An Immediate Real Time Detection and Prevention of Double-Spending in Electronic Cash Payment System

Princewill Aigbe  
Department of Mathematics and Computer Science  
College of Natural and Applied Sciences, Western Delta University, Oghara, Nigeria

Andrew Onibere  
Department of Computer Science  
Faculty of Physical Sciences.  
University of Benin, Benin, Nigeria

ABSTRACT

The introduction electronic commerce has created new financial needs that in many cases cannot be effectively fulfilled by the traditional payment systems. The success of electronic commerce business depends on the credibility of the available electronic payment systems. Electronic cash payment is one electronic payment systems developed to settle payments electronically, but suffers the problem of double-spending fraud. This paper identifies the possible causes of double-spending fraud and presents techniques to mitigate this type of fraud. These techniques involve the modification of existing electronic cash payment system, and a demonstration of the modified system to determine its double-spending fraud resistant capability.

General Terms

Electronic cash payment system, electronic commerce

Keywords

eCash, algorithm, registration certificate, double-spending, fraud resistant capability, merchant, real time, entity, public key infrastructure, subscriber registration authority.

1. INTRODUCTION

An electronic cash payment system allows users to withdraw eCash, represented as bit strings from a bank, and spend those eCash anonymously at participating merchants, so that the bank cannot link spent eCash to the user who withdraws them. A variety of schemes with various security properties have been proposed for this purpose, but because strings of bits are inherently copyable, they must all deal with the problem of double-spending. Double-spending is the failure of electronic cash payment systems allowing the possibility to spend the same or given eCash twice. Different electronic cash payment schemes have been proposed with various security features to prevent or fight against double-spending.

[1] proposed the first untraceable electronic cash system based on blind signature which allows the requester to obtain a message signature from a signer without revealing the message content and makes the signer unable to link any signed message to its signature. This initial proposal required an online broker to clear eCash before merchants could provide their services, to protect against double-spending. This solution was regarded as a cut and choose technology, and very expensive.

[2] improved on [1] by proposing an electronic cash payment scheme in which each eCash contains a hidden reference to the eCash owner. If the is spent once it is untraceable, while spending the eCash twice allows the broker to extract the identity hidden inside the eCash.

[3] introduced two new concepts in protecting against double-spending. The two concepts are:

i. Restrictive blind signatures to ensure traceability of double-spending after transactions. The realisation of this concept requires a great sacrifice in efficiency and questionable security.

ii. Wallets with observers guarantee prior restraint of double-spending, while still offering traceability of double-spenders after transactions in case tamper-resistant is compromised.

[4] proposed an electronic cash payment scheme in which every eCash generated is embedded with secret information. The secret information is used to verify any eCash that is double spent.

[5] proposed an electronic cash payment scheme with double-spending tracing based on elliptic curve cryptography. The scheme checks and traces double spending by requiring the merchant sends transcript of the execution of payment protocol to the bank, which verifies if that eCash has been double spent. In the payment protocol after the merchant sends challenge d to the user, the user calculates three responses (r1, r2, r3) and sends them to the merchant. If the user spent the same coin twice then the merchant should send to user another challenge d and user calculates another three responses (r1', r2', r3') to the merchant. In the deposit protocol the merchant sends (r1, r2, r3) and (r1', r2', r3') to the bank, and then the bank uses the responses to trace or reveal the identity of the user that double spent a given eCash.

[6] proposed electronic cash payment scheme that protects against double-spending by associating a pair of challenge c, and response r, keys to the transactions between a customer and merchant. The scheme uses the pair (c, r) to extract the identification information of any customer that double spent a given eCash.

[7] proposed a scheme that provides for users’ anonymity and detects double-spending by traceability. The proposed scheme attaches expiration date to eCash so that the banking system can manage its databases more efficiently.

[8] proposed an electronic payment scheme that allows the user to withdraw a single divisible eCash and spends the sub-eCash by dividing the value of the eCash. The scheme protects against double-spending by conducting a look up of the eCash’s serial number in a table of previously spent eCash.

The electronic cash payment system used different techniques to fight against double-spending. The different technique used by electronic cash payment system also shown that double-spending occurs before its detection and a trace of the
payment transaction is carried out to determine the double spender.

2. CAUSES OF DOUBLE-SPENDING
Double-spending fraud in electronic cash payment system occurred as a result of the following:

2.1 Concurrent executing transactions
The double spending fraud occurs when two or more concurrent executing transactions in a distributed environment read the old value of a variable (shared data) and then use it to calculate the new value.

Figure 1 shows a customer C with eCash token of 50 units performs two transactions with merchants M1 and M2 at the same time. In transaction T1, with merchant M1, customer C pays 40 units with eCashID:eCashValue(50) and retains 10 units while in transaction T1+1 with merchant M2, customer C pays 30 units with the same eCashID:eCashValue(50) and retains 20 units.

Figure 1: Two concurrent executing transactions with the same eCash:eCashValue(50)

The outcome of the two transactions shows that double-spending occurred as the accumulated units of eCashID for the two transactions is more than the available units of eCashID belonging to the customer. Both transactions have read the original units of eCashID available to the customer before either writes the new updated units of eCashID. This also shows that transaction T1+1 should have not been successful as the new available units of eCashID tokens after transaction T1 is insufficient to meet or satisfy the transaction, and this is a mutual exclusion problem.

2.2 Duplication of Electronic cash identification (eCashID)
Electronic cash identifications (eCashID's) are duplicated when they are transferred from one system to another without removing them from the original system. This normally occur when there is a full system backup and drive ghosting or image creation of a system. A duplicate electronic cash eCashID can cause double-spending when the electronic cash system implementation does not track or authenticate whether or not electronic cash eCashID have been spent by a customer or user before allowing a payment operation to be completed successfully.

Figure 2 shows duplicate copies of electronic cash identification (eCashID:eCashValue(60)) from customer C used to pay for goods or services from two merchants (M3 and M4) with different timestamps. This happens because the electronic cash payment system implementation has no mechanism of identifying duplicate copies of electronic cash identifications (eCashID)

Figure 2: Duplicate copies of eCashID:eCashValue(60) used for two different transactions

3. MECHANISMS FOR SOLVING THE DOUBLE-SPENDING FRAUD
The identified causes of double-spending fraud can be addressed by the application of distributed mutual exclusion algorithm and the introduction of a new entity to the existing electronic cash payment model.

3.1. Serialization of concurrent executing transactions
Distributed mutual exclusion algorithms based on message-passing will be implemented to coordinate the activities of concurrent executing transactions on a shared resource (shared data). Mutual exclusion is required to prevent interference and ensure consistency when accessing the shared resource. This means that the double spending fraud cannot happen if one transaction is performed before the other, because the later transaction will read updated value written by the earlier one, [9].

Since the serially equivalent interleaving of two transactions produces the same effect as a serial one, double spending fraud can be solved by means of serial equivalence. Each concurrent transaction employs the serialization technique in order to access and execute the shared data in critical section by distributed mutual exclusion mechanism [10].

For mutual exclusion (ME) to be enforced, the following requirements are essential:

i. At most one transaction may execute in the critical section, (ME1: safety).

ii. Requests to enter and exit the critical section eventually succeed, (ME2: liveness).
iii. If one request to enter the critical section happened-before another, then entry to the critical section is granted in that order, (ME3: ordering).

A study of the different algorithms that implement mutual exclusion was investigated to select the algorithm with the best performance characteristics. These algorithms are of two categories:

a. Non token-based Algorithms

These algorithms require two or more successive rounds of message exchanges among the contending transactions. These algorithms are assertion based because a transaction can enter its critical section (CS) when an assertion defined on its local variables becomes true.

Mutual exclusion is enforced because the assertion becomes true only for one transaction at any given time. These are: i. Lamport’s Algorithm
ii. Ricart and Agrawala’s Algorithm
iii. Maekawa’s voting Algorithm

b. Token-based Algorithms

A unique token (i.e., a privilege message) is shared among the transactions. A transaction is allowed to enter its critical section if it possesses the token and it continues to hold the token until the execution of the critical section is over. These are: i. Suzuki-Kasami
ii. Singhal’s Heuristic
iii. Raymond’s tree-based Algorithm

The performance of the various mutual exclusion algorithms is evaluated using the following criteria:

i. Number of Messages per Request

The total number of messages required to enter critical section (CS) is an important and useful parameter to determine the required network bandwidth for that particular algorithm.

ii. Response Time, (R)

The Response Time R is measured as the interval between the request of a transaction to enter a critical section (CS) and the time it finishes executing the critical section (CS). When the system is lightly loaded, twice the number of message transfer times and the execution time (E) of the critical section (CS) success results in light-load = 2T + E units. Under heavy load conditions, assuming at least one message is needed to transfer the access right from one transaction to another, then heavy-load = w(T + E) where w is the number of waiting requests.

iii. Synchronization Delay (S)

The synchronization delay S is the time required for a transaction to enter a CS once another transaction finishes executing it. The minimum value of S is one message transfer time T, since it requires one message success to transfer the access rights to another transaction.

[11] carried out the evaluation of the performance characteristics of the two categories of distributed mutual exclusion algorithms as shown in table 1.

<table>
<thead>
<tr>
<th>Non Token-Based Algorithms</th>
<th>Messages (Light-Load)</th>
<th>Messages (Heavy-Load)</th>
<th>Response time</th>
<th>Synchronization Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lamport</td>
<td>3(N-1)</td>
<td>3(N-1)</td>
<td>2T+E</td>
<td>T</td>
</tr>
<tr>
<td>Ricart-Agrawala</td>
<td>2(N-1)</td>
<td>2(N-1)</td>
<td>2T+E</td>
<td>T</td>
</tr>
<tr>
<td>Maekawa</td>
<td>3√N</td>
<td>5√N</td>
<td>2T+E</td>
<td>2T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Token Based Algorithms</th>
<th>Messages Light-Load</th>
<th>Messages Heavy-Load</th>
<th>Response time</th>
<th>Synchronization Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suzuki-Kasami</td>
<td>N</td>
<td>N</td>
<td>2T+E</td>
<td>T</td>
</tr>
<tr>
<td>Singhal’s Heuristic</td>
<td>N/2</td>
<td>N</td>
<td>2T+E</td>
<td>T</td>
</tr>
<tr>
<td>Raymond</td>
<td>Log(N)</td>
<td>4</td>
<td>T log(N) + E</td>
<td>T log(N)/2</td>
</tr>
</tbody>
</table>

Table 1 shows that token-based mutual exclusion algorithms have superior performance characteristics because they required fewer number of messages to transfer access right from transaction to another however, they have one serious drawback. Their resiliency to failure is poor because if a transaction having token fails or token is lost in transit, a complex process of token regeneration or recovery has to be started [12].

Also from table 1, the Maekawa’s voting algorithm has a superior performance in terms of the number of messages per entry and exit from the critical section but, with worse synchronization delay of 2T (i.e., 2-message time). The Maekawa’s voting algorithm is also deadlock prone. The Ricart-Agrawala (RA) algorithm is considered the most efficient non token-based mutual exclusion algorithms in distributed message-passing systems. The RA algorithm requires $2(N-1)$ messages per critical section access with a synchronization delay of one message time. T. The Ricart-Agrawala (RA) algorithm totally satisfies the essential requirements of mutual exclusion implementation and has been tested in different applications, [13]. The Ricart-Agrawala (RA) distributed mutual exclusion algorithm is therefore selected for the implementation of mutual exclusion in solving double spending fraud in the modified electronic payment system.

3.2 Control of duplicate eCash identification (eCash$_{id}$)

The various proposed electronic cash payment systems based on [11] model, checks for double-spending fraud after a transaction has been completed. If a transaction involves double-spending of eCash$_{id}$, a trace of the transaction is conducted to determine the identity of the double spender.

Therefore, to check double-spending occurrence, an immediate real-time detection and prevention technique is
required to successfully mitigate double-spending fraud in electronic cash payment system. This is done by introducing a new entity to the existing electronic cash payment system.

3.2.1 The new Entity in the modified system

The introduced new entity known as the subscriber registration authority (SRA) will be responsible for the following functions:

i. It generates and stores digital certificate containing key pair (private and public keys) for every participating entity.

ii. It mines or creates electronic cash tokens for the customers (subscribers) through a financial institution.

iii. It validates electronic cash tokens presented for payments.

iv. It invalidates electronic cash identifications (eCashID).

v. It also authenticates every participating entity in the payment process.

3.2.2. The other entities in the system are:

Merchant

The merchant is an entity that provides a wide range of online services (e.g. academic services) to customers or subscribers (e.g., students).

Customer (Subscriber)

This is an entity that requests different online services (e.g. academic services) from the merchants.

Financial Institution

This is an entity that provides (or renders) financial services for the entities involved in the online payment system, e.g. bank.

The modified electronic cash payment system is shown in figure 3:

![Figure 3: A block diagram of the modified electronic cash payment system](image-url)

where:

C₁ Customer request for digital certificate
C₂ Customer receives digital certificate
M₁ Merchant request for digital certificate
M₂ Merchant receives digital certificate
FI₁ Financial institution request for digital certificate
FI₂ Financial institution receives digital certificate

1. Customer sends digitally signed message to financial institution requesting for eCash.
2. Customer receives digitally signed message containing the requested eCash.
3. Customer initiates a transaction and receives payment request from the merchant.
4. Merchant generates and sends transaction notification message to subscriber registration authority
5. Subscriber registration authority receives and validates the transaction notification message to check for double-spending fraud.
6. Merchant receives transaction status message and generates a transaction acknowledgement report.
7. Customer receives transaction acknowledgement report and delivery of goods or services if successful.

4. TWO MAJOR FEATURES OF THE MODIFIED ELECTRONIC CASH PAYMENT SYSTEM

There are two major features of the modified system. These are:

4.1 Entity registration

Each participating entity registers with the subscriber registration authority (SRA) to obtain a subscriber registration certificate (SRC). A subscriber registration certificate binds a set of data items of a subscriber to a public key. It provides the means to know whether or not a public key does belong to a particular entity.

4.2. Mutual authentication of entities

For mutual authentication and secured messages to be exchanged among the entities in the modified system, public key infrastructure (PKI) is required. This is provided by an appropriate digital signature algorithm. A public key infrastructure (PKI) enables users of a basically unsecure public network such as the Internet to securely and privately exchange data and money through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority, [14]. The public key and private key pair is generated by digital signature schemes based on asymmetric design, and employ different mathematical techniques such as integer factorization problem, e.g. RSA; intractability of the discrete logarithm problem, e.g. ElGamal, DSA, Diffie-Hellman; and elliptic curve discrete logarithm problem, e.g. ECDSA, ECDH.

The application of a given digital signature scheme is selected based on some factors. The most viable of these factors is signature verification time, and RSA digital signature scheme has the best signature verification time, [15].
The generation or creation of digital signatures requires cryptographic hash functions. A cryptographic hash function is an algorithm that takes a group of characters (called a key) and maps it to a value of a certain length (called a hash value or hash, or message digest). The hash value or message digest is representative of the original string of characters, but is normally smaller than the original. One of the most secure hash functions used today is SHA-1 [16]. SHA-1 is a cryptographic hash function designed by the United States National Security Agency and is a U.S. Federal Information Processing Standard published by the United States NIST. SHA stands for "secure hash algorithm", SHA-1 produces a 160-bit (20-byte) hash value. A SHA-1 hash value is typically rendered as a hexadecimal number, 40 digits long. It is selected to produce message digest for RSA digital signature algorithm for the following reasons:

i. It is very simple and fast in generating a hash value or message digest.
ii. It has greater resistance to attacks.
iii. It is widely used security applications and protocols such as TLS, SSL, IPsec, etc.

5. APPLICATION OF THE MODIFIED ELECTRONIC CASH PAYMENT SYSTEM IN A TRANSACTION

The steps involved in the application of the modified electronic cash payment system in a payment transaction process are highlighted in sections 5.1 and 5.2. These steps are represented in algorithms.

5.1 Entity registration process

The first step in the application of the modified electronic cash payment system in a transaction is for an entity registration process with subscriber registration authority. This means that all participating entities (i.e., customer, merchant, and Bank) wishing to use eCash as a means of payment for goods and services online must register with subscriber registration authority to obtain a certificate. Algorithm 1 shows the algorithm for subscriber registration process.

Algorithm 1: Subscriber registration process
1. Collects(relevantdata); from the entity
2. Captures (relavantdata)
3. Generates (subscriberRegistrationCertificate) {to requested entity}
4. Sends (subscriberRegistrationCertificate) {to requested entity}

The subscriber registration certificate contains the following:

\[
\text{SRC} \rightarrow (\text{VRC, CSN, SAI, IN, PK}_{\text{SRA}}, V, \text{EN}, \text{PK}, \text{P}, \text{ACCOUNT}_{\text{ID}}, \text{ES})
\]

where

- VRC is the version of the subscriber registration certificate
- CSN is certificate serial number
- SAI signature algorithm identifier
- IN is issuer name
- PK_{SRA} is issuer public key
- V is the validity period of the certificate
- EN is entity name
- PK is entity public key
- P is entity private key

ACCOUNT_{ID} is entity account identification

ES is entity signature

When the financial institution received the signed message from the customer, it checks for message authenticity (i.e., if the message is sent by the claimed customer) as in the steps of algorithm 3.

Algorithm 3: Verification of eCash message
1. Computes \( h_{c} = (C_{DS})^{PK_{C}} \mod n \) \( \text{[PK}_{C}\text{ is customer’s public key]} \)
2. if \( h_{c} = \text{hashf(eCash}_{\text{RM}}) ; h_{c} \text{ is hash value of customer’s digital signature} \)
   i. accepts(eCash_{RM})
      ii. computes \( F_{DS} = (\text{hashf(eCash}_{\text{RM}}))^{P_{FI}} \mod n \) \( \text{[P}_{FI}\text{ is financial institution’s private]}
      iii. signs (eCash_{RM})
   iv. forwards(eCash_{RM}) \{to subscriber registration authority (SRA)\}
3. else
   message not authentic
   The message forwarded by the financial institution to the subscriber registration authority is of the form: signed eCashMessage \( \rightarrow (\text{eCash}_{\text{RM}}, \text{PK}_{FI}, F_{DS}, \text{Timestamp}), \) where

- \( \text{PK}_{FI} \text{ is financial institution’s public key} \)
- \( F_{DS} \text{ is financial institution’s public key} \)
- \( \text{Timestamp} \text{ is date and time of message generation} \)

Having obtained a registration certificate, the customer can now apply to a participating bank to request or purchase eCash by generating and sending a digitally signed message to the bank. Algorithm 2 shows the algorithm for eCash request.

Algorithm 2: eCash request message
1. requests (eCash_{RM}) \{eCash_{RM} is electronic cash request message\}
2. computes \( \text{CDS} = (\text{hashf(eCash}_{\text{RM}})PC \mod n \} \{PC \text{ is customer’s private key}\}
3. Signs (eCashRM)
4. Sends (eCashRM) \{ to financial institution \}

The signed electronic cash request message sent by customer contains the following items of data:

\[
\text{eCash}_{\text{RM}} \rightarrow (\text{Account}_{\text{ID}}, \text{eCash}_{\text{V}}, \text{PK}_{C}, C_{DS}, \text{Timestamp}),
\]

where

- Account_{ID} is customer’s account identification
- eCash_{V} is the value of electronic cash in units of a given currency
- PK_{C} is customer’s public key
- Timestamp is date and time of customer message
- C_{DS} is customer digital signature

36
When the subscriber registration authority received the message from the financial institution, it verifies and mines eCash as requested. The required steps are shown in algorithm 4.

Algorithm 4: Generates eCash
1. receives (eCash_{RM}) { from financial institution }
2. computes _h_ = (hashf(eCash_{RM}))^PKM \pmod{n} 
3. determines the existence of subscriber's accountID in the message 
4. if  _h_ = hashf(eCash_{RM}) and accountID exists 
   i. accepts (eCash_{RM}) 
   ii. generates (eCash) 
   iii. computes SRADS = (hashf(eCash)PSRA \times \pmod{n}) {PSRA is SRA private key} 
   iv. sends (eCash) { to financial institution } 
5. else 
   message is invalid and therefore, rejected

The eCash message generated by the subscriber registration authority is of the form:
eCashmessage \rightarrow (accountID, eCashID, eCashV, eCashSECRET, \text{PK}_{SRA}, \text{SRA}_{DS}, \text{Timestamp}), where 
accountID is customer’s account identification 
eCashID is electronic cash identification 
eCashV is electronic currency unit value 
eCashSECRET is eCash protected code 
\text{PK}_{SRA} is subscriber registration authority’s public key 
\text{SRA}_{DS} is subscriber registration authority’s digital signature 
Timestamp is date and time of eCash generation

On receiving the eCash message, the financial institution verifies the authenticity of the message from the subscriber registration authority as shown by the steps in algorithm 5.

Algorithm 5: eCash message verification 
1. computes _h_{SRA} = (\text{SRA}_{DS})^{PKSRA} \pmod{n} 
2. if _h_{SRA} = hashf(eCash) 
   i. eCash message is authentic 
   ii. deposits (eCash) {in customer’s account} 
   iii. sends (eCash notification) {to customer} 
3. else 
   eCash message is invalid
Finally, the customer receives the eCash requested.

5.2 Online Payment Procedure for goods and services
A customer visits a merchant store via the web to request goods or services as shown in algorithm 6.

Algorithm 6: Customer initiates a transaction
1. initiates (transaction) 
2. invoke serialisation(transactions) {using selected mutual exclusion algorithm} 
When the merchant receives the transaction, it generates a payment request for the customer as shown in algorithm 7.

Algorithm 7: Generation of payment request
1. Generates (paymentRequest) {with the necessary parameters} 
2. computes MDS = (hashf(paymentRequest))^PKM \pmod{n} 
3. signs (paymentRequest) 
4. sends (paymentRequest) {with digital signature (MDS)}
The digitally signed payment request sent by merchant to the customer is of the form:
paymentRequest \rightarrow (PR_ID, M_ID, T_A, T_D, PK_M, M_DS, Timestamp), where 
PR_ID is payment request identification 
M_ID is merchant identification 
T_A is transaction amount to be paid 
T_D is transaction description 
PK_M is merchant’s public key 
M_DS is merchant’s digital signature 
Timestamp is date and time of payment request generation

On receiving the payment request from the merchant, the customer performs message verification process as shown in algorithm 8.

Algorithm 8: Verification of payment request and generation of payment grant message
1. computes _h_ = (M_{DS})^{PK_SRA} \pmod{n} 
2. if _h_ = hashf(paymentRequest)) : payment request is authentic 
   i. accepts (paymentRequest) 
   ii. selects (eCashID) 
   iii. computes CDS = (hashf(paymentGrant)) {generates payment request} 
   iv. signs (paymentGrant) 
   v. sends (paymentGrant) {to merchant} 
3. else 
   payment request is invalid and not authentic
The payment grant message sent by the customer to the merchant consists of the following parameters:
paymentGrant \rightarrow (AccountID, eCashID, eCashSECRET, PK_C, C_DS, Timestamp), where 
AccountID is customer’s account identification 
eCashID is electronic cash identification 
eCashSECRET is eCash protected code 
PK_C is customer’s public key 
C_DS is customer’s digital signature 
Timestamp is date and time of payment grant message generation

The merchant receives payment grant message from customer and performs the actions in algorithm 9.

Algorithm 9: Verification of payment grant message and generation of transaction notification message
1. computes _h_ = (C_DS)^PK_C \pmod{n} 
2. if _h_ = hashf(paymentGrant)) 
   i. accepts (paymentGrant)
Algorithm 10: Verification of transaction notification message

1. computes \( h_m = (\text{hashf(transactionNotification)})^{\text{PKM}} \pmod{n} \) 
   \[ \text{generates transaction notification} \]
2. if \( h_m = (\text{hashf(transactionNotification)}) \)

2.1. accepts (transactionNotification)

2.2. if eCashID matches eCashSECRET

2.2.1. accepts (transactionNotification)

   i. generates(new secret)
   ii. reassigns (new secret) \{ to eCashID \}
   iii. changes(ownership of eCashID) \{ to MID \}
   iv. records (eCashID ownership exchange) \{ in transaction log \}

   v. generates (transactionStatus)
   vi. computes \( \text{SRA}_{DS} = (\text{hashf(transactionStatus)})^{\text{PSRA}} \pmod{n} \) \{ PSRA is subscriber registration authority’s private key \}
   vii. signs (transactionStatus)
   viii. sends (transactionStatus) \{ to merchant \}

2.2.2. if eCashID has changed

2.2.2.1. accepts (transactionNotification)

   i. eCashSECRET has changed
   ii. eCashID has been spent \{ immediate detection of double-spending attempt \}
   iii. terminates(transaction) \{ double-spending prevented \}
   iv. generates (transactionStatus)
   v. computes \( \text{SRA}_{DS} = (\text{hashf(transactionStatus)})^{\text{PSRA}} \pmod{n} \)

   vi. signs (transactionStatus)
   vii. sends (transactionStatus) \{ to merchant \}

3. else

   transaction notification message is invalid hence, not authentic

The transaction status message sent to the merchant consists of the following parameters:

\[ \text{transactionStatus} \rightarrow (T_{\text{STATUS}}, \text{AccountID}, \text{NeCashID}, \text{NeCashSECRET}, \text{PK}_{\text{SRA}}, \text{SRA}_{DS}, \text{Timestamp}) \]

where \( T_{\text{STATUS}} \) is transaction status

AccountID is customer’s account identification

NeCashID is new eCash identification

NeCashSECRET is new eCash protected code

PK_{SRA} is subscriber registration authority’s public key

SRA_{DS} is subscriber registration authority’s digital signature

Timestamp is date and time of transaction status message generation

When the merchant receives the transaction status message, it validates the message as shown in algorithm 11.

Algorithm 11: Verification of transaction status message and generation of transaction acknowledgement report

1. computes \( h_m = (\text{hashf(transactionAcknowledgement)})^{\text{PKM}} \pmod{n} \)

2. if \( h_m = (\text{hashf(transactionAcknowledgement)}) \)

2.1. accepts (transactionAcknowledgement)

2.2. if transactionStatus is successful

2.2.1. accepts (transactionStatus)

i. merchant deposits(eCashID)

\[ \text{with the new assigned eCashSECRET} \]

ii. generates (transactionAcknowledgement)

iii. computes MDS = (\text{hashf(transactionAcknowledgement)}) \pmod{n}

iv. signs (transactionAcknowledgement)

v. sends (transactionAcknowledgement)

2.3. else

   i. transactionStatus is unsuccessful
   ii. discards(eCashID)

   iii. computes MDS = (\text{hashf(transactionAcknowledgement)}) \pmod{n}

   iv. signs (transactionAcknowledgement)

   v. sends (transactionAcknowledgement) \{ to customer \}

3. else

   transaction status is invalid and not authentic

The customer receives the transaction acknowledgement message and carries out validation in algorithm 12.

Algorithm 12: Verification of transaction acknowledgement message

1. computes \( h_m = (\text{hashf(transactionAcknowledgement)})^{\text{PKM}} \pmod{n} \)

2. if \( h_m = (\text{hashf(transactionAcknowledgement)}) \)

2.1. accepts (transactionAcknowledgement)

2.2. if status is successful

i. removes (eCashID) \{ from wallet \}

ii. receives(goods or access contents) \{ sent by merchant \}

else
i. sends (transaction acknowledgement) 
    { to subscriber registration authority
     as a confirm acknowledgement request message}
3. else invalid message and hence, not authentic

6. CONCLUSION AND FUTURE WORK
This presents a modified electronic cash payment system to check double-spending fraud by the techniques of distributed mutual exclusion algorithm and introduction of a new entity that mines and validates electronic cash, and a demonstration of the modified system in a transaction process to show the detection and prevention of double-spending in real time.

The future work combines the different techniques discussed in this paper to carry out a real life implementation of the proposed modified electronic cash payment system.

7. REFERENCES