks $e \stackrel{?}{=} H(m, g^{R+\beta} h^e mod\ p)$. If yes, she accepts the valid e-coin from her A/C with the value C.

4.2.3. Payment

This protocol enables the customer to pay the merchant in exchange for merchandise without the involvement of the Bank during payment. It also empowers the merchant to verify the e-coin received from the customer. It proceeds as follows, and Fig. 4 depicts the dataflow of different steps of this protocol.

(1) The customer → The merchant (m, ρ, σ, e): Firstly, the customer computes $\rho = R + \beta\ mod\ q$, and an authentication element σ such as $\sigma = g^{-\beta}\ mod\ q$. After that, the customer sends the e-coin m with verification credentials (ρ, σ, e) to the merchant.
(2) After receiving the e-coin from the customer, the merchant checks whether $e \stackrel{?}{=} H(m, g^{\rho} h^{e}\ mod\ p)$ or not. If it is not same, the merchant terminates the protocol.
(3) Otherwise, the merchant accepts it as a valid e-coin. He then checks the expiration date-time E_{DT} to confirm that the e-coin is not outdated. If outdated, he terminates the protocol. If not, he calculates the transaction-date||time (T_{DT}) and computes $T = (Y_M||T_{DT})$ where Y_M is the merchant's public identity and he sends T to the customer.
(4) After receiving T from the merchant, the customer checks T_{DT} to confirm the validity of the transaction's date-time concerning the current date-time. If it is a valid one, the customer decrypts r_C to obtain r as $r = D_C(r_C)\ mod\ N_C$. After that, she computes $w = S_C.r.T - \beta$ and sends w to the merchant.
(5) After receiving w from the customer, the merchant checks whether $\sigma \stackrel{?}{=} g^{w} Y_r^{T} mod\ q$ or not. If yes, he accepts the e-coin and finally sells goods to the customer. Additionally, he stores the e-coin m and verification credentials (ρ, σ, e, w, and T) until he successfully deposits the e-coin. Since the e-payment system is offline, the merchant does not need to deposit the e-coin to the Bank immediately; he can deposit it at his convenient time.

4.2.4. Deposit

This protocol empowers the merchant to deposit his e-coin obtained from the customer to his Bank A/C. Herein, when the merchant goes to deposit, the Bank performs multiple-spending checking on that e-coin. If the same merchant already deposited this e-coin, then the Bank warns him. Besides, the Bank reveals the e-coin owner (*i.e.*, the customer) with the help of the CA if the coin is already spent (as it will be discussed in the Tracing protocol). This protocol proceeds as follows, and Fig. 5 depicts the dataflow of different steps of this protocol.

(1) The merchant → The Bank (AID_M, Y_M, P_M): The merchant sends his Bank A/C number (MBAN) (*i.e.*, AID_M), the merchant's public ID number (MPIDN) (*i.e.*, Y_M) and PIN (*i.e.*, P_M) to the Bank to be authenticated. The Bank then checks $AID_M \stackrel{?}{=} H(Y_M, S_B)$ or not where S_B is the Bank's RSA secret signing key. If PHC_M is found corresponding to AID_M in Bank A/C information database, the merchant is authenticated correctly. The Bank then proceeds to deposit the e-coin for the merchant as follows.

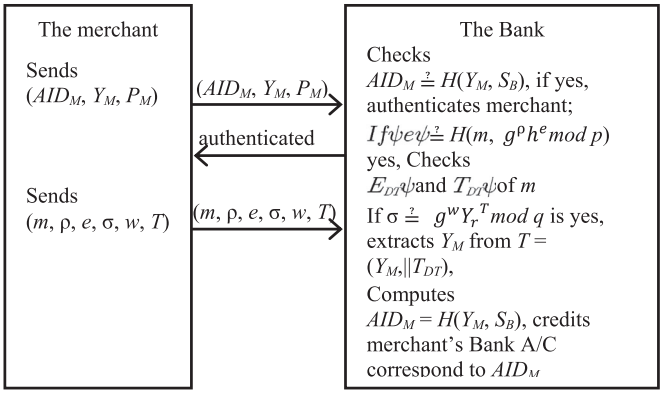

Fig. 5. The dataflow diagram of the Deposit protocol.

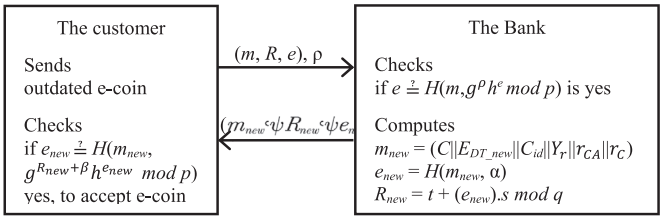

Fig. 6. The dataflow diagram of the Renewal protocol.

(2) After being authenticated, the merchant sends the coin (m) and the verification credentials (ρ, e, σ, w, T) to the Bank. The Bank then checks the coin in the deposit database if it exists or not.
(3) Additionally, the Bank verifies the coin, *i.e.*, if $e \stackrel{?}{=} H(m, g^{\rho} h^{e}\ mod\ p)$ is yes, the coin is verified validly.
(4) Next, the Bank checks the expiration date-time E_{DT} and the transaction date-time T_{DT}, respectively, of m.
(5) After that, if the Bank finds that the e-coin exists in the withdrawal database, as shown in Table 3 and does not exist in the deposit database, as shown in Table 4, it goes to step 6. Otherwise, the Bank rejects the e-coin and sends a rejection message to the merchant.
(6) Subsequently, the Bank extracts the merchant's A/C number as follows: if $\sigma \stackrel{?}{=} g^{w} Y_r^{T} mod\ q$ is yes, then gets Y_M from $T = (Y_M||T_{DT})$, computes MBAN, *i.e.*, $AID_M = H(Y_M, S_B)$ and credits the merchant's A/C corresponding to AID_M.
(7) Finally, the Bank stores the e-coin with the deposit date-time D_{DT} and the transaction date-time T_{DT}, as shown in Table 4 in the merchant's Bank A/C.

4.2.5. Renewal

This protocol facilitates the customer to renew the unspent but outdated e-coin. When an unspent e-coin expires, the customer renews it from the Bank with a new expiration date. The protocol proceeds as follows, and Fig. 6 depicts the dataflow of different steps of this protocol.

(1) Already the customer has computed $\rho = R + \beta\ mod\ q$ and $\sigma = g^{-\beta}\ mod\ q$. Now, she sends ρ along with an unspent but outdated e-coin (m) along with the verification credentials (R, e) to the Bank.
(2) The Bank then checks whether $e \stackrel{?}{=} H(m, g^{\rho} h^{e}\ mod\ p)$ or not. If no, the Bank terminates the protocol.
(3) Otherwise, the Bank checks whether the e-coin exists in the deposit database or not, as shown in Table 4. If it does not exist, the protocol proceeds as follows.
(4) The Bank → The customer (m, R, e): The Bank chooses a new expiration date E_{DT_new} to renew the e-coin and ex-

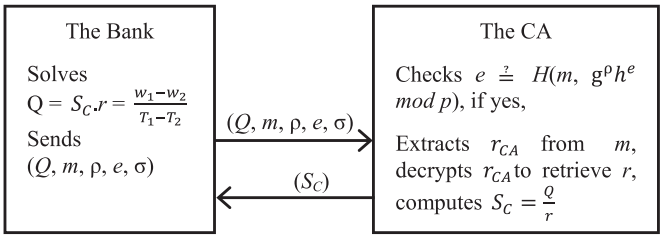

Fig. 7. The dataflow diagram of the Tracing protocol.

tracts α from the withdrawal database. Next, it computes $m_{new} = (C||E_{DT\_new}||C_{id}||Y_r||r_{CA}||r_C)$, $e_{new} = H(m_{new}, \alpha)$, $R_{new} = t + (e_{new}).s \ mod\ q$. It then sends the updated renewed e-coin m_{new} with new verification credentials (R_{new}, e_{new}) to the customer.

(5) After receiving the new e-coin from the Bank, the customer checks $e_{new} \stackrel{?}{=} H(m_{new}, g^{R_{new}+\beta} h^{e_{new}} \ mod\ p)$. If yes, she accepts it as a renewed valid e-coin.

(6) Finally, the Bank replaces the previous e-coin with the renewed one, *i.e.*, (m_{new}) along with verification credentials (R_{new}, e_{new}).

4.2.6. Tracing

This protocol is designed to identify the customer when she performs multiple spending. When an e-coin is spent more than once, the Bank receives the same e-coin multiple times from different merchants. If multiple-spending occurs, the Bank reveals the customer who owned this e-coin, with the help of the CA. This protocol proceeds as follows, and Fig. 7 depicts the dataflow of different steps of this protocol.

(1) When the Bank receives an e-coin from the merchant for depositing, the Bank checks whether the e-coin exists in the deposit database and the withdrawal database, or not. If the e-coin exists in the withdrawal database but not in the deposit database, the Bank terminates the protocol.

(2) If the e-coin exists in both databases, the Bank gets two different pairs of (T, w) for the same e-coin.

(3) Then, the Bank extracts the security parameter of the e-coin owner customer as follows.

(4) Let the two pairs of (T, w) are (T_1, w_1) and (T_2, w_2). Two linear equations are formed from each of the two pairs as follows: $w_1 = S_C.r.T_1 - \beta$ and $w_2 = S_C.r.T_2 - \beta$. After solving these two equations, the Bank can find out the value of $S_C.r$ that contains the customer's CSIDN (*i.e.*, S_C) and solves $Q = S_C.r = \frac{w_1 - w_2}{T_1 - T_2}$.

(5) The Bank sends Q along with the e-coin (m) and its' verification credentials (ρ, e, σ) to the CA to disclose the multiple time spender's CSIDN.

(6) If $e \stackrel{?}{=} H(m, g^{\rho} h^{e} \ mod\ p)$ is yes, then the CA extracts r_{CA} from m, decrypts it to retrieve r as $r = D_{CA}(r_{CA}) \ mod\ N_{CA}$, and computes $S_c = \frac{Q}{r}$.

(7) Thus, the CA detects the multiple time spender and reveals her identity to the Bank.

5. Experimental analyses

5.1. Experimental setup

In order to estimate the computation time required for different protocols of the proposed scheme, a prototype system was developed under the environment of 64-bit Windows 10 operating system. The configuration of the environment was an Intel(R) Core™ *i*7 3.60 GHz processor with 8 GBytes of RAM. For coding, GMP [40] with 1024-bit modulus was used. All computation time did not consider the communication time. Besides, operations that were not related to cryptography were not considered.

5.2. Experimental results and comparisons

Different protocols of the proposed scheme consist of authentication, blinding, signing, verification, encryption, decryption, challenge-response pair of ZKP, etc. cryptographic operations. Here, considering the customer, the merchant and the Bank singly, Table 5 shows the number of significant operations, *i.e.*, exponents (E), hashes (H), modular multiplication (M), etc. required by various protocols of the proposed scheme and the schemes proposed in [8,14] and [18] where the scheme [8] did not consider the Tracing protocol and the scheme [18] considered only three basic protocols as shown in the table. Besides, based on the computation time complexity involved in different protocols of these schemes, the table also presents another comparison in terms of equivalent computational cost where M, H, E etc. are in a 1024-bit modulus. For equivalent cost computing, according to [41], it is assumed that $E \approx 240$ M and $H \approx M$ and the table shows that the proposed scheme needs the almost lowest overall computational cost among the considered schemes. Moreover, Table 6 presents the computation time required by the proposed scheme and the scheme proposed in [14].

Again, considering M and E operations, Table 7 presents a performance comparison among the same schemes based on some other criteria, *i.e.*, the computational cost of (a) withdrawing and paying of e-coin by a customer (c_1); (b) the Bank during the Withdrawal protocol (c_2); verifying the e-coin by a merchant (c_3). Besides, the mode of transaction (c_4) and the underlying hard problem of the e-coin scheme (c_5) are also considered. From the table, it is easily understandable that the proposed scheme is more practical and efficient than other schemes. Furthermore, it satisfies the security requirements of e-payment systems more simply.

5.3. Security analyses

The proposed scheme satisfies security requirements of e-payment systems as follows.

Unforgeability: An e-cash scheme is unforgeable if no one except the Bank only can create valid coins. Suppose an adversary *Adv* wants to forge an e-coin m with verification credentials (R, e). To do so, it should change m and R and attempts to alter any component $(C||E_{DT}||C_{id}||Y_r||r_{CA}||r_C)$ of m. To create a valid e-coin, the *Adv* must need to change R ($R = t + es \ mod\ q$), but *Adv* would not be able to generate the valid Schorr's signature of the Bank over e where $e = H(m, \alpha)$ because s and t are the Bank's secret parameters of Z/qZ. There is no way to know or generate valid s and t untill the Bank discloses them to the *Adv*. Suppose, the *Adv* chooses invalid s' and t' to generate forged R'. While any entity checks the validity of Schnorr's BS of the Bank on the e-coin through $e \stackrel{?}{=} H(m, g^{\rho} h^{e} \ mod\ p)$, it will not be satisfied because h is the verification key of the Bank which was calculated as $h = g^{-s} \ mod\ p$ where s is the secret parameter chosen by the Bank.

Anonymity: An e-payment system is said to be anonymous, while no one can reveal the identity of the customer from the e-coin. Hence, to maintain this property, it is required that the element of the e-coin should not disclose the identity of the customer who got the coin in the Payment protocol or any other way. During payment, the merchant got an e-coin m with verification credentials (ρ, σ, e) and the payment challenge w from the customer. Here, $m = (C||E_{DT}||C_{id}||Y_r||r_{CA}||r_C)$ where Y_r holds the customer's public ID number (CPIDN) Y_C. Since in the Withdrawal protocol, while generating the e-coin, Y_C gets blinded into Y_r (*i.e.*, $Y_r = Y_C{}^r mod\ p$) by a random blinding factor r (only known to the customer and the CA) by the CA, no one (except the customer and the CA) can disclose Y_C from Y_r. Thus, the e-coin does not leak any information of the e-coin owner to the *Adv*. From the equation $w = S_C.r.T - \beta$, no one can disclose r because w includes two secret parameters S_C and β. Similarly, S_C cannot be revealed from w also due to secrets r and β. Thereby, it gives no information to the *Adv*.

Multiple spending detection: In case of multiple spending, the Bank would be able to find out the malicious spender's (the customer) CSIDN, (*i.e.*, S_c) through the equation $Q = S_c.r = \frac{w_1 - w_2}{T_1 - T_2}$ as described in the Trac-

Table 5
Comparision based on required number of operations and its equivalent computation cost.

Computation cost (based on computation time complexity, $E \approx 240$ M and $H \approx M$ [41])								
Scheme (→)	[8]		[18]		[14]		Proposed	
Protocol (↓)	*Opns	+Cost	*Opns	+Cost	*Opns	+Cost	*Opns	+Cost
Registration	2E+3M	≈483M	–		18E+16M	≈4336M	2E+2M+2H	≈484M
Withdrawal[1]	7E+9M	≈1689M	9E+6H+9M	≈2175M	10E+9M	≈2409M	8E+8M+3H	≈1931M
Payment[2]	5E+5M+1H	≈1206M	7E+5H+6M	≈1691M	8E+8M	≈1928M	5E+5M+1H	≈1206M
Deposit[3]	1E+2M	≈242M	4E+4H+4M	≈968M	5E+5M	≈1205M	4E+2M+3H	≈965M
Renewal	7M+9M	≈1689M	–	–	6E+6M	≈1446M	4E+3M+3H	≈966M
Tracing	–	–	–	–	4E+4M	≈964M	3E+2M+1H	≈723M
Cost (1 + 2 + 3)		3137M		4834M		5542M		4102M

* Opns = Number of operations, and +Cost = Equivalent computational cost.

Table 6
Computation time required by the proposed and another schemes.

Protocol	Required time (ms) for schemes	
	proposed	[14]
Registration	4	–
Withdrawal	18	29
Payment	13	24
Deposit	10	22
Renewal	10	–
Tracing	7	–

Table 7
Performance comparisons.

Case	[8]	[18]	[14]	proposed scheme
c_1	6E+8M	1E+1M	7E+11M	4E+5M+1H
c_2	2E+1M	9E+6H+9M	6E+6M	2E+3M+1H
c_3	6E+2M	6E+5H+5M	3E+3M	2E+2M+1H
c_4	Offline	Offline	Offline	Offline
c_5	Factoring, DLP	DLP	Factoring	Factoring, DLP

ing protocol, and subsequently, it finds her identity with the help of the CA.

Impersonation attack: One of the severe attacks by the malicious Bank is impersonation. In this case, the Bank can steal the withdrawing information of the customer and illegally withdraw the e-coin from the customer's account, and over-spends the e-coin without her permission. The proposed scheme tackles this attack as follows: (a) While registration and A/C opening, the CSIDN (S_c) of the customer remains concealed to the Bank. (b) Although the Bank can generate a valid e-coin from the customer's e-coin using the CPIDN (Y_C), it can't spend the e-coin validly because, during the payment, S_c must be used as follows (as already discussed in Section 4.2.3): $w = S_C.r.T - \beta$. Thereby, the malicious Bank or any other *Adv* can't extract S_C from the Y_C component of the e-coin as $Y_C = g^{S_C} mod\ N_{CS}$, where S_C is the discrete logarithm (DL) in the base of g. To obtain S_C, the malicious entity must be able to solve the DLP, where solving DL is assumed to be practically infeasible [8].

Framing attack: If any customer spends her same e-coin multiple-times, the Bank obtains Q as $Q = S_C.r = \frac{w_1 - w_2}{T_1 - T_2}$ and discloses her identity with the help of the CA. After multiple spending detection, the malicious Bank can create fraud signature w' as $w' = Q.T' - \beta'$ using T' and β' where $Q = S_C.r$ is already known to the Bank. Consequently, the malicious Bank can perform illegal payment or conspires with the merchant. For avoiding this situation, the customer must change her existing CSIDN (S_C) through re-registration while her identity goes revealed due to multiple spending.

Fraudster control: This section discusses possible frauds targeted by the *Adv* and how the scheme tackles them.

(a) Firstly, the merchant may also attempt to deposit an e-coin more than once. For this purpose, the merchant has to forge T' and w'. Alongside, he needs to alter the unique e-coin id (C_{id}). It was already argued (unforgeability) that no one can forge the e-coin. If the merchant tries to deposit an e-coin with T' and w', $g^{w'} {Y_r}^{T'}\ mod\ q$ never yields valid σ' (as $\sigma' != g^w {Y_r}^T\ mod\ q$) as only the customer can produce a valid $w = S_C.r.T - \beta$.

(b) Secondly, suppose a malicious merchant M gets an e-coin from the customer and deposits it to his Bank A/C. Then M tries to spend the e-coin to another merchant. The merchant can't produce such w that satisfies $\sigma = g^w {Y_r}^T\ mod\ q$, since $w = S_C.r.T - \beta$, where S_C is only known to the customer. As the merchant must check $\sigma \stackrel{?}{=} g^w {Y_r}^T\ mod\ q$ before receiving an e-coin, a malicious merchant can't perform multiple spending.

(c) Thirdly, a malicious Bank worker can alter an e-coin or withdraw from the customer's Bank A/C successfully without the permission of the customer. But it can't spend the coin.

(d) Fourthly, if someone steals the customer's e-coin, it will try to spend the coin immediately. However, it also can't do that.

(e) Fifthly, a malicious merchant M steals the e-coin and (w, T) from a valid merchant before they are submitted to the Bank and tries to deposit it. But it can't deposit the e-coin to the malicious merchant's Bank A/C, because T holds the valid merchant identity information, *i.e.*, $T = (Y_M || T_{DT})$ where Y_M is the merchant's public identity. To alter T, he needs to change w, and can't do that. Up to now, the considered fraud cases (*i.e.*, cases from a to e) do not occur because S_C is the secret identity of the customer; only she knows it, and to solving S_C is a type of DLP.

(f) If someone steals an e-coin and tries to renew it, according to the Renewal protocol, the Bank is only the lawful entity to revive the expiration date E_{DT} in m where $m = (C||E_{DT}||C_{id}||Y_r||r_{CA}||r_C)$. Since Y_r is not alterable, the stealer can't spend renewed e-coin. Where Y_r blindly holds the customer's secret S_C, and Y_r is used for verification in Payment protocol by the merchant.

6. Conclusions

By incorporating Schnorr's BS protocol, the proposed offline e-payment scheme satisfies all essential security requirements through simplified computations. Thereby, it has become practical and efficient. In the scheme, the usage of Schnorr's BS ensures the anonymity of the customer reasonably and straightforwardly. Further, the customer is anonymous as long as she remains honest, *i.e.*, not liable for multiple-spending. Additionally, it confirms other security requirements, namely, unforgeability of the e-coin, untraceability between the e-coin and its customer, multiple-spending detection, fraudster control, no-swindling, no impersonation, no framing attack, etc. altogether. The comparison of computational cost and the security analyses also attest its improvement of efficiency. A further plan of enhancement is to develop a mobile e-commerce application for the scheme.

Declaration of Competing Interest

The authors declare that they have no conflict of interest.

References

[1] Z. Bezhovski, The future of the mobile payment as electronic payment system, Eur. J. Bus. Manag. 8 (8) (2016) 127–132.
[2] H. Tewari, A. Hughes, Fully anonymous transferable Ecash, IACR Cryptol. ePrint Arch. 107 (2016).
[3] R. Martínez-Peláez, F.J. Rico-Novella, New electronic cash model: a script anonym, in: Proceedings of the IADIS International Conference on E-Commerce, 2006, pp. 392–396. e-commerce'06.
[4] D. Chaum, A. Fiat, M. Naor, Untraceable electronic cash, in: Advances in Cryptology CRYPTO88, Springer, New York, 1988, pp. 319–327.
[5] C.I. Fan, B.W. Lin, S.M. Huang, Customer efficient electronic cash protocol, J. Organ. Comput. Electron. Commer. 17 (3) (2007) 259–281.
[6] R.S. Anand, C.V. Madhavan, 'An online, transferable e-cash payment system, in: Progress in Cryptology INDOCRYPT 2000, Springer, 2000, pp. 77–91.
[7] W.S. Juang, H.T. Liaw, A practical anonymous multi-authority e-cash scheme, Appl. Math. Comput. 147 (3) (2004) 699–711.
[8] B.Takhaei Y.Baseri, J. Mohajeri, Secure untraceable off-line electronic cash system, Sci. Iran. 20 (3) (2013) 637–646.
[9] H. Wang, Y. Zhang, Untraceable off-line electronic cash flow in e-commerce, Australas. Comput. Sci. Commun. 23 (1) (2001) 191–198.
[10] Y. Chen, J.S. Chou, Cryptanalysis on "Secure untraceable off-line electronic cash system", IACR Cryptol. ePrint Arch. (2014) 63.
[11] C.H. Wang, Untraceable fair network payment protocols with off-line TTP, in: Advances in Cryptology-ASIACRYPT 2003, 2003, pp. 173–187. pp..
[12] Z. Eslami, M. Talebi, A new untraceable off-line electronic cash system, Electron. Commer. Res. Appl. 10 (1) (2011) 59–66.
[13] C.I. Fan, W.Z. Sun, H.T. Hau, Date attachable offline electronic cash scheme, Sci. World J. (2014).
[14] M.A.A.R. Kutubi, K.M.R. Alam, R. Tahsin, G. Ali, P. Chomg, Y. Morimoto, An offline electronic payment system based on an untraceable blind signature scheme, KSII TIIS 11 (5) (2017) 2628–2645.
[15] J. Muleravicius, I. Timofejeva, A. Mihalkovics, E. Sakalauskas, Security, trustworthiness and effectivity analysis of an offline E-cash system with observers, Informatica 30 (2) (2019) 327–348.
[16] D. Pointcheval, J. Stern, Provably secure blind signature schemes, in: Advances in Cryptology–Asiacrypt '96, LNCS, Vol. 1163, Springer, Berlin, Heidelberg, 1996, pp. 252–265.
[17] D. Chaum, Blind signatures for untraceable payments, in: Advances in Cryptology (CRYPTO'83), Springer, Boston, USA, 1983, pp. 199–203.
[18] E. Sakalauskas, I. Timofejeva, A. Michalkovič, J. Muleravičius, A simple off-line E-cash system with observers, J. Inf. Technol. Control 47 (1) (2018) 107–117.
[19] S. Brands, Untraceable off-line cash in wallet with observers, in: Advances in Cryptology-CRYPTO'93, Springer, 1993, pp. 302–318.
[20] F. Stalder, Failures and successes: notes on the development of electronic cash, Inf. Soc. Int. J. 18 (3) (2002) 209–219.
[21] L. Srivastava, R. Mansell, Electronic Cash and the Innovation Process: A Use Paradigm, University of Sussex, SPRU, 1998.
[22] G. Davida, Y. Frankel, Y. Tsiounis, M. Yung, Anonymity control in e-cash systems, in: Proceedings of the International Conference on Financial Cryptography, Springer, 1997, pp. 1–16.
[23] S.J. Aboud, Analysis of offline E-cash schemes, Int. J. Adv. Res. 2 (8) (2014) 406–410.
[24] A. Chan, Y. Frankel, P. MacKenzie, Y. Tsiounis, Mis-representation of identities in e-cash schemes and how to prevent it, in: Advances in Cryptology-ASIACRYPT'96, Springer, 1996, pp. 276–285.
[25] C.I. Fan, V. Huang, Y.C. Yu, User efficient recoverable offline e-cash scheme with fast anonymity revoking, Math. Comput. Model. 58 (1–2) (2013) 227–237.
[26] Y. Chen, J.S. Chou, On the privacy of user efficient recoverable off-line E-cash scheme with fast anonymity revoking, Int. J. Netw. Secur. 17 (6) (2015) 708–711.
[27] Y.F. Chang, W.L. Tai, Y.C. Liu, H.W. Chen, Vulnerability of Baseri et al.'s untraceable offline electronic cash system, in: Proceedings of the IEEE International Conference on System Science and Engineering (ICSSE), Taiwan, 2018, pp. 1–5.
[28] J. Zhang, L. Huo, X. Liu, C. Sui, Z. Li, J. Ma, Transferable optimal-size fair E-cash with optimal anonymity, in: Proceedings of the IEEE International Symposium on Theoretical Aspects of Software Engineering, 2015, pp. 139–142.
[29] S. Canard, D. Pointcheval, O. Sanders, J. Traoré, Divisible E-cash made practical, IET Inf. Secur. 10 (6) (2016) 332–347.
[30] L. Batten1, X. Yi, Off-line digital cash schemes providing untraceability, anonymity and change, Electron. Commer. Research. 19 (1) (2019) 81–110.
[31] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, [online] Available: http://bitcoin.org/bitcoin.pdf.
[32] I. Miers, C. Garman, M. Greenand, A.D. Rubin, Zerocoin: anonymous distributed E-cash from Bitcoin, in: Proceedings of the IEEE Symposium on Security and Privacy, 2013, pp. 397–411.
[33] A. Lipton, T. Hardjono, A. Pentland, Digital trade coin: towards a more stable digital currency, R. Soc. Open Sci. 5 (7) (2018) 180155.
[34] L. Zhong, Q. Wu, J. Xie, J. Li, B. Qin, A secure versatile light payment system based on blockchain, Future Gen. Comput. Syst. 93 (2019) 327–337.
[36] V. Buterin, A next-generation smart contract and decentralized application platform, Whilepaper 3 (2014) 37.
[37] D. Schwartz, N. Youngs, A. Britto, in: The Ripple Protocol Consensus Algorithm, Ripple Labs Inc White Paper, 5, 2014, p. 8.
[38] K. Alam, K.M.R. Alam, O. Faruq, Y. Morimoto, A comparison between RSA and El-Gamal based untraceable blind signature schemes, in: Proceedings of the 2016 International Conference on Networking Systems and Security (NSysS), IEEE, 2016, pp. 1–4.
[39] C.P. Schnorr, Efficient identification and signatures for smart cards, in: Proceedings of the Crypto'89, Springer, New York, 1989, pp. 239–252.
[40] T. Granlund. "GNU Multiple Precision Arithmetic Library (GMP)". Software available at http://gmplib.org /July 2020.
[41] A. Menezes, P. van Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press, New York, NY, USA, 1997.
[42] J. Liu, R. Sun, W. Kou, Fair e-payment protocol based on simple partially blind signature scheme, Wuhan Univ. J. Nat. Sci. 12 (1) (2007) 181–184.
[43] Y. Seurin, On the exact security of Schnorr-type signatures in the random oracle model, in: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, 2012, pp. 554–571.
[44] G. Fuchsbauer, A. Plouviez, Y. Seurin, Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model, in: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2020, pp. 63–95. Springer.