CHAPTER – V

PART II

Appendix

Design and Development of Threshold Protocols for Mobile-Commerce (M-Commerce) in Near Field Communication (NFC)
CHAPTER 5

DESIGN AND DEVELOPMENT OF THRESHOLD PROTOCOLS FOR MOBILE-COMMERCE (M-COMMERCE) IN NEAR FIELD COMMUNICATION (NFC)

This chapter presents the design of a family of fair exchange protocols based on NTRU cryptosystem for e-commerce transactions focusing on the electronic signing of contracts, certified message (SMS), e-payment and e-goods delivery. In these contracts, the parties (Customer/Trader/Bank) involved gain from their relationship. These gains could be e-payment, e-goods delivery, services or others. The contracts are particularly important in Customer-to-Business (C2B) (Peer-Peer) and Business-to-Business (B2B) (Group – Group)[39] commerce when establishing long-term relationships or offering services using Non-Server communication i.e. Near Field Communication.

Section 5.1 presents the Threshold Fair Integrated Data Exchange System (TFIDES) of our NTRU protocol using Non-Server. Section 5.3 presents the Contract Signing Phase of our NTRU Protocol. In section 5.4 Threshold certified e-mail delivery system, 5.5 thresholds certified e-goods delivery system, 5.6 generating e-payment system using NTRU, 5.7 Conclusion.

5.1 THRESHOLD FAIR INTEGRATED INFORMATION EXCHANGE SYSTEM (TFIIES)

This section presents the design, implementation and evaluation of a prototype mobile -commerce B2B system named TFIIES (Threshold Fair Integrated Information Exchange System). TFIIES is built around the family of fair exchange protocols presented in this thesis in order to demonstrate how business users can exchange valuable items across the Internet securely and fairly. Design and implement an end-to-end distributed, modular and portable service to secure fair contract signing and transaction.
The FIIES prototype enables execution of exchanging information, payment and certified delivery of e-goods between business parties, which can also be used as a part of procurement process. Secure message delivery has been considered in the prototype as it is more suitable for integration within communication process.

The design of the FIIES system is based on two main security requirements:

(1) Non-repudiation [26]: to provide protection against false denial of transmission or reception of communication in the NFC.

(2) Strong fairness: to ensure that either all parties participating in an exchange process receive the expected items or neither party receives anything valuable.

In addition, the following requirements have been identified in order to provide end-to-end security for business transactions:

(3) User authentication and access control: to ensure that the transfer and exchange services can only be accessed by authenticated and authorized users in Near Field Communication (NFC).

(4) Accountability: to allow secure logging and management of records of transactions undertaken within the system, so that all users are accountable for their actions. This requirement is necessary for potential dispute resolution.

(5) Data confidentiality: to prevent parties external to an exchange process from gaining any information of data being transferred in the NFC (Near Field Communication).

(6) Data integrity: to prevent parties external to an exchange process from unauthorized alteration of data being transferred.

In addition to these functional requirements, the FIIES architecture should be modular to enable re-using of its components and integration with other NFC and portable in order to work in heterogeneous business environments.
5.1.1. Our FIIES Architecture

FIIES is designed as a distributed system for securing services for contract signing and certified e-goods delivery between businesses. It consists of three main components:

• FIIES Server
• FIIES Client
• FIIES Code using NTRC

5.1.2. FIIES Architecture

Figure 5.0 FIIES architecture

In the proposed architecture, each organization individual runs and maintains its own FIIES Details like Server and FIIES Clients. Each System is the core of the system, and is responsible for serving requests from other components. It communicates internally to FIIES Clients used by business users from within the organization, and externally to other FIIES Servers belonging to business partners which is also using FIIES. FIIES is responsible for secure negotiation and execution of transactions, and for initiating transaction recovery with FIIES ZTTP Server in case of disputes. It also manages business user authentication and access control and enables the storage of transactional evidence for audit purposes and possible dispute resolution.
In NFC Business users use FIIES Clients to securely access the services provided by the other FIIES Server and to carry out m-commerce/e-commerce transactions. FIIES provide a graphical user interface that allows business users to view and manage items (i.e. contract documents, m-goods and m-payments) that are used in exchange transactions, as well as to initiate transactions with view and accept/reject transaction requests coming from external business partners.

FIIES ZTTP Server is run by a trusted third party authority for providing inter-organizational dispute resolution services. It is involved in recovery sub-protocol executions with FIIES Servers. There may be one or more ZTTPs in the FIIES system.

5.1.3. FIIES Client

Mobile users\Client has been implemented as an Android application rather than a JAVA application. The Client\Server uses Java socket programming to establish connection with TFIIES server.

In the diagram 5.1. Near Field Communication the mobile user can access any one of the access zone for his\her transaction where as in the old transaction he\she should swipe there card in the terminal. While Peer-Sender (Client) is used any one of the access zone in the near filed then he\she can’t access other access zone otherwise he\she should abort or complete the transaction in the access zone.

**Figure 5.1: Near Field Communication Mobile processing architecture**
5.1.4. Flat File No-Database Server

Database Sever implements in non-server application is very difficult, expensive and maintains database located on the mobile\PDA is very difficult. This component is responsible for secure and persistent storage of information regarding business users and transactions, including:

1. Information about business Credentials (users, their passwords, roles and privileges),
2. Business items that are exchanged during transactions. (e.g. contract documents, m-payments, m-goods, m-receipts, etc.),
3. Transactional records gathered during transaction execution,
4. Business partners that the organization does business with through the FIIES system.

5.2 CONTRACT SIGNING PHASE

In contract signing (CS) protocols, Zero-Trusted Message Party Service (ZTMPS) provide security and confidence [26,27] to the system. However, if the Trusted Message Party Service has to participate in a lot of transactions, a bottleneck can develop. For this reason, except in certain circumstances, it is desirable that the Trusted Third Party participates as little as possible in the execution of the protocol. Thus, it is preferable to have a Zero Knowledge TMPS instead of a TTP. In such scenarios, to obtain a valid contract it is mandatory that the contract is signed by a TTP, which acts as an e-notary. For example, Indian or Other laws establish that some contracts, such as those related to royalties or legacies, must be signed by an e-notary. Thus, the e-notary validates the contract and records it. Similar scenarios could be the agreement of rights of copyrighted works or a B2B contract update.

However, these proposals present some problems, such as not guaranteeing abuse freeness, or the over important participation of the ZTMPS in the protocol. The aim is that the protocol reduces the ZTMPS load to the minimum possible, i.e., that ZTMPS participate in very few messages in the protocol and makes the minimum number of cryptographic operations. The security during this process is also important.
and we should protect the information exchanged, maintaining its integrity, confidentiality and protecting the parties against attacks from parties not involved in the negotiation.

We propose a new protocol that is based on Zero Knowledge TMPS. However, in our protocol, unlike the best of the previous proposals based on a TTP, the ZTMPS does not participate in all the messages exchanged between the parties. In our proposal the ZTMPS only participates during the last phase of the protocol, in order to sign the agreement reached. Its main added value against other proposals is the incorporation of important features such as confidentiality and the secure negotiation of the contract. Furthermore, we have implementing the protocol so that it works with any public key cryptosystem that provides signature and ciphering (RSA, Jk-RSA) etc., but RSA, Jk-RSA is takes more space, storage and reduces efficiency in the portable devices. Thus, users could utilize the protocol with the keys and certificates that they own, because it is not necessary either to generate new keys or to engage in a registration process with a ZTMPS.

A paper-based contract is a signed document where two or more parties express an agreement. Contract signing protocols appeared to allow two or more parties to establish a contract over a network in a fair way. By fair it is meant that each honest party sending a signed contract is assured that if the other party obtains it, then he will also obtain the necessary signed contract of that party. In a contract signing protocol, the signing parties, at the very least, must participate. Additionally, other parties can participate, the role of the ZTMPS is to guarantee fairness and provide confidence to the system, because the signing parties might not be trustworthy.

In this kind of protocol there could be other parties (malicious entities) in the network that want to interfere in the signing of the contract by either modifying the content of the contract or by impersonating one of the parties or replaying old messages or by seeking to obtain confidential information about the contract. Basically, these attacks may pursue several goals, such as preventing a successful agreement being reached, obtaining confidential information, being detrimental to another party and obtaining the benefits of a contract by impersonating another party.
In this section we introduce the requirements that we need for electronic or mobile contract signing protocols in order to negotiate and sign contracts in a secure way and make use of a ZTMPS on-line for scenarios where the electronic signature of a ZTMPS is essential, for example, in some business-to-business or by legal requirement as was stated in the introduction. We therefore analyze related work.

The main disadvantage of an execution of the contract signing protocol, for each offer is that it supposes the exchange of a lot of messages, which makes it slow and not efficient. The main disadvantage of the use of a secure negotiation-specific protocol with a contract signing protocol is that in both processes we have to generate similar cryptographic material and use similar operations to obtain a secure communication. In both cases, as an additional drawback, we can mention that the negotiation is not linked to the contract. In fact, there are other fields of electronic business/commerce that have followed this approach of combining negotiation with another process. For example, in the field of electronic payments, the negotiation of the price and the payment of an electronic product are combined. Therefore, a contract signing protocol that provided all previous requirements would make the whole lifecycle of the negotiation and contract signing secure and efficient.

The goal of the protocol proposed NTRU is to guarantee non-repudiation [26,27,28,29] in the delivery of the messages, transaction, and it could be used for contract signing. In fact, it was proposed for collaborative e-commerce or m-commerce.

The protocol offers evidence as to whether one of the parties is not behaving correctly. It also offers confidentiality, and the ZTMPS will never know the content of the contract. However, there are two main drawbacks. The most important is that the protocol does not guarantee the abuse freeness property because User obtains a signed contract before Trader/Multi Level-Trader/Bank obtains it.

Another important problem is that the ZTMPS has a significant participation in the execution of the protocol. This could lead to the ZTMPS becoming a bottleneck if there were many concurrent executions of the protocol. Furthermore, if we want to negotiate the terms and conditions of the contract, it would be necessary to offer an additional mechanism. This is because if we used this protocol for each message in
the negotiation, then the complete process (negotiation plus contract signing) would present a high overload in the system, given the number of messages exchanged and the cryptographic operations made. Finally, one minor problem that could be solved easily is that the protocol does not satisfy the timeliness property.

We present a new variant threshold contract signing protocol which satisfies the requirements introduced in the previous section to provide a robust contract signing protocol. Our novel protocol is based on ZTMPS for those situations in which an e-notary\m-notary has to authenticate a contract by signing it. Our protocol reduces the participation of the ZTMPS to the minimum possible expression as regards the number of cryptographic operations and messages sent and received. It is also important to mention that it could be very expensive while using the existing public cryptosystem like RSA, J_{k}-RSA instead of that using NTRU it’s very accurate, affective this offers signature and ciphering.

![Image](image_url)

**Figure 5.2:** Mobile processing architecture in the NFC with different Devices
5.3. DEFINITION OF A THRESHOLD CONTRACT IN NTRU SYSTEM

For this protocol, a contract has the following structure:

\{\text{UID}, \text{MS}, \text{MR}, \text{TS, Nonce, H(CD), H(SC-MR), H(SC-MS)}\} \text{ ZTMPS, SC-MR, SC-MS}

The contract is basically a document signed by the e-notary (ZTMPS) and it contains the following information:

- **UID (Unique Identifier):**
  
  This is a unique identifier of the contract. The \text{UID} identifies the transaction performed between Trader, Bank and Customer. Although this identifier may globally unique, it is used to distinguish between the different negotiations or transactions performed by the same parties. Thus, the UID is used to identify the transaction to which a message belongs.

- **MS, MR:**

  These are the identifiers of the parties (Sender(Customer) and Receiver (Trader)) between whom the contract is signed. This type of identifier is the digest of the party’s public key.

- **CD (Contract Document):**

  A document that reflects the agreed contract terms between different parties such as Trader, Bank and Customer

- **TS (Time Stamp):**

  Time Stamp it is used to distinguish between the different negotiations or transactions performed by the same parties. Thus, the TS are used to identify the transaction to which time message belongs.

- **H(CD):**

  This is the hash of the document which represents the contract terms.
• $H(SC-M_C), H(SC-M_R)$:

These represent the hashes of the contract signed by Receiver such as Trader, Bank and Sender such as Customer, respectively. We can advance that it is basically the electronic/mobile signature of the contract with other information necessary to guarantee the non-repudiation and fairness of the protocol.

By means of this contract, the ZTMPS testifies that Receiver such as Trader and Sender such as Customer have reached an agreement or contract (identified by the UID). This is reflected in a document that contains some information, such as the hash of the contract document ($H(CD)$), the hashes of the signatures made by Receiver and Sender, and the corresponding signatures. It also contains information about when the contract is signed by the ZTMPS (Timestamp). To sum up, our contract is an electronic signature made by the ZTMPS that links the contract terms signed individually by each party. Thus, the contract contains the signatures of the ZTTP and the participating parties, as supposed.

5.3.1 General Mode of Threshold Contract Signing Protocol using NTRU

In the General Mode, the protocol (a signed contract document, m-goods or swipe technology or SMS (Short Message Service or Multimedia Message Service) is composed of the messages that appear in below diagram.
STEP I.

**Sender => Receiver: Contract Signing Request**

```
{{UID, TS, SeqN, [Cred], Customer, EnKey, SignKey, Flag}C^{-1}, CD,H(CD)}
```

Receiver

Where: *UID (Unique Identifier)*:

The *UID* identifies the negotiation being performed between Sender and Receiver. Although this identifier might not be globally unique, it is used to distinguish between the different negotiations or transactions performed by the same parties. Thus, the UID is used to identify the transaction to which a message belongs. This identifier or label could be a randomly generated number.

These design principles recommend that the label has the following properties: verifiability, uniqueness and secrecy. Therefore, we have generated this identifier as:

\[ H(S,R,ZTMS,H(EnKey),H(SignKey)) \].

*Time1*. This is the time until which Sender will wait for a response from Receiver. In general, we represent with *TimeX* the time one party will wait for a response from the other one. This time should include a date and hour. It will delimit the time of a possible response. In this case, if a response does not arrive before *Time1*, Sender will consider that Receiver is not interested in continuing with the negotiation. This may be because Receiver is not interested in her conditions. Thus, the negotiation is considered finished without an agreement. Therefore, there will be no contract signing process and no more messages are exchanged.

*SeqN (Sequence Number)*. During the negotiation phase, it is possible to exchange several messages. Every message in this sequence must be unique in order to prevent reply attacks. For this reason, a *SeqN* field is present in the negotiation messages and each party must increment the *SeqN* value after receiving this type of messages.

*Cred (Credentials)*. This is an optional field which can be used to provide the user’s credentials. For example, these could be a User name / PIN/ Pattern or Password etc.
Receiver’s identifier. This field is the digest of Receiver’s public key. It identifies the intended receiver of this negotiation request.

EnKey (Encryption Key). This is an Asymmetric key generated by Alice that is used to provide confidentiality to the following messages exchanged between Sender and Receiver.

SignKey (Signing Key). This is an Asymmetric key generated by Sender. It is used to provide integrity to the subsequent messages exchanged between Sender and Receiver. The default cryptographic checksum function to Mobile MAC.

The Contract Signing Request message indicates the beginning of the execution of the protocol in the Near Field Communication (NFC). The message is used by one of the parties when it decides to initiate the negotiation of the conditions of a contract in order to reach an agreement that will be reflected in a signed contract. The contract terms appear in the proposed contract document in the field called ContractDoc. In this step, the contract documents (ContractDoc) and $H(ContractDoc)$ are not signed by Sender to avoid the abuse freeness, since if Sender signed these data, then Receiver would be able to present them to another party, so gaining an advantage over Sender.

STEP II. Receiver => Sender : ResponseStep

\{UID, Time2, SeqN, [Cred], ContractDoc, Flag|SignKey,EnKey\}

Receiver sends this response message to make a new offer (counter-offer) if he does not agree with the contract terms. If Receiver does agree to the conditions, he sends a Handshake message instead. Thus, the Handshake indicates the end of the response and the initiation of the contract signing process. The ResponseStep message is used by Sender and Receiver on various occasions until one of the following conditions is reached:

- One of them accepts the conditions of the other party in the last Response Step message. In this case, a Handshake message is sent.
- One of them receives a ResponseStep message containing a last offer (flag is activated) that it does not agree with. In this case, the communication is closed.
STEP III. Sender => Receiver: Handshake

|RecKey|EnKey, SC

Where:

- \( SC = \{UID, Time3, C, TTP, \{H(ContractDoc)\}\} \) represents the contract signing by Sender

- \( \text{RecKey (Receipt Key). This is a symmetric key which will be used to receive the contract signed by the ZTMPS. Its value is the result of performing the hash on the } \text{Nonce}\).

- \( Time3\). This indicates the deadline until which Sender considers the agreement is possible with Receiver. This time must be taken into account by Receiver and ZTMPS. Even if Receiver signed his part of the contract, if it is signed after time \( Time3\), the ZTMPS will not sign it and there will not be a valid contract.

This is the step where Sender decides to accept the contract by sending it to Receiver. Thus, this message indicates the end of the negotiation process and the initiation of the contract signing process. However, in order to avoid Receiver gaining an advantage over Sender, the signed contract is ciphered with the ZTMPS’s public key. The hash of the symmetric key, \( \text{RecKey} \), is also sent signed, so Sender and the ZTMPS could be sure the symmetric key was not changed by another party, because this key will be used to send the signed contract in a ciphered form.

STEP IV. Receiver => ZTMPS: Threshold Agreement

There are two possibilities in this step:

a) \( SC, SC_T \) to ZTMPS or

b) \( SC, SC_T \) to ZTMPS, |ContractDoc|RecKey

Where: \( SC_T = \{UID, Time4, ZTMPS, H(ContractDoc)\}^{T-1} \)

This represents the signing of the contract by Receiver.
• **Time4.** This indicates the time until which Receiver will wait for the contract to be signed. This time should not be greater than Time3. This is because the ZTMPS also has to take into account that Sender, in Time3, indicated that she wants the contract to be signed before this time. If Receiver put a greater time, it would be useless because the ZTMPS would have to sign the contract before Time3 indicated by Sender. If Receiver agrees with the time specified by Sender he should put the same time. Time4 has been introduced in case Receiver wants to further limit the deadline for signing the contract by ZTMPS. This message represents the agreement between the two parties for a contract. It contains part of the information received in the Handshake message (step III), i.e. the signing of the contract by Alice. Furthermore, it includes signed information by Receiver that is sent ciphered to the ZTMPS with its public key. This information reflects Receiver’s conformity with the contract. It is sent ciphered to the ZTMPS in order to avoid Sender intercepting the message, which she could show to another party to gain an advantage over Receiver. There are two versions of this message. In the first, Sender and Receiver decide that the ZTMPS does not need to know the content of the e-contract. But, in the second, they deem it necessary that the ZTMPS knows the e-contract, e.g. in those scenarios where it is required that the ZTMPS certifies, records and saves the contract. In order to avoid outside entities knowing the content of the contract, it is sent ciphered with the symmetric key (RecKey). When the ZTMPS receives this message, then the ZTMPS will check that the signatures are valid and that the contract document signed by each party is the same using NTRU systems. In this case, the protocol continues in the following step, otherwise it finishes.

**STEP V. ZTMPS=>Sender, Receiver: Threshold SignedContract**

|{UID,C,T,Timestamp,Nonce,H(ContractDoc),H(SCC),H(SC_r)}ZTMPS-1, SC_r|RecKey

Where Timestamp indicates the date and the time when the ZTMPS signed the contract, once Sender and Receiver had reached an agreement. This time should be less than the minimum between Time3 and Time4. That is, the ZTMPS would choose the most restrictive time. As commented in the previous message, there are two possibilities: either both Time3 and Time4 are equal or Time4 is less than Time3. The
message represents the approval of the signed contract between Sender and Receiver. It is signed by the ZTMPS in order to prove that both entities agreed to the contract reached. Receiver’s signature is included so that Sender can have a copy of the signed contract by Receiver. Thus, in case of dispute, the ZTMPS will not be necessary.

5.3.2 Contract Signing between Senders, Receiver in the Near Field Communication

![Diagram: Threshold Contract Signing between Senders, Receiver](image)

Fig: 5.4. Threshold Contract Signing between Senders, Receiver
MM1: Sender (Customer) communicates the Receiver (Trader) in the NFC location.
Sender → M_C, M_R

MM2: Receiver accepts the request of the Sender in the NFC and send the response to Customer (Sender)
Trader → M_R, M_C, TS, ACK

MM3: Customer Sends the Message (Items or Message) to the trader in the NFC
Customer → Customer: M_R, M_C, TS, MS

ME1: Customer request for connection establishment with trader
Customer → Trader: 

ME2: Trader response for connection establishment with Customer

ME3: Customer request for Contract Signing
Customer → Customer: =\{UID, C_{EnKey}, CS\}

ME4: Trader response the Contract Signing
Trader → Customer: =\{UID, Time2, SeqN, [Cred], Flag|SignKey,C_{EnKey}\}

ME5: Trader process the Customer information Credentials with ZTMPS (Verifying Customer using Virtual Card information of the Customer \ Sender)
Trader → ZTMPS: check the credentials

ME6 = DT \{ ME5 \}, H(CD)Ec

Status ? Ok : resolve

Trader verifies the correctness of CS and E_T(CS,M). Trader also verifies Customer’s signature H(CD)Dc by decrypting it with Customer’s public key to obtain a hash value M’ and comparing it with the hash value H_{Dc} contained in certificate CS.
If the two hash values do not match, Trader may ask Customer to re-send message (ME5) or terminate the protocol execution.

Otherwise, Trader generates the VRES for his signature

ME9: Trader searches the items and availability etc of E-Goods or M-Goods selected by the customer

Trader →Customer: CS, Cost

**Status? Ok : resolve**

$$ME_9 = \{ D_T(UID, I_c, I_T, T_s, flag, ET(CS,M), H(CD)D_T) \}$$

Trader verifies the correctness of CS and E_c(EG,M). CS. If Trader may ask Customer to re-send message (ME5) or terminate the protocol execution.

Otherwise, Trader generates the VRES for his signature

ME7: ZTMPS sends the Contract Signing the Customer through the Trader (Receiver)

$$ZTMPS \rightarrow Trader: CS, Cost, PaymentRequest process from the Trader side$$

$$ME_7 = \{ E_T(UID, I_c, I_T, I_z, T_s, flag, ET(CS,M), H(CD)D_c), CS \}$$

ME8: Customer process the ZTMPS Contract Signing (Decrypteds the message ME7)

Customer →Customer: Ic, I_T, T, CS, CD, SigC, CertC, CertT

$$ME_8 = D_T(ME_7)$$

Status?OK: Resolve

Trader verifies the correctness of CS and E_c(EG,M). Trader also verifies Customer’s signature H(CD)Dc by decrypting it with Customer’s public key to obtain a hash value M’ and comparing it with the hash value H_{Dc} contained in certificate CS. If the two hash values do not match, Trader may ask Customer to re-send message (ME5) or terminate the protocol execution. If matches then trader request the bank to process the payment

Otherwise, Trader generates the VRES for his signature

ME9: Customer sends the ack to from the Trader

Customer →Trader : Ic, I_T, T, CS, SigC, CertC

ME10: Trader sends the ack to from the Sender/Customer
**5.3.2.1 Threshold Contract Signing Abort Sub Protocol**

In this section we present the abort sub protocol. We have defined it taking into account that it is possible that in some circumstances, a party that has signed a contract wants to cancel it prior to its being signed by the ZTMPS. Thus, a party that has initiated the contract signing process can abort it fairly. As commented previously, the signing process is initiated with the *Handshake* message, it continues with the *Agreement* message and it finishes with the *SignedContract* message. Therefore, if a party wants to abort this signing process once initiated, he/she has to execute the abort protocol before the *SignedContract* message is generated by the ZTMPS. For this abort protocol, the messages that we have needed to define are:

**STEP I. Customer => ZTMPS: Abort**

\{UID, T_{f}, ZTMPS, \{Nonce, H(ContractDoc)\}ZTMPS\}Ic-1

This message is used by Customer in order to inform the ZTMPS that she wants to abort the protocol. If the contract has not yet been signed, step II of the abort protocol is executed. Otherwise, the last message of the execution of the protocol is re-sent (*SignedContract*).

**STEP II. ZTMPS =>I_{C},I_{T}: ConfirmedAbort**

\{UID,Timestamp, I_{C},I_{T}, Abort\}TTP-1

This message is received as confirmation that the protocol was aborted. This is indicated by the flag *Abort*. 

---

\[ ME_{10} = \{ E_{C}(UID, I_{C}, I_{T}, I_{Z}, T_{10}, flag, ET(CS,M),H(CD)Dc), CS\} \]

Status ? OK: Resolve

*Customer* verifies the correctness of *CS* and \( E_{T}(EG,M) \). *Customer* also verifies Trader’s signature \( H(CD)D_{T} \) by decrypting it with Trader’s public key to obtain a hash value \( M' \) and comparing it with the hash value \( H_{DT} \) contained in certificate *CS*.

\( ME_{11} : \) ZTMPS sends the ack to Trader about customer CS

\[ ZTMPS : \text{Sends ack of Customer} \]
5.3.2.2 Threshold Resolve Protocol

It could occur that when one party, after having sent its contract signature to the other one or to the ZTMPS, after the deadline, does not receive any response in access zone. We will comment on the sequences of messages to exchange supposing that it was Customer who did not receive the response (answer) in the NFC.

**STEP I. Customer => ZTMPS: Threshold Resolve**

\[\{\text{UID, TimeX, I_T, ZTMPS, Nonce, } H(\text{ContractDoc}),\ RecKey\}\text{ZTMPS} \]

Where `TimeX` is the time indicated in the *Handshake* or *Agreement* message. The time, when this message is sent, should be later than `TimeX`. Depending on the messages sent (or not) to the ZTMPS, in the execution protocol, Customer could receive one of the following messages:
a) STEP II. ZTMPS => Customer: Threshold Signed Contract

This message is the same as we commented in step V of the protocol and it would be received if the contract was signed and sent to the ZTMPS by Trader.

b) STEP II. ZTMPS => Customer: Threshold ConfirmedAbort

\{UID,TimeStamp,IC,IT,Abort\} ZTMPS-1

This message would be received if Trader cancelled the signing of the contract.

c) STEP II. ZTMPS => Alice: NoContract

\{UID, IC, IT, Timestamp\} ZTMPS-1

If after the TimeX, the ZTMPS has not received an abort or a signed contract, ZTMPS sends a signed message to Customer indicating that there was no agreement.

---

**Fig. 5.6 Contract Signing Resolve Protocol**
5.3.3 Threshold Contract Signing between Peer (Client) and Group

Our novel approaches to NTRU CS, In Group(Bank\ Multilevel Trader) we have Manager, Clerk both should have keys for this we can use ring NTRU. One should send the message from Manager Signer and Clerk co-signer both should be authorize then only signature can be validated otherwise signature is not validate.

Threshold Contract Signing between Senders, Bank (Group)

Fig. 5.7 Threshold Contract Signing between Sender, Bank
MM1: Sender (Customer) communicates the Group in the NFC location sending the broadcast messaging.

Sender → M_C, M_G

MM2: Receiver accepts the request of the Sender in the NFC and send the response to Customer (Sender)

Group → M_G, M_C, TS, ACK

MM3: Customer Sends the Message (Items or Message) to the trader in the NFC

Customer → Customer: M_G, M_C, TS, MS

ME1: Customer request for connection establishment with Group

Customer → Group: (\{UID, TS, SeqN, [Cred], I_C, T_EaKey, C_SignKey, Flag\}_A^1, CD,H(CD))

ME2: Group Owner response for connection establishment with Customer

Group → Group: (\{UID, TS, SeqN, [Cred], I_C, T_EaKey, C_SignKey, Flag\}_A^1, CD,H(CD))

ME3: Group Owner request for the Group Member to response for Customer

Group Owner → Group O-Group M: (\{UID, TS, SeqN, [Cred], I_C , T_EaKey, C_SignKey, Flag\}_A^1, CD,H(CD))

ME4: Group response for connection establishment with Customer

Group → Customer: ={UID, Time2, SeqN, [Cred], Flag|SignKey,C_EaKey}

ME5: Customer request for Contract Signing

Customer → Customer: ={UID, C_EaKey, CS}

ME6: Group response the Contract Signing

Group → Customer: ={UID, Time2, SeqN, [Cred], Flag|SignKey,C_EaKey}

ME7: Group process the Customer information Credentials with ZTMPS (Verifying Customer using Virtual Card information of the Customer \ Sender)

Trader → ZTMPS: check the credentials

Trader → ZTMPS: Customer: \{ D\_T(UID,I_C,I_T,T_5,ET(CS,M),H(CD)Dc,)}

or ME_6 = D\_T \{ ME_5\}, H(CD)Ec

Status ? Ok : resolve
Group verifies the correctness of CS and \(E_2(CS,M)\). Group also verifies \(Customer\)’s signature \(H(CD)Dc\) by decrypting it with \(Customer\)’s public key to obtain a hash value \(M'\) and comparing it with the hash value \(H_{Dc}\) contained in certificate \(CS\). If the two hash values do not match, \(Trader\) may ask \(Customer\) to re-send message (ME5) or terminate the protocol execution.

Otherwise, Group generates the VRES for his signature

ME8: Group searches the items and availability etc of E-Goods or M-Goods selected by the customer

\[Trader \rightarrow Customer: CS, Cost\]

\textbf{Status ? Ok : resolve}

\[ME9 = \{ D_T(UID, I_c, I_T, I_z, flag, ET(CS,M), H(CD)D_T)\}\]

Group verifies the correctness of CS and \(E_c(EG,M)\). If Group may ask \(Customer\) to re-send message (ME5) or terminate the protocol execution.

Otherwise, Group generates the VRES for his signature

ME 10: ZTMPS sends the Contract Signing the Customer through the Group (Receiver)

\[ZTMPS \rightarrow Group: CS, Cost, PaymentRequest process from the Trader side\]

\[ME_{10} = \{ E_T(UID, I_c, I_T, I_z, T_s, flag, ET(CS,M), H(CD)D_c), CS\}\]

ME11: Customer process the ZTMPS Contract Signing (Decrypteds the message ME10)

\[Customer \rightarrow Customer: I_c, I_T, T, CS, CD, SigC, CertC, CertT\]

\[ME_{11} = D_T(ME_{10})\]

\textbf{Status?OK: Resolve}

Group verifies the correctness of CS and \(E_c(EG,M)\). Group also verifies \(Customer\)’s signature \(H(CD)Dc\) by decrypting it with \(Customer\)’s public key to obtain a hash value \(M'\) and comparing it with the hash value \(H_{Dc}\) contained in certificate \(CS\). If the two hash values do not match, Group may ask \(Customer\) to re-send message...
(ME5) or terminate the protocol execution, if matches then Group request the bank to process the payment if required for the business transaction.

Otherwise, Group generates the VRES for his signature

ME11: Customer sends the ack to from the Group

\[ \text{Customer} \rightarrow \text{Group: } Ic, I_T, CS, \text{SigC, CertC} \]

ME12: Group sends the ack to from the Sender:Customer

\[ ME_{12} = \{ E_c (\text{UID, } I_c, I_T, I_z, T_{10}, \text{flag, ET}(CS, M), H(CD)D_c), CS \} \]

\text{Status ? OK: Resolve}

Customer verifies the correctness of CS and \( E_t(EG, M) \). Customer also verifies Group's signature \( H(CD)D_t \) by decrypting it with Group's public key to obtain a hash value \( M' \) and comparing it with the hash value \( H_{DP} \) contained in certificate CS.

ME13: ZTMPS sends the ack to Group about customer CS

\[ \text{ZTMPS : Sends ack of Customer} \]

Otherwise, Banker-clerk generates the VRES for his signature

5.3.3.1 Threshold Resolve Protocol

It could occur that when one party, after having sent its contract signature to the other one or to the ZTTP, after the deadline, does not receive any answer. We will comment on the sequences of messages to exchange supposing that it was Client/Banker who did not receive the answer.

i. Client , ZTTP Threshold Resolve Protocol

STEP I. Client =>ZTTP: Threshold Resolve

\[ \{ \{ \text{UID}, \text{TimeX}, I_T, \text{ZTTP, Nonce, H(ContractDoc), RecKey} \} \}_{Ic-1} \} \text{ZTTP} \]

Where TimeX is the time indicated in the Handshake or Agreement message. The time, when this message is sent, should be later than TimeX. Depending on the messages sent (or not) to the ZTTP, in the execution protocol, Client could receive one of the following messages:
a) **STEP II. ZTTP => Client: Threshold Signed Contract**

This message is the same as we commented in step V of the protocol and it would be received if the contract was signed and sent to the ZTTP by Trader.

b) **STEP II. ZTTP => Client: Threshold ConfirmedAbort**

\{UID,TimeStamp,I_c,I_T,Abort\}ZTTP-1

This message would be received if Trader cancelled the signing of the contract.

c) **STEP II. ZTTP => Client: Threshold NoContract**

\{UID, I_c,I_T,TimeStamp\}ZTTP-1

If after the TimeX, the ZTTP has not received an abort or a signed contract, ZTTP sends a signed message to Client indicating that there was no agreement.

**ii. Banker, ZTTP Threshold Resolve Protocol**

**STEP I. Banker =>ZTTP: Threshold Resolve**

\{{{UID,TimeX,I_Z,ZTTP,Nonce,H(ContractDoc), RecKey}_B^{-1}}\}ZTTP

Where TimeX is the time indicated in the Handshake or Agreement message. The time, when this message is sent, should be later than TimeX. Depending on the messages sent (or not) to the ZTTP, in the execution protocol, Banker could receive one of the following messages:

a) **STEP II. ZTTP => Banker: Threshold SignedContract**

This message is the same as we commented in step V of the protocol and it would be received if the contract was signed and sent to the ZTTP by Banker.

b) **STEP II. ZTTP => Banker: Threshold ConfirmedAbort**

\{UID,TimeStamp,I_Z,I_c,I_T,Abort\}ZTTP-1

This message would be received if Trader cancelled the signing of the contract.
c) **STEP II. ZTTP =>Trader : NoContract**

\[\{\text{UID, I_z, I_B, Timestamp}\}\text{ZTTP-1}\]

If after the TimeX, the ZTTP has not received an abort or a signed contract, ZTTP sends a signed message to Client indicating that there was no agreement.

### 5.3.3.2 Bank, ZTTP Threshold Resolve Protocol

![ZTTP Threshold Resolve Protocol Diagram](image)

**Fig. 5.8 Group (Bank), ZTTP Threshold Resolve Protocol**

### 5.4 THRESHOLD CERTIFIED CHAT\SMS\MMS\E-MAIL DELIVERY SYSTEM

With the growing on Chat\SMS\MMS\e-mail as the main business communication tool, there is an increasing demand for a reliable Chat\SMS\MMS\e-mail service that has embedded robust security services in the portable devices in the Near Field Communication in the area. The basic Chat\SMS\MMS\e-mail security services include the provision of privacy (only the intended recipient can read the message) and authentication (the recipient can be assured of the identity of the sender)
without using Server i.e., Non-Server. Cryptographic mechanisms for providing these security services, i.e. symmetric encryption and digital signatures/certificates, have been applied in security system.

In addition to sender authentication and message privacy can also provide a signed receipt service. A signed receipt from the recipient (requested by the sender) serves as a non-reputable proof of receipt for a specific Chat\SMS\MMS\e-mail. However, the return of this receipt relies on the willingness of the recipient to honor the sender’s request and provides no protection to the sender if the recipient chooses not to sign and return the acknowledgement after having read the message. In other words, this technique does not truly provide non-repudiation of the receipt security service. For a viable certified Chat\SMS\MMS\e-mail service, the following security properties are needed:

**Non-repudiation of origin** - the recipient must have a way of proving that a specific Chat\SMS\MMS\e-mail indeed originates from the sender.

**Non-repudiation of receipt** - the sender must have a way of proving that the recipient has indeed received a specific Chat\SMS\MMS\e-mail.

**Strong fairness for the exchange of Chat\SMS\MMS\e-mail for a receipt** - the recipient should obtain a specific e-mail if and only if the sender obtains a receipt for it.

Digital signature mechanism provides the basis for the provision of non-repudiation security service. A digital signature on a message establishes the authenticity of the message and the identity of its originator. Therefore, the sender’s signature on a message serves as a non-reputable proof of its origin, and a proof of receipt is represented with the recipient’s signature on the received message. A certified e-mail protocol must prevent a misbehaving recipient from refusing to return his receipt, i.e. from selectively acknowledging a message, after seeing its content. As the receipt depends on the message, the recipient cannot simultaneously receive the Chat\SMS\MMS\e-mail and sign the receipt. Therefore, adequate security protocols are required to simulate this pseudo-simultaneous exchange over a serial communication network and ensure that neither party suffers a disadvantage in the process.
The new efficient NTRU - CEMD protocol for the two communicating parties to fairly exchange Chat\SMS\MMS\e-mail message for using NTRU-based in Near Field Communication. The NTRU-CEMD protocol is based on a cryptographic primitive called verifiable and recoverable encrypted signature (VRES). VRES enables the recipient of Chat\SMS\MMS\e-mail to encrypt his receipt (i.e. signature) in such a way that the sender can verify the correctness of the receipt without accessing its content, and, at the same time, the sender is convinced that an agreed Zero knowledge trusted Message Party Service (ZTMPS) can recover the receipt from its encryption, should the recipient refuse to do so. In this way the services of the ZTMPS are invoked only in case of dispute.

The main advantage of our novel NTRU protocol for VRES allows fair certified delivery in a more efficient manner in comparison with other related VRES-based certified Chat\SMS\MMS\e-mail protocols, in terms of number and length of messages required and number of expensive cryptographic operations used in formation of the messages. Protocols employ interactive zero-knowledge (ZK) proofs for VRES verification, which can be computationally consuming and generate an excessive communication overheads, whereas VRES verification in our protocol is performed efficiently without any on-line interactions between the sender (Trader/Bank) and the recipient (Sender).

The following assumptions are used in the design of the NTRU-CEMD protocol. Party Receiver (Trader/Bank) wishes to send Chat\SMS\MMS\e-mail message $M$ to party Sender in exchange for Sender’s receipt for Trader/Bank and Customer have agreed to employ an ZTMPS(Iz) to help them with the exchange if they cannot reach a fair completion of the exchange themselves.

The basic protocol contains the following steps. Before a Receiver such as Trader/Bank can send a chat\SMS\MMS\e-mail message to a Customer, the Trader/Bank must submit a sending request to ZTMPS to obtain a ticket. The ticket contains necessary information for the Sender in order to read the message. Then, the Receiver such as Trader/Bank sends both ticket and encrypted message to the Sender. In order to read the encrypted message, the Sender requests the key from ZTMPS. After ZTMPS issues the key to the Sender, ZTMPS sends out the receipt of the message to the Receiver (Trader/Bank).
5.4.1 Threshold Certified message delivery between Senders, Trader Protocol in NFC

MM1: Sender (Customer) communicates the Receiver (Trader) in the NFC location.
Sender → MC, MR

Fig.5.9 Threshold Certified message delivery between Sender, Trader Protocol
MM2: Receiver accepts the request of the Sender in the NFC and send the response to Customer (Sender)

\[
Trader \rightarrow M_R, M_C, TS, ACK
\]

MM3: Customer Sends the Message (Items or Message) to the trader in the NFC

\[
Customer \rightarrow Customer: M_R, M_C, TS, MS
\]

ME1: Customer request for connection establishment with trader

\[
Customer \rightarrow Trader: ([{UID, TS, SeqN, [Cred], I_c, T_{EnKey}, C_{SignKey}, Flag}] A^{-1}, CD, H(CD))
\]

ME2: Trader response for connection establishment with Customer

\[
Trader \rightarrow Customer: ={UID, Time2, SeqN, [Cred], Flag|SignKey, C_{EnKey}}
\]

ME3: Customer request for Contract Signing

\[
Customer \rightarrow Customer: ={UID, C_{EnKey}, CS}
\]

ME4: Trader response the Contract Signing

\[
Trader \rightarrow Customer: ={UID, Time2, SeqN, [Cred], Flag|SignKey, C_{EnKey}}
\]

ME5: Trader process the Customer information Credentials with ZTMPS (Verifying Customer using Virtual Card information of the Customer \ Sender)

Trader \rightarrow ZTMPS: check the credentials

\[
Trader \rightarrow ZTMPS: Customer: \{ D_T(UID,I_c,I_T,T_5,ET(CS,M),H(CD)Dc),)\}
\]

or \(ME_6 = D_T \{ ME_5\}, H(CD)Ec\)

\[\text{Status ? Ok : resolve}\]

\[
Trader \text{ verifies the correctness of } CS \text{ and } E_{T}(CS,M). \text{ Trader also verifies } Customer’s \text{ signature } H(CD)Dc \text{ by decrypting it with } Customer’s \text{ public key to obtain a hash value } M’ \text{ and comparing it with the hash value } H_{Dc} \text{ contained in certificate } CS. \]

If the two hash values do not match, \(Trader\) may ask \(Customer\) to re-send message (ME5) or terminate the protocol execution.
Otherwise, *Trader* generates the VRES for his signature

ME9: *Trader* searches the items and availability etc of E-Goods or M-Goods selected by the customer

*Trader* → *Customer*: CS, Cost

**Status? Ok : resolve**

\[ ME9 = \{ D_7(UID, IC, IT, T_8, flag, ET(CS,M),H(CD)D_7) \} \]

*Trader* verifies the correctness of CS and \( E_s(EG,M) \). CS. If *Trader* may ask *Customer* to re-send message (ME5) or terminate the protocol execution.

Otherwise, *Trader* generates the VRES for his signature

ME 7: ZTMPS sends the Contract Signing the Customer through the Trader (Receiver)

\[ ZTMPS \rightarrow *Trader*: CS, Cost, PaymentRequest process from the Trader side \]

\[ ME_7 = \{ E_7(UID, IC, IT, Iz, T_9, flag, ET(CS,M),H(CD)Dc), CS \} \]

ME8: Customer process the ZTMPS Contract Signing (Decrypted the message ME7)

*Customer* → *Customer*: Ic, IT, T, CS, CD, SigC, CertC, CertT

\[ ME_8 = D_7(ME_7) \]

Status? OK: Resolve

*Trader* verifies the correctness of CS and \( E_s(EG,M) \). *Trader* also verifies *Customer*’s signature \( H(CD)Dc \) by decrypting it with *Customer*’s public key to obtain a hash value \( M' \) and comparing it with the hash value \( H_{Dc} \) contained in certificate CS. If the two hash values do not match, *Trader* may ask *Customer* to re-send message (ME5) or terminate the protocol execution. if matches then trader request the bank to process the payment

Otherwise, *Trader* generates the VRES for his signature

ME9: Customer sends the ack to from the Trader

\[ *Customer* \rightarrow *Trader*: Ic, IT, T, CS, SigC, CertC \]
ME10: Trader sends the ack to from the Sender:Customer

\[ ME_{10} = \{ E_c(UID, IC, I_z, T_{10}, flag, ET(CS,M), H(CD)Dc), CS \} \]

**Status ? OK: Resolve**

*Customer* verifies the correctness of *CS* and \( E_t(EG,M) \). *Customer* also verifies Trader’s signature \( H(CD)D_t \) by decrypting it with Trader’s public key to obtain a hash value \( M' \) and comparing it with the hash value \( H_DT \) contained in certificate \( CS \).

ME\(_{11} \): ZTMPS sends the ack to Trader about customer CS

\[ ZTMPS : Sends \text{ack of Customer} \]

\[ \text{Sender} \rightarrow \text{Trader : Receipt} \]

\[ ME_{11} = \text{Receipt} \]

**5.4.2 Threshold Certified E-mail delivery between Sender, Trader The Abort Sub Protocol**

In this section we present the abort sub protocol. We have defined it taking into account that it is possible that in some circumstances, a party that has signed a certified e-mail wants to cancel it prior to its being signed by the ZTTP. Thus, a party that has initiated the certified e-mail process can abort it fairly. As commented previously, the certified e-mail process is initiated with the *Handshake* message, it continues with the *Agreement* message and it finishes with the certified e-mail *message*. Therefore, if a party wants to abort this certified e-mail process once initiated, he/she has to execute the abort protocol before the certified e-mail message is generated by the TTP. For this abort protocol, the messages that we have needed to define are:
Fig. 5.10 Certified message delivery between Sender, Trader Abort sub Protocol

5.4.3 Threshold Certified E-mail delivery between Sender, Trader The Recovery Protocol

The recovery sub-protocol comprises steps (MR1)-(MR3), as shown in and is executed as follows.

Fig. 5.11 Threshold Certified E-mail delivery Recovery Protocol
ZTTP provides for each Sender $I_C$ a reading queue of messages (M) sent to $I_C$ by any Trader/Bank. Messages in the reading queue are ordered according to their sending requests to ZTTP. When $I_C$ requests to read a message in the queue, ZTTP computes a range of messages in the queue that are allowed to read by $I_C$.

(MR1):

_Sender process the information with symmetric key from ZTTP request the transfers the items e-certified mail (i.e. signature) recovery._

$Sender \rightarrow Trader: \text{CertE}, S_k I_c, I_T$.

(MR2): _ZTTP verifies the correctness of Trader’s recovery request by verifying Sender authorization token to ensure that Sender has presented the correct key. If this verification fails, ZTTP rejects Sender’s request. Otherwise, ZTTP recovers MR1 and sends it to Sender._

$ZTTP \rightarrow Sender: MR_2$

(MR3): _ZTTP also forwards the information to Trader._

$ZTTP \rightarrow Trader: S_k I_c, I_T$

### 5.5 Threshold Certified M-Goods/M-Pay Delivery System

M-goods refer to commercial products that can be represented in electronic form and transmitted over the Internet. Examples of such m-goods are video/audio content, software, m-newspapers, m-magazines, etc. m-goods delivery over the Internet is a business process where a piece of m-goods from a Trader is exchanged for its e-payment or an acknowledgement of its receipt (should a conventional payment method such as credit cards be used) from a Sender. Licensed retailers can distribute entertainment e-goods, such as music or films, instead of mailing them to Sender or selling them in shops. E-goods delivery involves several security concerns. In addition to authenticity and integrity assurance of the contents of e-goods and their confidentiality protection while in transit, there is another security service that is important in this context and that is **certified m-goods delivery**. This security service
ensures that e-goods is delivered to its intended recipient if and only if the sender can obtain a receipt proving that the recipient has indeed received the m-goods. The receipt is normally a digital signature signed by the recipient on the received m-goods, and the sender can use this receipt as a proof of delivery, should any dispute arise (for example, if the recipient falsely claims that the m-goods have never arrived). The sender typically also attaches his signature on the m-goods and transmits it together with the m-goods to ensure the receiver of the authenticity and origin of the m-goods.

Certified m-goods delivery requires two important security properties: fairness and content/quality assurance. Fairness guarantees that a recipient will receive the m-goods together with a proof of its origin from the sender if and only if the sender receives a non-reputable proof of the reception from the recipient. Content/quality assurance ensures that the received m-goods indeed match the expected description/quality. Achieving fairness over the Internet can prove to be a problem since the Internet, as a serial communication network, cannot support simultaneous message exchange.

We are developing new threshold protocol for an efficient Certified E-Good Delivery (NTRU-CEGD) protocol to support a fair exchange of m-goods for NTRU based receipt. The protocol is designed based upon a novel scheme enabling verification and recovery of encrypted signatures (NTRU-VRES) and joint m-goods and key certification. The protocol achieves both fairness and content/quality assurance.

The VRES scheme represents an encryption of an NTRU signature such that its recipient can verify the correctness of the signature without obtaining the signature itself. If the verification is positive, the recipient is guaranteed that an agreed ZTMPS can recover the original signature from the VRES, should any dispute arise. In this way, the ZTMPS can stay off-line, i.e. the parties can execute the exchange protocol without any involvement of the ZTMPS. If the parties cannot reach a fair completion of the exchange themselves, the recovery service offered by the ZTMPS is invoked and the dispute is resolved electronically. If the ZTMPS has no on-line presence, the
parties can submit the transaction evidence, for example using m-book’s or e-books or payment and the dispute can be resolved using a conventional method such as court of law. In addition, the involvement of the ZTMPS in the signature recovery process is transparent - the signature recovered by the ZTMPS is indistinguishable from that generated by the original signer. Both the signature and the m-goods enjoy confidentiality protection against unauthorized access by any third party, including the ZTMPS. The joint m-goods and key certification concept is used in the protocol design to allow the recipient of m-goods or m-payment to verify the correctness of the encrypted m-goods or m-payment and its decryption key during a protocol execution. In this way, content/quality assurance of the m-goods or m-payment can be achieved guaranteeing the recipient that the m-goods or m-payment obtained after decryption at the end of the exchange will indeed match with what is expected. The linkage between the original m-goods or m-payment, e-goods or m-payment encryption and the corresponding encryption/decryption key is validated and certified by a certification authority (CA) prior to the exchange.

5.5.1 Our Threshold NTRU-CEGD Protocol

In this section we formally present the protocol designed based on the VRES scheme described. The protocol describes 2 sub-protocols: the exchange sub protocol and the recovery sub-protocol. Parties Sender and Trader execute the exchange sub protocol in an attempt to exchange the items fairly without any involvement of ZTMPS. If this attempt is not successful, the recovery sub-protocol is invoked during which ZTMPS recovers the disputed signature.

For the design of protocol NTRU -CEGD, we assume that party Sender has valuable e-goods Da and symmetric key Sk for encryption and decryption of Da. Sender wishes to send the e-goods Da to Trader in exchange for Trader’s receipt for Da. To guarantee the correctness of content of Da, it has been certified by a specialized CA prior to exchange. Upon verifying Da, the CA issues Sender with certificate CertDa.
5.5.2 Threshold Certified m-Goods Delivery between Sender, Trader \ Group

![Diagram of Threshold Certified m-Goods Delivery between Sender, Trader \ Group]

**Fig. 5.12 Threshold Certified m-Goods Delivery between Sender, Trader\Group**

**MM1:** Sender (Customer) communicates the Group in the NFC location sending the broadcast messaging.  
Sender → M<sub>C</sub>, M<sub>G</sub>

**MM2:** Receiver accepts the request of the Sender in the NFC and send the response to Customer (Sender).  
Group → M<sub>G</sub>, M<sub>C</sub>, TS, ACK

**MM3:** Customer Sends the Message (goods purchase) to the trader in the NFC  
*Customer* → Customer: M<sub>G</sub>, M<sub>C</sub>, TS, MS

**ME1:** Customer request for connection establishment with Group

*Customer* → *Group*: (\{UID, TS, SeqN, [Cred], IC, TEnKey, CSignKey, Flag\}A<sup>1</sup>, CD,H(CD))

**ME2:** Group Owner response for connection establishment with Customer

*Group* → *Group*: (\{UID, TS, SeqN, [Cred], IC, TEnKey, CSignKey, Flag\}A<sup>1</sup>, CD,H(CD))
ME3: Group Owner request for the Group Member to response for Customer

Group Owner → Group O-Group M: \{\{UID, TS, SeqN, [Cred], I_c, T_{Esk}, C_{signKey}, Flag\}A^4, CD,H(CD)\}

ME4: Group response for connection establishment with Customer

Group → Customer: =\{UID, Time2, SeqN, [Cred], Flag\{SignKey, C_{Esc}\}

ME5: Customer request for Contract Signing

Customer → Customer: =\{UID, C_{Esc}, CS\}

ME6: Group response the Contract Signing

Group → Customer: =\{UID, Time2, SeqN, [Cred], Flag\{SignKey, C_{Esc}\}

ME7: Group process the Customer information Credentials with ZTMPS (Verifying Customer using Virtual Card information of the Customer \ Sender)

Trader → ZTMPS: check the credentials

Trader → ZTMPS: Customer: \{ D_T(UID, I_c, I_s, T_5, E_T(CS, M), H(CD)Dc) \}

or ME_6 = D_T \{ ME_5 \}, H(CD)Ec

Status ? Ok : resolve

Group verifies the correctness of CS and E_T(CS, M). Group also verifies Customer’s signature H(CD)Dc by decrypting it with Customer’s public key to obtain a hash value \( M' \) and comparing it with the hash value \( H_{DC} \) contained in certificate \( CS \). If the two hash values do not match, Trader may ask Customer to re-send message (ME5) or terminate the protocol execution.

Otherwise, Group generates the VRES for his signature

ME8: Group searches the items and availability etc of E-Goods or M-Goods selected by the customer

Trader → Customer: CS, Cost

Status ? Ok : resolve
ME9 = \{ D_T (UID, I_c, I_r, T_s, flag, ET(CS,M), H(CD)D_T) \}

Group verifies the correctness of CS and E_c(EG,M). CS. If Group may ask Customer to re-send message (ME5) or terminate the protocol execution.

Otherwise, Group generates the VRES for his signature

ME 10: ZTMPS sends the Contract Signing the Customer through the Group (Receiver)

\[ ZTMPS \rightarrow \text{Group: CS, Cost, PaymentRequest process from the Trader side} \]

\[ ME_{10} = \{ E_T (UID, I_c, I_r, I_z, T_s, flag, ET(CS,M), H(CD)D_c), CS \} \]

ME11: Customer process the ZTMPS Contract Signing (Decrypteds the message ME10)

Customer \rightarrow Customer: I_c, I_r, T, CS, CD, SigC, CertC, CertT

\[ ME_{11} = D_T (ME_{10}) \]

**Status?OK: Resolve**

Group verifies the correctness of CS and E_c(EG,M). Group also verifies Customer’s signature H(CD)D_c by decrypting it with Customer’s public key to obtain a hash value M’ and comparing it with the hash value H_{D_c} contained in certificate CS. If the two hash values do not match, Group may ask Customer to re-send message (ME5) or terminate the protocol execution. If matches then Group request the bank to process the payment if required for the business transaction.

Otherwise, Group generates the VRES for his signature

ME12: Group sends the checks the payment of the order to Bank

Group \rightarrow Bank : ME12: I_c, I_r, T, CS, SigC, CertC, Msg, TSD

ME13: payment status sends to the group to accept the order

ME13: Bank \rightarrow Group: I_c, I_r, T, CS, SigC, CertC, Msg, TSD

ME14: Group sends the ack to from the Sender\Customer

ME14 = \{ E_c (UID, I_C, I_T, I_z, T_10, flag, ET(CS,M), H(CD)D_c)CS \}
\( ME_{12} = \{ E_{c} (U_{ID}, I_{c}, I_{T}, I_{Z}, T_{io}, \text{flag, } ET(CS,M), H(CD)D_{c}), CS/ \)

Status ? OK: Resolve

Customer verifies the correctness of CS and \( E_{t}(EG,M) \). Customer also verifies Group’s signature \( H(CD)D_{T} \) by decrypting it with Group’s public key to obtain a hash value \( M' \) and comparing it with the hash value \( H_{D_{T}} \) contained in certificate CS.

ME15 : ZTMPS sends the ack to Group about customer CS

\( ME_{15} = \{ ET(IC, IT, T_{7}) \} \)

ME12: Group sends the ack to from the Sender/ Customer

Otherwise, Sender generates the VRES for his signature

5.5.3 Threshold Certified m-goods delivery recovery protocol

The recovery sub-protocol comprises steps (MR1)-(MR3), as shown in , and is executed as follows.

Fig. 5.13 Threshold Certified E-goods delivery recovery protocol

(MR1): Sender transfers the items M Goods list, Ic, I_{T} , to Trader or Group to request the receipt (i.e. Signature) recovery.

\( Sender \rightarrow \text{Trader or Group: CertEG, Ic, IT, IZ} \)

(MR2): ZTMPS verifies the correctness of Sender’s recovery request by verifying Trader’s or Group authorization token to ensure that Sender has presented the correct key. If this verification fails, ZTMPS rejects Sender’s request. Otherwise, ZTMPS recovers \( MR_{1} \) and sends it to Sender.
ZTMPS → Sender: MR₂

(MR3): ZTMPS also forwards the information to Trader.

ZTMPS → Trader: Ic,IT₁, IT₂

MB₁: Group sends the payment information to Bank

\[
\text{Group \ Trader} \rightarrow \text{Bank-Manager} : \text{Ic,IT₁, IT₂, Cost,Payment,SigC,CertC,CertT,M}
\]

\[
MB₁ = E_{BM}(\text{Ic,IT₁, IT₂, Cost,Payment,SigC,CertC,CertT})
\]

MB₂: Bank-Manager sends the payment information to Bank-Clerk for transaction

\[
\text{Bank-Manager} \rightarrow \text{Bank-Clerk} : \text{Ic,IT₁, IT₂, Cost,Payment,SigC,CertC,CertT,M}
\]

\[
MB₂ = E_{BC}(\text{Ic,IT₁, IT₂, Cost,Payment,SigC,CertC,CertT})
\]

MB₃: Bank-Clerk decrypts the message and processes the payment

\[
\text{Bank-Clerk} \rightarrow \text{Bank-Clerk} : \text{Ic,IT₁, IT₂, Cost,Payment,SigC,CertC,CertT,M}
\]

\[
MB₃ = D_{BC}(\text{Ic,IT₁, IT₂, Cost,Payment,SigC,CertC,CertT})
\]

MB₄: Bank-Clerk sends the payment status to Trader via Bank-Manager

\[
\text{Bank-Clerk} \rightarrow \text{Bank-Manager} \rightarrow \text{Trader} : \text{Ic,IT₁, IT₂, Cost,Payment,SigC,CertC,CertT,M,Status}
\]

\[
MB₄ = E_{T}(E_{BC}(\text{Ic,IT₁, IT₂, Cost,Payment,SigC,CertC,CertT}))
\]

MB₅: Bank-Clerk sends the payment status to Trader via Sender

\[
\text{Bank-Clerk} \rightarrow \text{Bank-Manager} \rightarrow \text{Trader} : \text{Ic,IT₁, IT₂, Cost,Payment,SigC,CertC,CertT,M,Status}
\]

\[
MB₅ = E_{s}(E_{BC}(\text{Ic,IT₁, IT₂, Cost,Payment,SigC,CertC,CertT}))
\]
5.5.4 Threshold Recovery Protocol

The recovery sub-protocol comprises steps (MR1)-(MR3), as shown in , and is executed as follows.

**CEGD Recovery protocol**

![Diagram](image)

Fig. 5.14 Threshold CEGD Recovery protocol

(MR1): *Sender* transfers the items M Goods list, Ic,IT, to ML *Trader or Trader or Group* to request the receipt (i.e.signature) recovery.

Sender → ML Trader: CertEG, Ic,IT

(MR2): *ZTMPS* verifies the correctness of *Sender’s* recovery request by verifying *Trader’s* authorisation token to ensure that *Sender* has presented the correct key. If this verification fails, *ZTMPS* rejects *Sender*’s request. Otherwise, *ZTMPS* recovers MR1 and sends it to *Sender*.

ZTMPS → Sender: MR2

(MR3): *ZTMPS* also forwards the information to ML *Trader or Trader or Group*

ZTMPS → Trader: Ic,IMT
5.6. GENERATING M-PAYMENTS SYSTEM USING OUR NOVEL NTRU PROTOCOL

5.6.1 Structure of our Mobile Cheque Payment System

Most cheque based transactions will be between businesses -thus a special hardware is attached to Mobile\Portable for signing payments. Signature is encrypted by hardware. All public keys are generated using our Novel NTRU Protocol of business partners authenticated by certifying agencies.

5.6.2 Steps in transaction

1. Purchaser sends purchase order and payment advice signed with his private key to vendor. He also sends his public key certificate encrypted with vendor's public key to vendor

2. Vendor decrypts with his private key, checks certificate and cheque, attaches deposit slip, encrypts with bank's public key and sends it to bank. he also sends his public key certificate

3. Bank checks signatures, credits and clears cheque. Credit advice goes to vendor, and consolidated debit advice sent to purchaser periodically

Mobile Payment process in Electronically

5.6.3. Mobile Electronic Clearance (MEC) Payment Protocol using NTRU-MEC

To use our Novel EC protocol it is assumed that

1. Each party involved in m-commerce transaction has a public and private key. A public key encryption is used such as NTRU.

2. All parties have their public keys certified. (TX509C3 using NTRU )

3. A standard hashing algorithm is used to create message digest for signature verification.
Main Features

- Sender credit card number is not revealed to a Trader. It is revealed only to the acquirer who authorizes payment.

- Purchase invoice details are not revealed to the acquirer. Only the credit card number can be used as virtual card in the mobile and total amount are revealed to him.

Fig. 5.15 Mobile Clearance Payment Electronically

- Purchase invoice + Virtual credit card number which has been mapped to the mobile device is digitally signed by the Sender. In case of a dispute an arbitrator can use this to settle the dispute.

Steps followed in the protocol are:

1. Sender purchase information has 3 parts
   
   (i) Purchase Order (PO)

   (ii) Virtual Credit Card Number or Mobile Credit Card Number (VCCN\MCCN)

   (iii) Amount to be paid
2. Trader should know \((PO + Amount) = POA\)

3. Acquirer should know \((VCCN + Amount) = CCA\)

4. Hash POA using standard Hash algorithm such as MD5. Call it POD.

5. Hash CCA using MD5. Call it CCD

6. Concatenate POD and CCD. Call it \((POD||CCD)\)

7. Hash \((POD||CCD)\) giving PPD

8. PPD is encrypted using Private key of Sender. This is Sender’s Digitally Signed purchase order \(DS = \text{Encrypt (PPD)}\) with \text{CPRK}

\text{CPRK} is Private key of Sender. This is sent to Trader by Sender.

9. POA separately encrypted by Sender using Trader’s public key and sent to Trader

10. Trader decrypts it using his private key. He thus gets Purchase order Amount. He can hash it and get POD

11. CCD and DS also sent to Trader. From CCD Trader cannot find CCN.

12. Trader can decrypt DS using Sender’s public key and get PPD. Sender must have a certified public key for verification.

13. Trader can compute \(H(POD||CCD)\)

\(H(POD||CCD) = \text{PPD}, \text{then Sender’s signature is OK.}\)

14. Trader forwards to acquirer VCCA, POD, DS each separately encrypted using acquirer’s public key.

15. Acquirer’s forwards to bank.

16. Bank finds VCCN and Amount. Verifies balance amount. Bank also verifies Sender’s digital signature using CCD, POD and DS. If all OK acquirer is informed.

17. Acquirer OK’s transaction to Trader

5.6.4. Transaction between Sender, Acquirer, Trader, Bank

![Fig. 5.16 Transaction between Sender, acquirer, Trader, Bank](image)

**Step 1**: Start the communication. Trader near in the NFC location. Sender fills Purchase order, amount and swipes the virtual credit card number in his near terminal or receiver terminal A software in terminal read strips it into two parts Purchase Order + Amount (POA), Virtual Credit Card No. + Amount (CCA). POA is encrypted using Trader’s public key and CCA with bank’s public key. These are sent with Sender’s public key certificates, CCD and DS. Trader verifies DS.

**Step 2**: Trader forwards to acquirer DS and CCD (These are encrypted using acquirer’s public key)

**Step 3**: Acquirer forwards to bank. Bank decrypts CCA with its private key. Checks validity of Virtual credit card and balance. If OK informs acquirer

**Step 4**: Acquirer OK’s transaction to Trader and credits Trader's account.

**Step 5**: Acquirer sends status to Trader

**Step 6**: Sender’s order and proceeds to dispatch items.

**Step 7**: At the end of the Trader’s sends information to Sender.
5.7 CONCLUSION

This Chapter presents an extensive security analysis of our Threshold fair exchange protocols. We have showed that the protocols meet all the security requirements identified as their design objectives. These include strong fairness, non-repudiation, m-good’s\e-goods’ content assurance, fair recovery authorization, ZTMPS- invisibility, ZTMPS -confidentiality, limited role of the ZTMPS, and timeliness. Thus, we have confirmed our research hypothesis that our Threshold fair exchange protocols can be designed to provide all these functionalities. In addition, the formal verification of the protocols has further confirmed our intuitive analysis that the strong fairness property is indeed satisfied by our protocols. Performance evaluation and comparison with related work have showed that our protocols have stronger security features, yet achieve them with less computational overheads for the participating parties.