Enforcing Honesty in E-Commerce Fair Exchange Protocols

Alaraj, Abdullah M. S.

Enforcing Honesty in E-Commerce Fair Exchange Protocols

Abdullah M. S. Alaraj

A thesis is presented for the degree of Doctor of Philosophy

Department of Computer Science

Durham University

October 2008

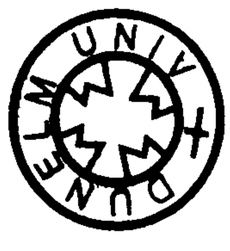

Table of Contents

Table of Figures

Table of Tables

Abstract

The growth of the Internet attracted many users either as merchants or customers. Merchants have products and services to sell whereas customers have money to pay for these products and services. Customers normally do not trust merchants and also merchants do not trust customers. When a customer is willing to buy any product or service, they want be sure that the merchant will send them the product that they want after they make the correct payment. Similarly, a merchant will not take the risk by sending the product before the customer makes the correct payment. Therefore, the party (customer or merchant) who send their item (payment or product) first will be at risk of the misbehaviour of the other party. This problem is known as the fairness problem.

The aim is to study the existing fair exchange protocols that solve the fairness problem. Then, propose more efficient protocols to solve the fairness problem.

The original idea in this thesis is enforcing honesty in fair exchange protocols. The idea of enforcing honesty is applied to produce a fair exchange protocol that encourages the merchant to be honest and enforces the customer to be honest and vice versa. The thesis shows that it is not possible to enforce both the customer and the merchant to be honest at the same time. Hence, it proposes a third fair exchange protocol that encourages the customer and the merchant to be honest. The protocols make use of a Trusted Third Party (TTP) but its use is kept to minimum when disputes arise. In this respect they are optimistic fair exchange protocols.

The proposed protocols have the following features: (1) only three messages are required to be exchanged between a customer and a merchant that apply the idea of enforcing a party to be honest; (2) the protocols guarantee strong fairness for both customer and merchant; (3) they allow both parties (customer and merchant) to check the correctness of the item of the other party before they send their item; (4) disputes are resolved automatically online by a Trusted Third Party (TTP); and (5) they are efficient in that they have a low number of modular exponentiations (which is the most expensive operation).

Applying the idea of enforcing a party to be honest has helped in proposing efficient fair exchange protocols for the exchange of payments and digital products between customers and merchants.

Acknowledgment

First of all, all thanks and praise would first go to God (Allah) for all the success.

My sincere thanks would go to my supervisor Professor Malcolm Munro for all his support, time and guidance. This thesis would not be completed without the in-depth discussions and comments from Professor Munro.

My sincere thanks would go to all my family (my parents, my wife, my brothers and sisters) for their support and prays.

Declaration

Parts of the work presented in this thesis have been published in the following journal, conference proceedings, and book chapter.

Journal publication:

1. Alaraj and M. Munro: An e-Commerce Fair Exchange Protocol that Enforces the Customer to be Honest. International Journal of Product Lifecycle Management, IJPLM. Inderscience publisher. ISSN: 1743-5110, (To appear)

Conference proceedings:

1. Alaraj and M. Munro "An e-commerce Fair Exchange Protocol for exchanging Digital Products and Payments", In Proceedings of IEEE/ACM ICDIM'2007, Lyon, October 2007. pp 248-253
2. A. Alaraj and M. Munro "An Efficient Fair Exchange Protocol that Enforces the Merchant to be Honest", In Proceedings of IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing 2007, CollaborateCom 2007, New York, November 2007, pp. 196-202
3. A. Alaraj and M. Munro: An efficient e-Commerce Fair Exchange Protocol that encourages Customer and Merchant to be Honest. In the proceedings of the 27th International Conference on Computer Safety, Reliability and Security, 2008 (SafeComp 2008). Lecture Notes In Computer Science, LNCS, Vol. 5219, pp. 193-206

Book chapter:

A. Alaraj and M. Munro: Enforcing Honesty in Fair Exchange Protocols. In the book: Emergent Web Intelligence, to be published by Springer Verlag.

Glossary and Notations

The following presents the notations used in the proposed protocols i.e. the notations are used in ECH protocol, EMH protocol and ECMH protocol. The notations are as follows:

- *C*: Customer
- *M*: Merchant
- *TTP*: Trusted Third Party, a party that is neither M nor C and is trusted by all parties
- *D*: Digital product
- *P*: Payment
- *CA*: Certificate Authority used to certify digital products
- *CB*: the Customer's Bank
- *desc.*: description of digital product specified by C
- *h(X)*: "a strong-collision-resistant one-way hash function, such as SHA-1" [Schneier96]
- *pkx* = (*ex*, *nx*): RSA [RiShAd78] Public Key of party *x* (*x* ∈{C, M, TTP, mt, ct}), where *nx* is a public RSA modulus and *ex* is a public exponent
- *skx* = (*dx*, *nx*): RSA [RiShAd78] Private Key of party *x*, where *nx* is a public RSA modulus and *dx* is a private exponent
- *kx*: a symmetric key used by party *x* (*x* ∈{C, M, TTP})
- *P-Cert*: Payment Certificate issued by CB. The contents of *P-Cert* are:
    - *amount:* the amount of payment
    - *payee:* the name of the receiving party i.e. name of the merchant
    - *hP:* hash value of payment
    - *heP:* hash value of encrypted payment with kc
    - *heKc:* hash value of encrypted *kc* with *pkct*
    - *Sig.CB:* CB's signature on *P-Cert*
- *D-Cert*: Digital-product's Certificate issued by CA. The contents of *D-Cert* are:
    - Price: price of D

  - *d*: Description of D, the description may be the ID of the digital product
  - *hD*: hash value of D
  - *heD*: hash value of encrypted D with *km*
  - *heKm:* hash value of encrypted *km* with *pkmt*
  - *Sig*.CA: CA's signature on *D-Cert*
- *DG-Cert*: another version of the digital product certificate used in the EMH protocol. The content of *DG-Cert* are:
  - Price: price of D
  - *d*: Description of D, the description may be the ID of the digital product
  - *hDG*: hash value of D
  - *Sig*.CA: CA's signature on *DG-Cert*
- *C.mt*: the certificate for the shared public key between M and TTP; *C.mt* is issued by TTP. A standard X.509 certificate can be used to implement *C.mt* [X.509]
- *C.ct*: the certificate for the shared public key between C and TTP; *C.ct* is issued by TTP. A standard X.509 certificate can be used to implement *C.ct* [X.509]
- *enc.pkx(Y)*: an RSA encryption [RiShAd78] of Y using the public key *pkx* (*ex*, *nx*) i.e. enc.pkx(Y) = $Y^{ex} \bmod nx = Z$
- enc.skx(Z) is an RSA decryption [RiShAd78] of Z using the private key *skx* (*dx*, *nx*) i.e. enc.skx(Z) = $Z^{dx} \bmod nx = Y$
- *enc.kx(Y)* : encryption of Y using a symmetric key *kx* (kx can be used for decrypting enc.kx(Y))
- *Sig.x (A)*: the RSA signature [RiShAd78] of party *x* on *A* i.e. encrypting the hash value of *A* using the private key *skx* (*dx*, *nx*) as follows:

  Sig.*x* (*A*) = $(h(A))^{dx} \bmod nx$
- A → B: X: A sends message X to B
- + : the concatenation operation
- ECH: Enforcing Customer Honesty protocol
- EMH: Enforcing Merchant Honesty protocol

- ECMH: Encouraging Customer and Merchant Honesty protocol

Chapter One

1. Introduction

1.1. Introduction

The research question to be addressed in this thesis is as follows. The way in which the current optimistic fair exchange protocols (which are the best type of fair exchange protocols, as will be discussed later) work is by, first, letting the parties exchange their encrypted items then the exchange of the decryption keys will take place afterwards. This exchange requires at least four messages to be exchanged between the parties. Therefore, is there a way of reducing the number of messages in the optimistic fair exchange protocols in order to have more efficient protocols?

E-commerce has grown dramatically in the last decade and many merchants now conduct their business online. That is, more and more merchants use the Internet to sell their products and services to people who live in different parts of the world. Therefore, their products and services are not limited to people in their areas. A recent national survey shows that the value of Internet sales by UK businesses grew from £101bn in 2005 to £130.4bn in 2006 [Stat07]. The growth of e-commerce has also changed the way in which some customers buy products and services. More and more customers rely on the Internet for buying. This is for many reasons. Firstly, it is much easier to make a purchase using the Internet because a customer does not need to wait in traffic jams and go to shops. Secondly, customers can compare prices with different merchants in seconds. Thirdly, customers can buy from anywhere in the world at any time that suits them. Fourthly, the product that a customer buys from the Internet will be delivered to the customer's home.

Using traditional commerce, a customer goes to the merchant's shop, finds the product they want, pay for it, and finally collects the product. In this scenario, the customer will not worry that they will not be given the product when they pay the money. Additionally, if the customer pays in cash then they will not worry that their financial information will be revealed to any other party. Furthermore, customers can be anonymous to merchants. In other words, if the customer does not want the merchant to trace them or to know their purchase habits then they can pay by cash by which merchants cannot trace customers.

The issues discussed in the traditional commerce can be problematic when using e-commerce. Customers need to trust the merchant from whom they buy. This is because if the merchant is dishonest then they may not send the product to the customer or they might send an incorrect product. Customers also need to be concerned when they make payment online because their financial information might be stolen if it is not encrypted while it is sent to the merchant. Additionally, even if the customer's financial information is sent securely from the customer to the merchant, the merchant must have mechanisms in place to secure the customers' financial information. In e-commerce, the anonymity of customers is not always protected. That is, when a customer uses a credit/debit card for the payment, their name is associated with the card. Therefore, information on the customer's purchase behaviour can be constructed by merchants.

Although the use of the Internet eases buying and selling of products and services, it also requires more attention to be paid for issues such as trust, security, anonymity, and privacy. Hence, there must be protocols to protect both customers and merchants when they use the Internet for buying and selling products and services.

1.2. Fair Exchange Protocols

A customer and a merchant are two parties who possibility do not know each other (and more likely they do not trust each other especially if they have not

previously dealt with each other). The customer wants to buy a digital product (such as a piece of music or software) online from the merchant. The merchant needs to receive the correct payment from the customer in order to send the digital product to them. The customer wants to be sure that they will receive the correct digital product when they send the payment to the merchant. That is, when the customer sends the payment, the merchant will not disappear without sending anything or sends the wrong digital product and disappears. This problem is called fair exchange of items in which both parties must receive each other's items or no party gets anything. Therefore, the first problem that fair exchange protocols address is how to ensure fair exchange of items between the parties. The second problem they must address is that if any disputes arise between the parties then the fair exchange protocol should be able to provide an online and automated dispute resolution that satisfies both parties.

Two types of fair exchange protocols exist: those that do not involve a Trusted Third Party (TTP); and those that do. The use of the TTP helps the parties to exchange their items fairly. The protocols that do not involve a TTP are based on dividing the items to be exchanged into parts. Then, each party sends a part of its item to the other party until the whole items are exchanged.

Protocols that involve a TTP can be divided into three types:

1. The first type uses a TTP for delivering the exchanged items, this is called an inline TTP based protocols. This involves each party sending their item to the TTP and the TTP delivering them to the parties. Involving a TTP will guarantee the fair exchange of items
2. The second type is the protocols that use online TTP. The online TTP is only used for validating the items to be exchanged. Therefore, the involvement of the TTP is reduced.
3. The third type is the protocols where there is minimal use of the TTP, usually when something goes wrong. In these protocols the two parties directly exchange their items and the TTP only gets involved to resolve disputes. This type of protocol is called optimistic fair exchange protocols.

Both types of fair exchange protocols (the one that involves a TTP and the one that does not) aim to ensure the fair exchange of items for all parties involved in the exchange. The protocols that do not require a TTP are not efficient in that they require many rounds to complete the exchange. Additionally, the parties must have the same computational power for the fairness to be ensured. The protocols that involve an inline TTP suffer from drawbacks such as the TTP could be the source of a bottleneck, it must always be available, and if the TTP crashes, the protocol will not deliver the items to the parties. Although the protocols that are based on online TTP reduce the involvement of the TTP, they have similar drawbacks as the protocols that use inline TTP. The optimistic fair exchange protocols seem the best way to solve the fairness problem. This is because the TTP is not involved in the protocol unless a problem arises.

This thesis has studied the existing fair exchange protocols in the literature and proposes three efficient optimistic fair exchange protocols that can be used for exchanging digital products (such as software) and payments between customers and merchants. The proposed protocols ensure fairness for all parties involved in the exchange and resolve disputes automatically online.

1.3. Criteria for Success

The following criteria for success are set:

1. Development of efficient optimistic fair exchange protocols

A number of optimistic fair exchange protocols have already been developed. This thesis will analyse them and develop new protocols that overcome some of the issues identified. Then, the new protocols will be compared against the relevant fair exchange protocols.

2. Specification of the efficient optimistic fair exchange protocols

The new protocols will be specified, the number of messages needed will be identified, and the content of these messages will be specified.

3. Built in automatic dispute resolution

Disputes between the merchants and customers are bound to arise. These are usually resolvable in a court of law. The new protocols should minimise the instances of disputes and should also enable their automatic resolution.

4. The protocols should ensure strong fairness for all parties

The new protocols should ensure that by the end of executing the fair exchange protocol then each party will get the item of the other party or none do.

5. Analysis of the new protocols for completeness

The new protocols should be analysed so that strong fairness is ensured in all possible scenarios, the ability of each party to check the correctness of each other's items, all dispute scenarios to be identified, and the new protocols to be formally verified.

6. A proof of concept implementation

A prototype proof of concept is needed to make sure that the new protocols are viable, implementable, and work correctly.

1.4. Thesis Structure

The thesis structure is as follows. Chapter 2 presents general background on electronic commerce (e-commerce). Chapter 3 studies the types of fair exchange

protocols, presents examples of fair exchange protocols, and discusses dispute resolution. Chapter 4 presents some cryptographic concepts and assumptions for the proposed protocols. Chapter 5 presents and analyses the Enforcing Customer Honesty (ECH) protocol. Chapter 6 presents and analyses the Enforcing Merchant Honesty (EMH) protocol. Chapter 7 presents and analyses the Encouraging Customer and Merchant Honesty (ECMH) protocol. Chapter 8 compares the ECH, EMH, and ECMH protocols against each other, and then compares the three protocols against the related fair exchange protocols from the literature. Chapter 9 formally verifies the proposed protocols. Chapter 10 presents the design and the implementation of the prototype of the proposed protocols. Finally, chapter 11 presents the conclusions and the future work.

1.5. Summary

In this chapter, a brief introduction to the e-commerce has been presented. The fairness problem is discussed. The criteria for success are presented. The thesis is outlined.

Chapter Two

2. Electronic Commerce

2.1. Introduction

Electronic Commerce (e-commerce) is defined as "the sharing of business information, maintaining business relationships and conducting business transactions by the means of telecommunication networks" [TsiakisStheph05]. E-commerce is performed using electronic communications such as computer networks, including the Internet, and are used by both a customer and a merchant, to carry out all their transactions. These transactions include buying, selling, transferring or exchanging products, services or information. A typical trade cycle that can be applied in the world of e-commerce is as follows:

1. a customer searches the web for products;
2. they place an order for the found product with a merchant;
3. the customer pays the price of the product to the merchant;
4. the merchant sends the product to the customer either by shipping it or through the Internet if it is a digital product such as a piece of music or computer software;
5. the customer can obtain after-sales services such as warranties

Figure 1 illustrates this process.

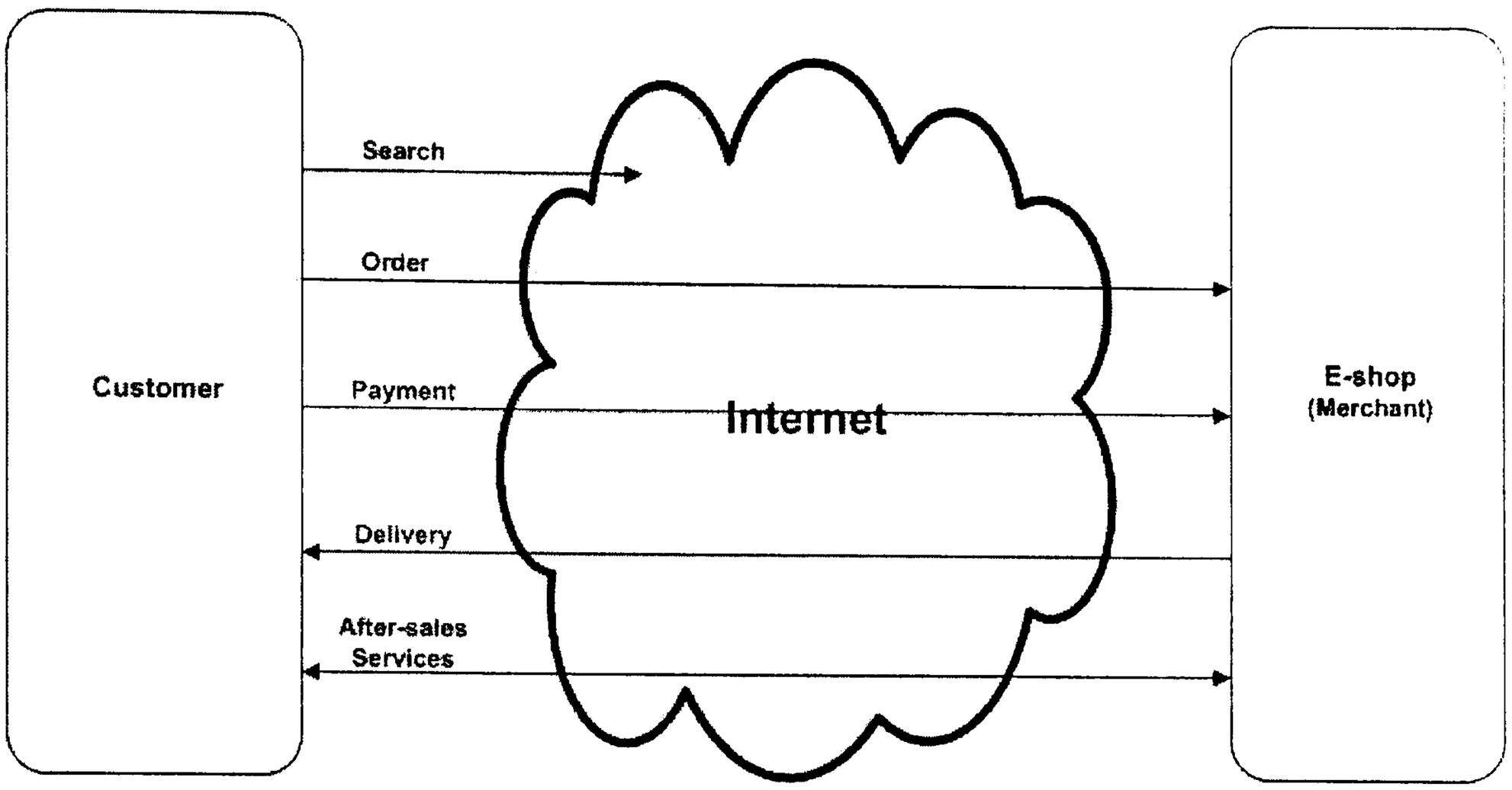

Figure 1: E-commerce Facilities (Adopted from [Whiteley00])

In Figure 1, the primary objective that a customer needs to get is the ordered product (whether it is a digital product or not) that a merchant has and at the same time the primary objective that a merchant needs to get is the payment that the customer has. Of course other services that the merchant offers for customers such as the ease of ordering and after-sales services are important but the primary objective is to have the correct product that the customer wants. Therefore, the customer and the merchant exchange their items which are the product (whether it is digital or not) and the payment. This exchange of items needs to be fair. That is, by the end of the transaction, both the customer and the merchant must have each other's items or neither of them gets anything. The process of fairly exchanging items is called a fair exchange protocol which will be discussed in the next chapter.

There are many types of e-commerce. The most common types are summarised as follows [Whiteley00]:

1. **Business-to-Consumer (B2C):** the online transactions made between business and individual consumers.

2. **Business-to-Business (B2B):** the online transactions made between business and another business, such as suppliers.
3. **Consumer-to-Business (C2B):** individual consumers who sell products or services to organizations.
4. **Consumer-to-Consumer (C2C):** individual consumer sells directly to another consumer.

Figure 1 represents B2C e-commerce as there is a merchant (business) and a customer who interact with each other. However, the processes that appear in Figure 1 can be applied to all four types of e-commerce. Therefore, in B2B e-commerce there are searching for the wanted products, placing orders, paying the price of the ordered products, delivering the products and providing after-sales services. In the same way these processes can be applied to the rest of e-commerce types.

2.2. Advantages, Disadvantages and Limitations

E-commerce has many advantages and benefits to customers and merchants; it also has some disadvantages and limitations. These advantages, disadvantages and limitations will be summarised in the following sections [TuKiLeVi04] and [Whiteley00].

2.2.1. E-commerce Advantages

- **Shopping from home:** customers can shop from their homes using the Internet. Hence, they avoid travelling; traffic jams, and pollutions
- **Shopping at any time:** there are no opening and closing times for e-commerce websites. As a result, customers can shop at any time they like day or night. This feature of e-commerce is beneficial for merchants as well because they can sell their products at any time

- **Cheaper prices:** products sold via the Internet usually have cheaper prices than products sold in traditional shops because for example traditional shops usually employ more people and hence there are higher running costs
- **Home delivery:** free home delivery is provided by most of e-commerce websites if the customer spends over a certain amount of money
- **Online sales support:** information on products is available on the e-commerce websites if customers need it
- **Global reach:** using e-commerce, geographical barriers are ignored as merchants can sell their products to any customer in the world
- **The number of employees is reduced:** in e-commerce, most of the operations are automated so the number of employees will be reduced
- **Instant delivery:** if the product to be bought from the merchant using the Internet is a digital product (such as software, movie) then it will be delivered to the customer immediately via a link in the merchant's website or e-mail
- **Shopping for all people:** all people can do shopping even those who live in rural areas or people who have special needs

2.2.2. E-commerce Disadvantages

Although e-commerce has many advantages, it has disadvantages which include the following:

- **Privacy and security:** they are the major concern for both customers and merchants. Customers want their privacy to be protected when they buy using the Internet. The security of their financial information is the most concern for customers. Merchants are also concerned about the security of their systems because if it is not well secured then this will affect their reputation
- **Delivery:** buying from the Internet may result in the delay of delivering the bought products. Sometimes it may result in not delivering the products to

the customer at all or delivering wrong products. This might occur if the merchant is dishonest. This problem is called fair exchange problem which is the main focus of this thesis. Hence, it will be discussed in the next chapter and in the rest of this thesis

- **Inspecting products:** buying from the traditional shops will allow customers to inspect and feel the products whereas this inspection is not available when buying from the Internet. Instead, a picture of the products will be displayed for customers
- **Social interaction:** buying from the traditional shops allows people to communicate with each other. On the other hand, buying from the Internet will keep people away from other people as customers will rely on computers for performing their purchases. Therefore, this might result in long term social problems
- **Returning products:** returning the products that have been bought via the Internet is more restricted, especially if the merchant is located in a different country from that of the customer

2.2.3. E-commerce Limitations

Some authors agreed that the e-commerce limitations can be classified into technological limitations and non-technological limitations [AhRyHa04], [KimKim05] and [TuKiLeVi04]. These limitations will be discussed as follows.

The technological limitations

The technological limitations are about the limitations that exist in the technology that has been used in e-commerce. These include the following:

- The universal standards of quality, security and reliability are not yet agreed
- The cost of Internet accessibility is still expensive in most countries
- The quality measurement factors are varied and yet there is no agreement on them

The non-technological limitations

The non-technological limitations are about the threats that users have experienced when they used e-commerce. These include the following:

- Security and privacy threats are the major concerns for most customers who buy products and services online. This is because of the number of incidents that have happened as a result of online payments (see E-Commerce Security section)
- Lack of trust in e-commerce because if customers do not trust the merchant, then they will not buy products online from them (see E-Commerce Trust section). Some customers also do not trust the technology because they want paper-based commerce
- The amount of fraud on the Internet prevents some customers from using e-commerce.

2.3. E-Commerce Security

The security element in e-commerce is crucial. Customers will buy products and services from merchants who have systems that are known to be secure. Therefore, merchants must spend a lot of money on securing their systems in order to encourage customers to buy from them. $6.2 billion was spent on security around the world in 1999; whereas $25 billion was spent in 2002 [TuKiLeVi04]. This indicates that merchants are aware of the security problem and they are spending a lot of money to solve it. It also indicates that the threats are growing as well.

E-commerce systems are vulnerable to abuse, misuse and failure [FordBaum97]. The consequences of these things on e-commerce systems will affect both merchants and customers. These consequences can be identified as follows [ibid]. Firstly, lots of money will be lost from both customers and merchants. In 2000, well-known e-commerce websites (including amazon.com, Buy.com, and eBay.com) were attacked and as a result the loss of money was about $1.7 billion [TuKiLeVi04]. Secondly, confidential information such as credit/debit card numbers might be stolen.

As an example of this, an attack targeted "*TK Maxx*" stores in 2007 and led to stealing information from 45.7 million payment cards (debit and credit) [BBC08]. Thirdly, the customers' trust in the e-commerce system will be compromised if it has any security breaches. As a result, customers will not buy from a merchant if they know that the system is not secured. Finally, resources in the system might be used by unauthorized people.

As can be seen from the consequences, the need to have secure e-commerce systems is crucial. By having secured e-commerce systems, customers will be encouraged to use them. At the same time, merchants will increase their profit if more and more customers use their systems to buy products and services.

Two kinds of attacks might compromise e-commerce systems [TuKiLeVi04]. The first type is the non-technical attack. This attack can be performed by persuading people who work with the system to reveal sensitive information or to perform actions that compromise security. The second type is the technical attack. To perform the technical attack, software and systems expertise are involved. Computer viruses, worms and Trojan Horses are examples of technical attacks.

Three fronts must be addressed in order to get secure e-commerce systems [Ghosh98]. These fronts are described as follows:

1) Customer's side security

The software (such as the Internet browsers) used by customers to purchase products and services from the Internet may compromise the security of the e-commerce if they are not well secured.

2) Data transport security

Customers need to pay money via the Internet or send confidential information when they use e-commerce systems. The sensitive information to be sent from customers to merchants via the Internet must be secured. Many approaches can be applied to secure the information; they include:

- Public key infrastructure (PKI): this includes private and public key encryption, digital signature and digital certificates

- Secure socket layer (SSL) protocol [SSL]
- Secure Electronic Transaction (SET) protocol [SET]

3) Merchant's side security

Securing the merchants' systems is very critical as very sensitive information (for both customers and merchants) is stored on these systems. Therefore, if the merchants' systems are not secured, the customers trust in the merchants will be compromised.

To sum up, an e-commerce system is used to perform buying and selling products (whether they are digital or not) between a customer and a merchant online. Therefore, in order to get secure e-commerce systems that are used by customers and merchants, the customer side should be secured as it is used for sending payments and confidential information, the merchant side should be secured as it is used for sending the digital products that the customer ordered and also used for storing customers' information; and the communication channel between the customer and the merchant should be secured to protect the sent items between the customer and the merchant. The customer side can be secured by using secured software. The merchant side can be secured by using a secured web server and a secured operating system for the network server. Finally, the communication channel can be secured by using strong security infrastructures and protocols such as SSL, SET, and PKI.

The other security problem is how to secure both the customer and the merchant from each others i.e. how to protect an honest customer from dishonest merchant and vice versa. This issue is the main focus of this thesis and will be studied starting from the next chapter.

2.4. E-commerce Trust

Trust is an important element in business. The importance of trust increases when the business is done online because it is more likely that both customers and

merchants do not know each other. Trust and security are seen as the most important issues emerging with regard to the development of e-commerce [RuUnHa03]. Lack of trust in merchants prevented customers from buying products online [CookLuo03]. A survey of 60 agents at U.S. companies shows that 45% of the agents said that a lack of trust prevented them from buying goods and trading online more frequently [Violino02]. The more customers trust merchants, the more they buy from them.

Trust has been defined as "*the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party*" [TsiakisStheph05]. Therefore, if each party trusts the other, each of them will act as the other party expects. When a customer trusts a merchant, they will be confident that the merchant will send the same product that they have paid for, within the agreed time.

In e-commerce, trust can be divided into two types. The trust in the technology used and the trust in the partner with whom a user will interact. The trust in the technology used is built through the reliability of the system. A lack of the reliability of the technology can affect the trust in the e-commerce [CoThYi03]. The trust with the partner is difficult to build but easy to lose and has cost implications [CookLuo03]. So, trust is something that can be built over the time. For example, for a customer and a merchant, the customer needs to deal with the merchant many times to have this kind of trust. The same for a merchant, there are customers who are trusted by a merchant and there are customers who are not trustworthy. Therefore, the previous experience will play an important role in building trust with the other party. Trust can also be earned from the experience of other people. For example, if a customer has a good experience with a merchant who is trustworthy, then this customer will suggest this merchant to other customers. So, the trust can be gained by having a good reputation.

Srinivasan [Srinivasan04] identifies five factors that contribute to customer trust in a merchant in e-commerce:

1. the ease of access to the description of products and services in a merchant's web site

2. an order can be placed in an easy way
3. when a customer placed an order, they receive order confirmation
4. customers are able to track their orders
5. customers are supported by after-sales services

These factors will give a customer confidence that a merchant is trustworthy especially when a customer is dealing with a merchant for the first time. After the customer has experienced the merchant whether they are trustworthy or not, they will be more confident to deal with them for the next time or even suggest them to other customers. Therefore, the trust can be built by past experience or by other factors such as good reputations.

Many methods can be used to increase the trust level between customers and merchants. In e-Bay [EBay], customers are allowed to add comments when they buy from a merchant. This is a manual process. These comments represent buyers' (customers') view of a merchant who sold items to them. The comments can be seen by any visitor of the e-Bay's website. Using this, merchants will do their best to have a good reputation which will encourage new customers to buy from them.

The idea of increasing the trust level used by e-Bay can be called a trust profile for a merchant. The same can be done for customers. So, when customers buy from a merchant, the merchant can view the trust profile of a customer in order to know whether they are trustworthy or not by viewing the previous actions of the customer. The trust profile is done manually and there is scope for automation, possibly through the use of fair exchange protocol or some other mechanisms.

As can be seen, the trust factor is very important for both a customer and a merchant. Trust in the partner with whom a user (a customer or a merchant) will interact can either encourage or discourage the user from using e-commerce systems. Therefore, there is a crucial need for having well designed e-commerce protocols that increase the trust between customers and merchants. Although there are users (customers and merchants) who are not trustworthy, e-commerce protocols should ensure fair exchange of payments and digital products between customers and merchants.

2.5. E-payment

Electronic payment (e-payment) is described as the transfer of payment from the payer to the payee using electronic means. E-payment systems imitate the payment mechanism in the real world as well as inventing new mechanisms for implementing the transactions [TsiakisStheph05].

There are at least four parties involved in any e-payment system. They are a payer (customer), a payee (merchant), an issuer, and an acquirer. The issuer is a third party such as a bank which processes payer's transactions i.e. the customer's bank. The acquirer is a third party such as a bank which processes payee's (merchant) transactions i.e. the merchant's bank.

Asokan et al [AsJaStWa97] identified two models of e-payment systems. They are "*cash-like*" e-payment systems and "*check-like*" e-payment systems. It seems that the reason for this classification is that in the "*cash-like*" e-payment systems the parties (payer and payee) do not have to have bank accounts i.e. the same as in the real cash money where payer directly gives the money to the payee; whereas in the "*check-like*" e-payment systems both parties (payer and payee) need to have bank accounts either with the same bank or in different banks.

In the "*cash-like*" e-payment system, a payer withdraws the money from the issuer before they use it for purchasing over the Internet. When the payer wants to buy something online, they send the money (that was previously withdrawn) to the payee. The payee then deposits the money using the acquirer. The acquirer then settles the money from the issuer [AsJaStWa97]

In the "*check-like*" e-payment system, the payer needs to get a card (either debit card or credit card) to be used for making payments and also the payee need to have a bank account. A credit/debit card can be defined as an electronic readable card that contains unique information for every card that is used for payment. The difference between a debit card and a credit card is as follows. In the credit card the card holder (payer) borrows the money from the issuer of the card and later the payer

pays the money back. In the debit card, the card holder (payer) uses the money that is available in their bank account.

The way in which the credit card e-payment system works can be summarised as follows [TuKiLeVi04]. When the payee (card holder/customer) fills in the payment form and submits it, the page will be transmitted to the payee (merchant). After that, the payer's credit card information and the payee identification number will be passed to the acquirer. Then, the acquirer sends the information to the credit card issuer in order to approve it. Subsequently, the credit card issuer sends back the response through the acquirer stating whether the transaction is approved or not. The payer (cardholder/customer) will be informed of all their transactions by the card issuer using card statements. The money flow in the credit/debit card e-payment system is shown in Figure 2.

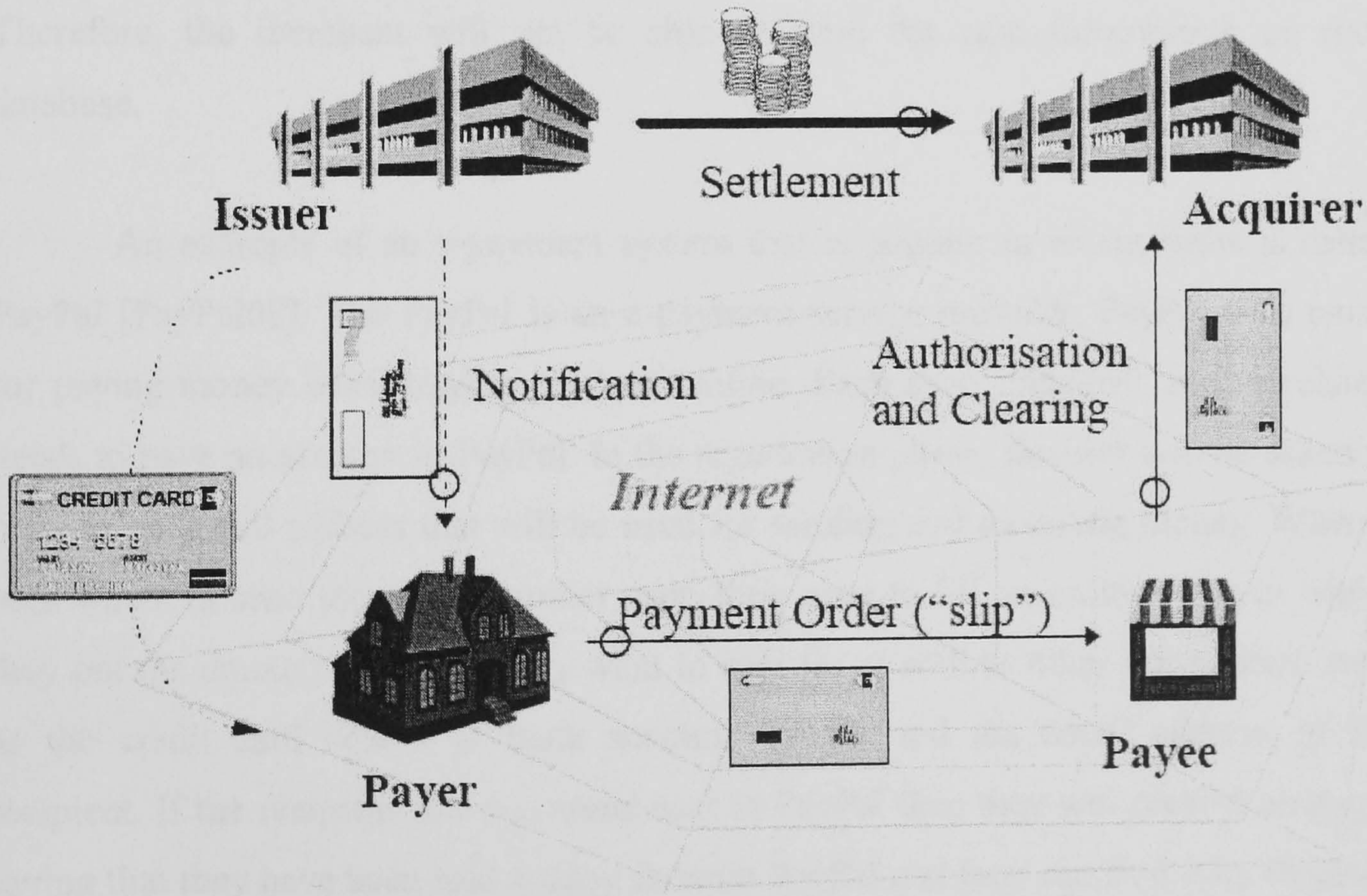

Figure 2: Money flow in credit/debit card-based e-payment systems (Adopted from [AsJaStWa97])

The customer's credit card information may be entered in three options [Wright02]. The first option is to enter the card information using a form that is provided by the merchant. Therefore, the card information will be accessed by the

merchant. The merchant may store the card information in their database. This may result in stealing the customers' cards information by hackers if the database is not very secured. A recent incident of such hacking into customers' cards information stored in the merchant's database, is the one targeted "TK Maxx" stores in 2007 and led to stealing information from 45.7 million payment cards (debit and credit) [BBC08]. The second option is to enter the card information using a form that is provided by a payment service provider. The payment service provider is not the merchant; it is a company that is specialised in providing e-payment services for merchants. An example of the e-payment service provider is "PayPal" which will be discussed later. In the second option, the merchant will not be able to view the card information and hence will not store it in their database. The third option is to use the Secure Electronic Transaction (SET) [SET]. In this option, the customer enters the card information and encrypts it using the public key of the card issuer. Then the customer sends the encrypted card information to the merchant who in turn sends it to the card issuer. The merchant cannot read the card information because it is encrypted. Therefore, the merchant will not be able to store the card information on their database.

An example of an e-payment system that is popular in recent years is called PayPal [PayPal08]. The PayPal is an e-payment service provider. PayPal uses email for paying money when buying products online. Each user (customer and merchant) needs to have an account in PayPal. In the registration phase, the user will be asked to provide an email address that will be used for sending and receiving money. When a user wants to send money to another user, they need to fill an online form in which they put the amount of money they want to transfer as well as other information such as the credit card details or bank account details, and the email address of the recipient. If the recipient is a registered user in PayPal then they will receive an email saying that they have been sent money through PayPal and they can find it by logging-in to their account in PayPal. If the recipient is not a registered user in PayPal, then they need to set up an account in PayPal in order to receive the money. The money in the users' accounts can be used to be transferred to other users. Users can withdraw money from their PayPal accounts by requesting a cheque from PayPal or by requesting a deposit to their bank accounts [Jones01], [Guadamuz03].

As can be seen, using PayPal payer card information (or bank details) will not be shown to the payee [PayPal08]. Hence, it gives more protection for payers from the distrusted payees. This encourages users to pay online since their information will only be dealt with by PayPal.

The major problems for e-payment systems are that of maintaining the security of the card information while sending it over a network and while it is stored in a database [Hsieh01, Wright02]. The security is the major concern for customers, merchants, card issuers and e-payment service providers. Another issue that concerns customers is their privacy when using e-payment systems. That is, some customers prefer to be anonymous to merchants. The anonymity of customers prevents merchants from tracing customers in order to build a profile of their purchases.

2.6. Summary

When a customer buys a product from a merchant, they are, actually, exchanging their items which are the payment that is held by the customer and the product that is held by the merchant. Therefore, at the end of the purchase, the customer wants to have the ordered product and the merchant wants to have the right payment. This is called fair exchange of items.

In this chapter, an introduction to the e-commerce has been presented. E-commerce advantages, disadvantages and limitations have been discussed; and some important issues have been raised such as the security of e-commerce. It has been shown that three things must be secured in order to get secure e-commerce systems. They are the software that is used by a customer, the channel that is used to send and receive information between a customer and a merchant, and finally the server on the merchant side.

The importance of trust between a customer and a merchant has also been discussed. Users of e-commerce need to trust two things. They are the technology by which the e-commerce has been implemented and the partner with whom a user will

interact. The later can be built over time and is difficult to build but easy to lose. E-payment has been discussed as well and the credit card example has been used to show how its information is exchanged between customer, merchant, and the credit card issuer.

One way to build trust between customers and merchants is to use protocols that ensure fair exchange of payments and products.

Chapter Three

3. Fair Exchange Protocols

3.1. Introduction

In e-commerce, protocols are needed to manage the interaction between parties involved in a transaction. E-commerce protocols should be well designed and secured to encourage users to use them for buying and selling products and services. Having such protocols will protect honest users from dishonest users; and as a result more people will use e-commerce systems. There is a set of protocols that can be used for fairly exchanging items between parties. These protocols are termed fair exchange protocols.

In this chapter, a discussion of e-commerce fair exchange protocols is presented. Additionally, some example protocols are analysed and discussed. Dispute resolution will also be identified in the context of e-commerce fair exchange protocols.

A Trusted Third Party (TTP) is a neutral party (entity) used in fair exchange protocols to ensure fairness for all parties involved in the exchange of items. The TTP is assumed to be available, trusted by all parties and will not collude with any party. There maybe more than one TTP involved in any exchange. Therefore, the TTP takes all or some of the following roles [NeZh03]:

- Ensures fairness in the exchange
- Acts as a delivery agent i.e. deliver items to parties
- Acts as an authority that is trusted by all parties

- Resolves disputes
- Validates items and issues certificates

3.2. Fair Exchange Protocols

The problem of fair exchange is concerned with exchanging the items of each party fairly. The parties probably do not know each other and (or) do not trust each other. Therefore, by the end of the exchange each party wants to have the item of the other party; or neither of the parties receives the other party's item. The fair exchange protocols are designed to ensure such fairness. That is, to protect honest parties from the dishonest ones.

There are two types of fairness that fair exchange protocols can ensure [AsScWa97] (1) strong fairness; and (2) weak fairness. If there are two parties involved in the fair exchange protocol then strong and weak fairness can be defined as follows. The fair exchange protocol ensures strong fairness if by the end of executing the fair exchange protocol the two parties will receive each other's items or none do. The fair exchange protocol ensures weak fairness if it either ensures strong fairness or provides the party (who has not received the other party's item) with proof that the other party received their item. This proof can be used later (i.e. not within the protocol) for dispute resolutions.

Fair exchange protocols appear in different contexts. These contexts are certified e-mail, certified delivery, contract signing, and fair purchase. However, the main concern for all of them is how to fairly exchange the items between the parties involved in the exchange. These contexts differ according to the items to be exchanged between parties. The following summarises these contexts [Schunter00].

Certified email

In the certified email fair exchange protocols [NeZhSh05, NeZhBa04, AtMeGo00, DeGoLaWa96, AbGlHoPi02, Micali03] the items to be exchanged between parties are the email and the receipt. That is, the sender of the email wants to

receive a receipt from the receiver of the email to avoid them denying they received the email.

Nenadic et al [NeZhBa04] proposed an optimistic fair certified email protocol that comprises of four messages to be exchanged between a sender (*Party A*) and a receiver (*Party B*) of the email. Their protocol is based on RSA [RiShAd78] and starts by *Party A* sending their first message to *Party B*. The first message includes the hash value of the email and the signature of the *Party A* on the email. On receiving the first message, *Party B* verifies/check if the signature of *Party A* is correct. This verification is done by decrypting *Party A*'s signature using the public key of *Party A*, then comparing the hash value of the email included in the signature with the hash value of the email included in the first message. If this comparison is correct then *Party B* sends the second message to *Party A* which includes their encrypted receipt of the first message. If *Party A* finds that the second message is correct then they send the un-encrypted email to *Party B*. When *Party B* receives the email, they send the fourth message to *Party A* which includes the decryption key to decrypt the encrypted receipt. The TTP is contacted if one party misbehaved.

Certified delivery

The certified delivery fair exchange protocols [NeZhChGo05] are very similar to the certified email fair exchange protocols. The difference is that the received item to be certified is not an email but any digital product or payment. The following is an example protocol for the certified delivery.

Nenadic et al [NeZhChGo05] proposed an optimistic fair exchange protocol that makes use of TTP in the case of disputes. Their protocol is for the exchange of digital product (that is held by a merchant) for its receipt from a customer when they receive the digital product i.e. the customer will send to the merchant a signature that represents a receipt of receiving the digital product. The protocol is called Certified Digital Product Delivery where the merchant sends the digital product to the customer and then the customer sends a confirmation to the merchant stating that the digital product was received.

The exchange phase of the Nenadic et al protocol comprises of four messages to be exchanged between the customer and the merchant. The merchant starts the exchange by sending the first message to the customer. The first message includes the encrypted digital product, the digital product's certificate and the merchant's signature on the digital product. On receiving the first message, the customer verifies it and if satisfied they send the second message to the merchant. The second message includes the customer's encrypted signature which represents the receipt of receiving the digital product. On receiving the second message, the merchant verifies the correctness of the encrypted signature. If it is correct then the merchant sends the third message to the customer. The third message includes the decryption key. On receiving the third message, the customer decrypts the digital product and sends the fourth message to the merchant which includes the decryption key to decrypt the customer's signature. By the end of executing the protocol, the merchant and the customer have fairly exchanged the digital product and the receipt (the customer's signature). The TTP should be contacted to resolve any disputes.

Contract signing

In the contract signing fair exchange protocols [Wang05, AsShWa98, Micali03, PaChSi03] the items to be exchanged between parties are the signature of parties on a contract. That is, if there is a contract to be signed by two parties then each party wants to receive the signed contract by the other party.

Asokan et al [AsShWa98] proposed an optimistic fair contract signing protocol between two parties, lets say A and B. The contract (N) to be signed is previously agreed to between the parties. The protocol comprises of four messages. Party A starts the protocol by, first, generating a random number *ra* and then computing the hash value of *ra*. Then, A signs both the hash value of *ra* and the contract N i.e. A's signature on {N and hash value of *ra*}. Then, A sends the signed message (*me1*) to party B. On receiving *me1*, B verifies it and if it is correct then B generates a random number *rb* and then computes the hash value of *rb*. Then, B signs both *me1* and the hash value of *rb;* and then sends them to A (B sends to A the message *me2*). On receiving *me2*, A verifies it and if it is correct then A sends to B the random number *ra*. On receiving *ra*, B sends to A the random number *rb*. By the end of executing the

protocol both A and B have a signed contract that includes (*me1, ra, me2, rb*). If anything went wrong, the TTP can be contacted to resolve any disputes.

Fair purchase

The fair purchase exchange protocols [Schunter00, AsScWa97, Ketchpel95, RaRaNa00, RayRay01, RayRay00a, RaRaNa05, ZhMaMa06, DeChPh07, AlarajMunro07a, AlarajMunro07b, AlarajMunro08a, AlarajMunro08b] are concerned with the exchange of digital products and payments. That is, one party (such as the merchant) has the digital product and the other party (such as the customer) has the payment. The fair purchase exchange protocols ensure the fair exchange of a digital product and a payment between the two parties.

The focus in this thesis is on fair exchange protocols that are for the exchange of payments and digital products (i.e. fair purchase exchange protocols) between customers and merchants.

3.3. Types of Fair Exchange Protocols

The fair exchange protocols (whether they are for certified email, certified delivery, contract signing or fair purchase) can be divided into two types according to the involvement of the TTP. The first type is the protocols that do not involve the TTP and the second type is the protocols that involve the TTP [RaRaNa05, Schunter00, and Nenadic05]. The two types are discussed in the following sections.

3.3.1. Protocols that do not involve a TTP

This type of protocol implements the exchange of items between two parties directly without any involvement of any other parties.

3.3.1.1. Gradual Exchange Protocols

The gradual exchange protocols [Blum83, EvGoLe85] are applicable to items that are easily decomposable into the same number of parts. This type of protocol is based on having a number of rounds to complete the exchange of items between the parties. In each round, each party sends a part of its item to the other party. The number of rounds equals the number of parts into which the items are divided. The sending of these parts will continue until each party receives the other party's item. Therefore, in each round each party sends a part of its item and also receives a part of the other party's item, see Figure 3. At any given moment the number of parts received by both parties is approximately the same [ShmMit99].

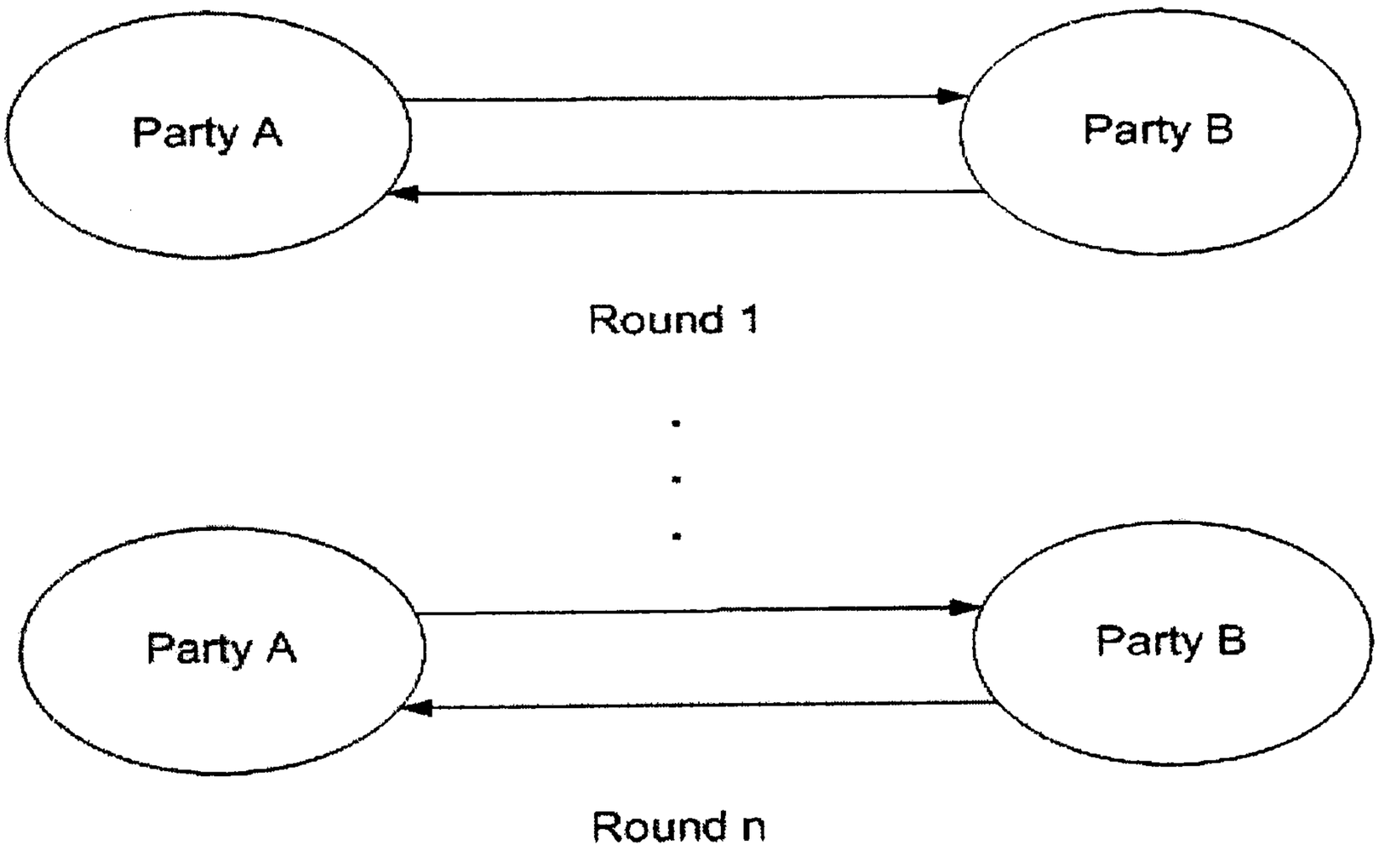

Figure 3: Gradual exchange protocols

One problem of this type of protocol is that many rounds are required to complete the exchange. Therefore, when the number of rounds is too large, there will be a possible load on the communication channel to be used for the exchange of the two items between the two parties. The items to be exchanged between the two parties are assumed to have the same size [KrMaZh02]. Therefore, items of different sizes are not supported in this type of protocol. Another problem is that this type of protocol does not involve a TTP, it is impossible to ensure fairness without recourse to a TTP

to resolve disputes [PaVoGa03]. To explain the reason why this type of protocol will not ensure fairness, consider the following scenario.

The first party (A) sends the first part of its item to the other party (B). Then B sends its first part to A. Then the exchange of parts will continue. If A sends its last part to B but B disappears before it sends its last part to A, then B will have the A's complete item but A will not have the complete item of B. So, there is a possibility that the fair exchange protocol will end in an unfair situation [PaVoGa03].

In the case of exchanging a payment for a digital product without any involvement of a TTP then it can be described as follows [Schunter00]. A customer sends a small amount of payment to a merchant and in turn the merchant sends a small part of the digital product to the customer. This process continues until both the customer and the merchant receive the complete items from each other i.e. the customer receives the complete digital product and the merchant receives the complete payment. There is a possibility, however, that the merchant does not send the last part of the digital product after receiving the last part of the payment. Hence, the fairness is not ensured.

This type of the exchange (i.e. exchanging small parts of payment for small parts of digital product) can be performed using the micro-payment systems [SoKoTa02] [Schunter00].

Jakobsson [Jakobsson95] proposed a new approach for fair exchange of a payment and a digital product without any involvement of a TTP. The protocol is based on splitting the payment into two parts. The two parts must be combined to be used i.e. the first part can not be used without the second part; and also the second part can not be used without the first part.

The protocol of Jakobsson [Jakobsson95] starts by the customer sending the first part of the payment to the merchant. On receiving the first part, the merchant sends the digital product to the customer. On receiving the digital product the customer sends the second part of the payment to the merchant. When the merchant receives the second part of the payment they combined it with the first part to

construct the whole payment. This protocol does not ensure fairness because the customer may disappear before sending the second part of the payment. That is, after the customer receives the digital product they do not send the second part of the payment. Therefore, the fairness is not ensured as the customer received the digital product but the merchant did not receive the second part of the payment to be able to construct the whole payment.

3.3.2. Protocols that involve a TTP

There are three kinds of this type of protocol according to how they use the TTP. These kinds are [KrMaZh02]:

(1) Protocols that are based on inline TTP,

(2) Protocols that are based on online TTP, and

(3) Protocols that are based on offline TTP

The three kinds of protocols will be discussed separately in the following sections.

3.3.2.1. Protocols that are based on inline TTP

The protocols based on inline TTP (such as [BuPf90, RaRaNa00, ZhGo96a]) use the TTP for delivering the exchanged items to the parties. That is, each party sends their item to the TTP, and then the TTP will verify the items and then will deliver them to the parties. For example, if the parties are a customer and a merchant; and the items to be exchanged are a payment (held by the customer) and a digital product (held by the merchant) then the way in which the protocol will work is as follows. The merchant sends the digital product to the TTP and the customer also sends the payment to the TTP. Then, the TTP verifies the received items then it sends the payment to the merchant and the digital product to the customer. Figure 4 shows a model of the fair exchange protocols that use inline TTP.

The TTP is actively involved in the exchange process. The active involvement of the TTP in this kind of protocol ensures the fair exchange of items between the

parties. The inline TTP based protocols do not normally require direct communication between customer and merchant.

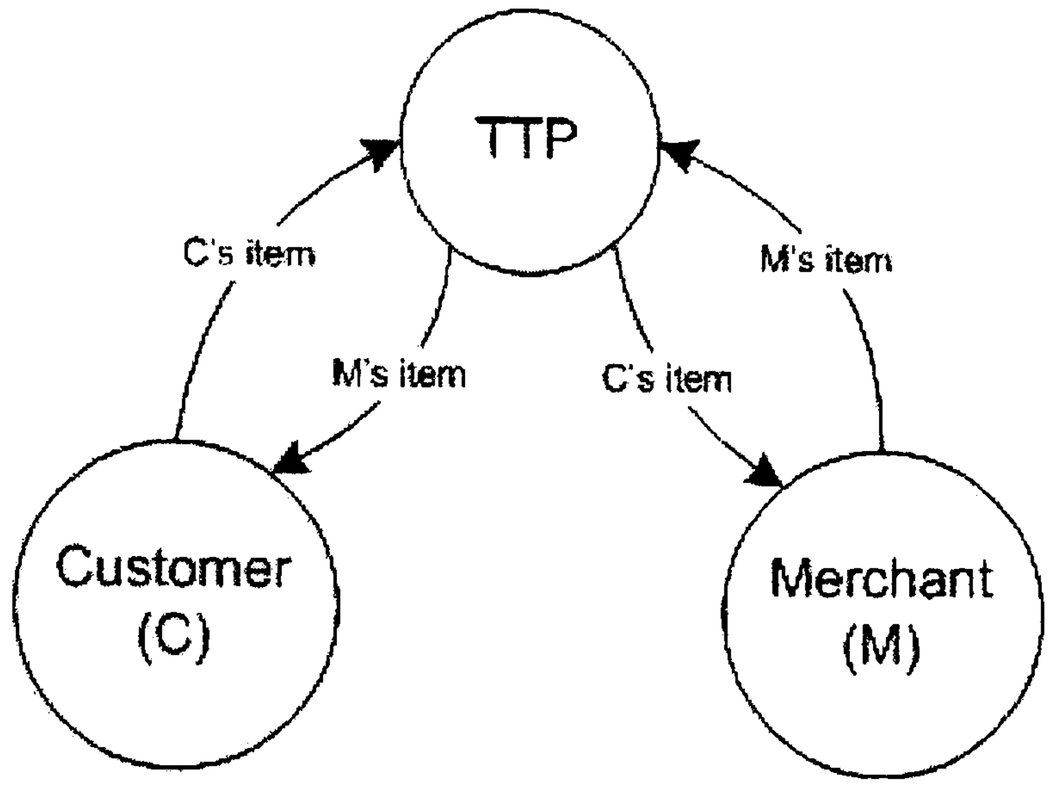

Figure 4: Inline TTP based fair exchange model

Although the protocols that use inline TTP ensure fairness for all parties as the TTP will deliver the items to all parties, they have some disadvantages. Firstly, protocols that use inline TTP require that the TTP be available during the execution of the protocol which will result in extra costs as the TTP is expensive to run [Micali03]. Secondly, this kind of protocols can lead to performance problems as the TTP may become a source of a communication bottleneck [LiNiJa00, RaRaNa05, AsScWa97 and ShmMit99]. This is because the exchange of items will be through the TTP. Thirdly, when the TTP crashes, the protocol will not be executed and the parties will not be able to receive the items that they want. Finally, the TTP will be the main target of attacks [LiNiJa00].

Burk and Pfitzmann Protocol

Burk and Pfitzmann [BuPf90] proposed a fair exchange protocol that is based on inline TTP. This protocol lets the parties who are willing to exchange their items (i.e. the parties are the customer and the merchant; and the items are the payment and the digital product) to agree on the items to be exchanged. Then, each party contacts the TTP to inform them of the agreement that they have with each other. The customer

then sends the payment to the TTP. On receiving the payment, the TTP checks whether it is according to the agreement that the parties agreed on or not. If it is according to the agreed price then the TTP sends to the merchant informing them of receiving the correct payment from the customer. The merchant then sends the digital product to the TTP. On receiving the digital product from the merchant, the TTP checks whether it is according to the agreement that the parties agreed on or not. If the digital product is according to the agreement then the TTP sends the payment to the merchant and the digital product to the customer.

Ray Protocol

Ray et al [RaRaNa00] proposed a fair exchange protocol that is based on the theory of cross validation [RayZhang08]. The cross validation theory states that [RayZhang08] "*if a message is encrypted with the product key of two compatible keys and another message is encrypted with either of the two compatible keys and the two encrypted messages compare, then the two original un-encrypted messages must also compare if a message is encrypted with the product key of two compatible keys and another message is encrypted with either of the two compatible keys and the two encrypted messages compare, then the two original un-encrypted messages must also compare*". Therefore, using the cross validation theory, the customer will be able to validate the encrypted digital product to be received from the merchant without decrypting it.

The protocol is for the exchange of a digital product (held by a merchant) and a payment (held by a customer). The protocol is based on the use of inline TTP. The merchant advertises their digital products in the TTP's catalogue. The advertisement process is as follows. The merchant generates a key pair $\boldsymbol{K1}$ (which is for encryption) and $\boldsymbol{K1}^{-1}$ (which is for decryption); and then sends the key pair and the digital product to the TTP. The TTP encrypts the digital product with $\boldsymbol{K1}$ and then advertises it along with a description for the digital product in its catalogue (the same thing is done for each digital product that the merchant wants to exchange for a payment i.e. to sell). A customer who is interested in buying a digital product (i.e. exchanging a digital product for a payment) searches the TTP's catalogue and if they are interested in one

of the digital products then they download an encrypted digital product from the TTP. The exchange of the payment and the digital product between the customer and the merchant will then take place.

The exchange phase of the protocol consists of six messages to be exchanged between the customer, the merchant, and the TTP. The exchange phase starts by the customer sending the first message to the merchant which includes the purchase order. On receiving the first message from the customer, the merchant validates it and if satisfied then the merchant does the following. The merchant creates another key pair $K2$ and $K2^{-1}$. This key pair is mathematically related to the first pair (that is sent to the TTP). Then, the merchant sends the second message to the TTP. The second message includes the merchant's signature on the purchase order, and $K2^{-1}$. In addition, the merchant sends the third message to the customer. The third message includes the encrypted digital product and the merchant's signature on the encrypted digital product. On receiving the third message, the customer compares the encrypted digital product that was received in the third message with the encrypted digital product that was downloaded from the TTP. If the two are compared then the customer sends the fourth message to the TTP. The fourth message includes the purchase order, the customer's signature on the encrypted digital product, and a signed payment. On receiving the fourth message, the TTP compares the hash value of the purchase order received from the customer (in the fourth message) with the hash value of the purchase order received from the merchant (in the second message). If they are compared then the TTP contacts the customer's banks for validating the payment. If the payment is valid, then the TTP sends to customer in the fifth message the decryption key for the encrypted digital product. Additionally, the TTP sends to the merchant the payment in the sixth message. If there are any disputes then the TTP is contacted.

As can be seen in this protocol [RaRaNa00], although the merchant sends the encrypted digital product to the customer in message three (i.e. there is a direct communication between the customer and the merchant), the inline TTP is used for delivering the items to the parties. That is, the inline TTP is used to deliver (1) the decryption key of the encrypted digital product to the customer (the encrypted digital

product was received from the merchant by the customer in message three), and (2) the payment to the merchant.

3.3.2.2. Protocols that are based on online TTP

The protocols that are based on online TTP (such as [DeChPh07, ZhMaMa06, ZhGo96b, AbGlHoPi02, and ZhSh96]) reduce the use of the TTP. In this kind of protocol, the TTP will be used during the protocol run but its use is not for delivering the parties' items. Rather, its use can be for validating an item, generating and/or storing evidence of a transaction [Nenadic05].

The following example illustrates the use of online TTP in the fair exchange protocols. If the items to be exchanged are a payment and a digital product between a customer and a merchant and the customer is the one who starts the exchange then after the merchant receives the payment from the customer then the merchant will validate it with the TTP (a bank for example) before they send the digital product to the customer. Therefore, the TTP must be online for the exchange to be completed. The TTP should be contacted if there is any dispute. Figure 5 shows a model of a fair exchange protocol that is based on online TTP.

Although this kind of protocol reduces the involvement of the TTP, it requires the existence of the TTP during the exchange of items. This is seen as a disadvantage because the TTP may become a source of a communication bottleneck. Additionally, the TTP might be the main target of attacks.

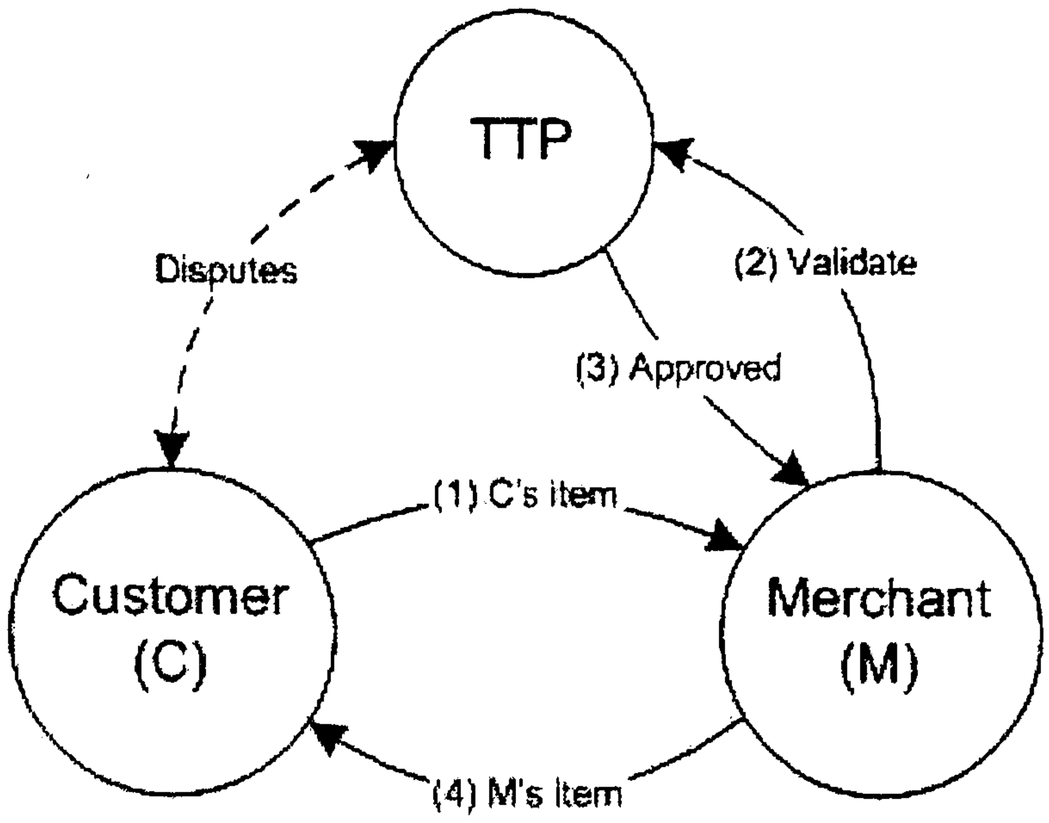

Figure 5: Online TTP based fair exchange model

Zhang-Q Protocol

Zhang et al [ZhMaMa06] proposed a fair exchange protocol that involves online TTP. The proposed protocol is for the exchange of a payment and a product which might be a physical product. Therefore, the payment is sent online (i.e. via the protocol messages) from the customer to the merchant whereas the product is delivered to the customer using a delivery agent i.e. the delivery of the product is not using electronic means. The protocol is based on the theory of cross validation [RayZhang08].

The customer starts the protocol by requesting a product from the merchant. Then, the merchant sends the invoice to the customer. If the customer is satisfied with the invoice then they, first, send an encrypted payment to the merchant and, second, send the encrypted payment to the TTP (the bank). The merchant is assumed to be able to download the encrypted payment (that was sent by the customer to the TTP) from the TTP (the bank). The merchant then compares the two encrypted payments (i.e. the one received from the customer and the one downloaded from the TTP). If they are compared then the merchant is sure that the encrypted payment is correct. After verifying the encrypted payment, the merchant sends the product to the delivery

agent. Then the customer collects the product from the delivery agent. When the customer finds that the product is the same as they expected, they send the decryption key to the merchant who will then decrypt the encrypted payment.

Devane et al Protocol

Devane et al [DeChPh07] (will be called hereafter Devane protocol) proposed a fair exchange protocol for the purchase of digital products over the Internet. The protocol ensures fair exchange of a digital product for a payment between a merchant and a customer. However, the protocol involves an online TTP which is a bank in which both a customer and a merchant have accounts. The exchange phase of Devane's protocol comprises of seven messages to be exchanged between customer, merchant, and the bank.

The customer initiates the protocol by sending the first message that includes a signed purchase request by the customer. On receiving the first message, the merchant verifies it and if satisfied sends the second message to the customer. The second message includes a signed invoice in addition to the encrypted digital product. On receiving the second message, the customer verifies it and checks the signed invoice. If the customer is satisfied then they send the third message to the merchant. The third message includes a signed payment. On receiving the third message, the merchant verifies it and if satisfied then they send the fourth message to the bank. The fourth message includes the decryption key for the digital product and the message three that was received from the customer and singed by the merchant (i.e. the merchant signs the signed payment by the customer and sends it to the bank). On receiving the fourth message, the bank verifies it. If message four is correctly verified then the bank sends the fifth message to the merchant. The fifth message includes the bank's signature on the signed payment and the decryption key. On receiving the fifth message, the merchant forwards it to the customer. On receiving the sixth message, the customer gets the decryption key and decrypts the encrypted digital product that was received in message number two. If the customer found that the decrypted digital product is the one they specified in message one then the customer sends message seven to the bank. Message seven includes customer's acknowledgement about the digital product. On

receiving the seventh message, the bank transfers the payment from the customer's account to the merchant's account.

3.3.2.3. Protocols that are based on offline TTP

The protocols that are based on offline TTP (such as [AsScWa97, RaRaNa05, RayRay00a, ZhShMeAs06, NeZhBa04, NeZhChGo05, AsShWa98, KrMa01]) allow the parties to directly exchange their items without any involvement of the TTP unless one party misbehaves. This kind of protocols is also called in the literature "Optimistic fair exchange protocols" [EzhShr05, RaRaNa05, AsScWa97, RayRay02, RayRay00a, Micali03, and Schunter00]. This kind of protocols will be referred hereafter as optimistic fair exchange protocols.

The following example illustrates how the optimistic fair exchange protocols work. If the items to be exchanged are a payment and a digital product between a customer and a merchant then the protocol will work as follows. The two parties directly exchange the payment and the digital product; and if one party misbehaves the TTP will be invoked to resolve the disputes. Figure 6 shows a model of a fair exchange protocol that is based on offline TTP (optimistic fair exchange protocols).

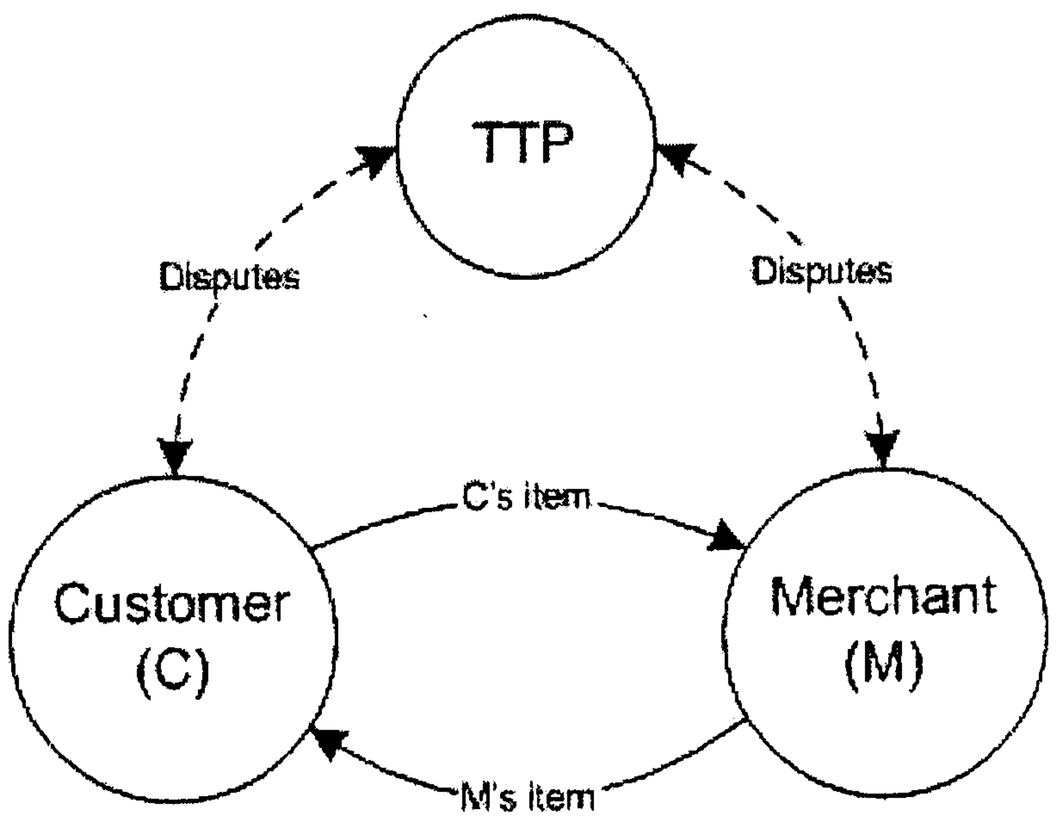

Figure 6: Offline TTP based fair exchange model

The optimistic fair exchange protocol reduces the problems of using inline TTP and online TTP. This is because optimistic fair exchange protocols use the TTP very rarely and do not involve it in every exchange. Therefore, the problem of having the TTP as a source of a communication bottleneck that exists in the inline and online TTP based protocols is reduced. This is because the optimistic fair exchange protocols reduce the use of the TTP and let the parties exchange their items directly. Another advantage is that the problem of having the TTP as a single point of failure is reduced as the parties will not need the TTP unless something goes wrong. Additionally, as the TTP will not be actively involved the cost of running the TTP will be reduced.

Ray et al Protocol

Ray et al [RaRaNa05] proposed an optimistic fair exchange protocol. The protocol is based on the idea of the theory of cross validation [RayZhang08].

Some steps take place before the protocol starts. A merchant (M) needs to register with TTP. The TTP generates the key pair $\boldsymbol{K}_{MI}$ and $\boldsymbol{K}_{MI}^{-1}$. The TTP then provides M with $\boldsymbol{K}_{MI}$ and keeps $\boldsymbol{K}_{MI}^{-1}$ with itself. A customer (C) needs to have an account in a bank. The bank generates the key pairs $\boldsymbol{K}_{CI}$ and $\boldsymbol{K}_{CI}^{-1}$. The bank then provides C with $\boldsymbol{K}_{CI}$ and keeps $\boldsymbol{K}_{CI}^{-1}$ with itself. The merchant needs to send the digital product, its description and its price to the TTP. The TTP encrypts the digital product using the key $\boldsymbol{K}_{MI}$ and then advertises it on its website i.e. the TTP's website. The customer visits the website and downloads the encrypted digital product from the TTP to be used for validating the digital product that they will receive from the merchant. If the customer is interested in the digital product then they can contact the merchant and starts the protocol as follows.

The actual interaction between the customer and the merchant in the Ray et al protocol [RaRaNa05] consists of four messages. The four messages can be summarised as follows.

The customer (C) sends to the merchant (M) the first message that includes (1) the purchase order and (2) the payment that is encrypted with the product key of $\boldsymbol{K}_{CI}$ x

K_{C2}. On receiving the first message, M verifies it and if it is correct then M sends the second message to C which includes the digital product that is encrypted with the product key of K_{M1} x K_{M2}. On receiving the second message, C compares the encrypted digital product that was received from the TTP with the encrypted digital product that is included in the second message. If the two are compared then C can be sure that the un-encrypted digital products will be compared as well (in here the cross validation theory is applied). Therefore, if the two encrypted digital products are compared and C is still interested in the exchange then C sends the third message to M which includes the decryption key for the encrypted payment. Finally, M sends the fourth message to C which includes the decryption key of the encrypted digital product. If there is any dispute, C can contact the TTP.

As can be seen in this protocol, the TTP will have a copy of all digital products that M wants to sell. Therefore, this will result in extra storage in the TTP side as well as extra security assurance. Another problem with this protocol is that the customer needs to download the digital product twice (one from the TTP before the exchange and then from the merchant). Therefore, the twice download of the digital product will be a communication overhead.

Asokan Protocol

Asokan et al [AsScWa97] proposed an optimistic fair exchange protocol i.e. the TTP will only be involved in the case of disputes. If the parties interested in the exchange want to exchange a payment and a digital product, the protocol comprises of four messages which can be explained as follows.

A merchant (M) and a customer (C) promise each other to exchange their items (i.e. the payment and the digital product). The agreement between C and M is represented in messages 1 and 2. That is, C sends to M the amount of payment that they are willing to pay; and then M will send to C the description of the digital product. Then, C sends the payment to M in message 3. On receiving message 3, M sends the digital product to C in message 4.

If C found that the digital product is not the same as described in the agreement, then C contacts the TTP. The TTP verifies C's request, if it is valid then the TTP cancels the payment. If the TTP could not cancel the payment then the TTP can provide affidavit proof to be used in the court to resolve the disputes.

This protocol is not the best solution for fairly exchanging digital product and payment because there is a possibility that the TTP could not cancel the payment in case of dispute which will result in not resolving disputes automatically. Imagining that the customer and the merchant are located in different parts of the world and the TTP can not resolve the dispute automatically; then, if something goes wrong i.e. the digital product was not the same as expected, then the customer needs to travel to the country of the merchant in order to go to court. Therefore, the protocol used for fair exchange should ensure fairness for all parties and in the case of dispute, it should provide online and automated dispute resolution as the parties might be located in different parts of the world.

One difference between Asokan et al's protocol [AsScWa97] and Ray et al protocol [RaRaNa05] is that Ray et al protocol allows each party to verify the correctness of the item that they are going to receive before they receive it; whereas in Asokan et al's protocol [AsScWa97], parties are able to validate the item only after receiving it. Moreover, Ray et al's protocol provides automated dispute resolution, while there is a possibility in Asokan et al's protocol that the TTP provides proof for the party who has dispute which can be used in a court of law after the protocol finishes.

Zhang Protocol

Zhang et al [ZhShMeAs06] (will be called hereafter Zhang protocol) proposed an optimistic fair exchange protocol. Their protocol is for exchanging two valuable documents (the two documents can be a payment and a digital product) between two parties; *party A* and *party B* (the two parties can be a customer and a merchant).

The exchange of items in Zhang's protocol comprises of four messages to be exchanged between *party A* and *party B*. *Party A* starts the exchange by sending the first message to *party B*. The first message includes the encrypted document of *Party A* and the encrypted key that decrypts the decrypted document. On receiving the first message, *party B* verifies it and if satisfied then they send the second message to *party A*. The second message includes the encrypted document of *party B* and the encrypted key that decrypts it. On receiving the second message, *party A* verifies it and if satisfied then they send the decryption key to *party B* in the third message. When *party B* receives the decryption key, they use it to decrypt the encrypted document that was received in the first message. Then *party B* sends to *party A* the fourth message which includes the decryption key. Finally, when *party A* receives the decryption key, they use it to decrypt the encrypted document that was received in the second message. If there is any dispute, the TTP will be contacted to resolve it.

3.4. Dispute resolution

It is likely to have disputes when buying and selling products. This is because some customers may find that the product they have bought from a merchant is not the same as they wanted or the product has some problems. In the case of online purchase the probability of having disputes is high as a customer or a merchant might not be trustworthy and also because it is not possible for a customer to feel or try the product that they are going to pay for.

Disputes can be resolved formally in a court. There are however some alternatives for resolving disputes without going to a court of law. They are called Alternative Dispute Resolution (ADR). Arbitration and mediation are examples of ADR.

In e-commerce the need to resolve disputes online is vital because customers and merchants might be located in different parts of the world and come under different legal systems. When the dispute resolution is done online, it is called Online Dispute Resolution (ODR). Many techniques are available for ODR [Hörnle02], [AlfuraihSnow05] and [GalKov06]:

Arbitration

Arbitration means having a neutral third party who collects information from the two parties who have a dispute. Then the third party makes a decision that is intended to be binding.

Evaluation

Evaluation is the same as arbitration but the decision made by the neutral third party is a recommendation rather than binding.

Mediation

Mediation means to have a third party who helps the two parties to reach an agreement. So, it is not the same as arbitration that lets the third party make the decision.

Automated negotiation

Automated negotiation is used to resolve disputes that are related to monetary amounts. It is based on having blind bids, in which the parties enter their suggestions to resolve the dispute. Each party does not know what the other party has offered. Finally, a computer program proposes a resolution when the offers of the parties are sufficiently close.

Mock Trial

Mock Trial is based on having a jury of peers who volunteer for making a decision on a case of dispute using web-based platform. The decision made by the jury is not binding.

Complaint Assistance

A Complaint Assistance is a tool that helps the person who writes a dispute. Some tools have interactive forms that the user needs to complete. This kind of tools provides advice, similar cases that have been assisted by the tool.

Credit Card Charge Back

This technique allows credit card issuers to act as a third party between a customer and a merchant. So, the credit card issuer will study the dispute and then if the customer dispute is accepted, it will give the customer's money back.

The above techniques are done online but they are not fully automated i.e. they require human involvement.

There are two types of fair exchange protocols that include dispute resolution [RaRaNa05]. The first type is the protocols that include dispute resolution during the execution of the protocol, and the second type is the protocols that do not include dispute resolution steps during the execution of the protocol.

For protocols that include dispute resolution during the execution of the protocol, the protocol includes a mechanism that ensures a resolution for any dispute. In these protocols a TTP is normally the one who receive the dispute requests and resolve them in a fair manner. Examples of this type of protocols include [RayRay00a, NeZhBa04, NeZhChGo05, ZhMaMa06, RaRaNa05, ZhShMeAs06, DeChPh07, and RaRaNa00].

The protocols that do not include dispute resolution during the execution of the protocol are based on storing evidences to be used in the case of a dispute. The evidence is used after the execution of the fair exchange. The resolution might be done using the ODR techniques or in a court of law. The supplied evidences will help resolving disputes. Examples of this type are [AsScWa97, CoTySi95].

To use the ODR techniques in the fair exchange protocols, the protocol should provide a way of storing evidence that can be used in one of the ODR techniques in order to resolve disputes. However, this way of resolving disputes (using these techniques) is called after the fact solution [RaRaNa05] in which the involved parties might disappear. Therefore, the dispute resolution should be part of the fair exchange protocol and not after its execution. To do so, there should be a way in which the disputes can be resolved automatically before the end of protocol execution and that is through the use of the TTP.

It is clear that the protocols that include dispute resolution during the execution of the protocol are better than the protocols that do not. This is because normally the parties involved in the exchange of items are located in different parts of the world. So, it will be difficult to go to a court of law in the other party's country. Additionally, one party may disappear before the completion of the exchange. Therefore, the nature of e-commerce makes it difficult to resolve disputes after the execution of the fair exchange protocol. Hence, the first type is more suitable for e-commerce.

There should be a way in which the possibility of having disputes should be reduced in order to reduce the need for dispute resolution in the first place. That is, before a customer and a merchant exchange their items, they should be sure that they are going to receive what they want from each other. By reducing the possibility of having disputes, the number of messages needed to resolve disputes will be reduced as well. As a result, the load on the communication channel between the customer and the TTP and between the merchant and the TTP will be reduced.

3.5. Summary

Fair exchange protocols have been discussed in this chapter. The protocol used by customer and merchant to exchange their items must ensure fairness for both parties. Fairness in the context of the fair exchange protocols between two parties is defined as at the end of the protocol execution, each party participating in the exchange should have the item of the other party or none do.

Many fair exchange protocols have been discussed. They are categorised into two types, the ones that do not use a TTP and the ones that use a TTP. The former let the parties exchange their items part by part until the whole items are exchanged. The latter is divided into three kinds. The first kind is the one that uses the TTP for delivering the items to the parties i.e. inline TTP based fair exchange protocols. The second kind is the one that uses online TTP where the load on the TTP is reduced. The third kind is the one that uses offline TTP (optimistic fair exchange protocols). In the optimistic fair exchange protocols the TTP is only used if something goes wrong

during the execution of the protocol. Examples of these protocols are summarised in Table 1.

Dispute resolution has been covered and some of its techniques were presented. The resolution of dispute has also been discussed in the context of fair exchange protocols. It has been shown that the dispute resolution should be part of the fair exchange protocol to protect a party if the other party misbehaves. Therefore, there is a need to have a fair exchange protocol that minimises disputes and also incorporates automated dispute resolution.

As can be seen in Table 1, the load on the TTP is low in the protocols that are based on offline TTP. This is because the TTP will only be contacted in the case of disputes. Hence, the efficiency in this type of protocol is high. In the protocols based on inline TTP, the load on the TTP is very high because the TTP will receive the parties' items, validate them, and if they are correct then TTP will send them to the parties. Therefore, the TTP does everything even if the parties are honest. Hence, the efficiency in the protocols that are based inline TTP is low. The load on the TTP in the protocols that are based on online TTP is high as the TTP will be used during the exchange of items. Hence, the efficiency of this type of protocols is medium. The number of messages needed for the exchange of items in the presented fair exchange protocols is at least four messages.

Protocol	Items to be exchanged	Use TTP	TTP Type	Load on TTP	Efficiency	# messages (exchange phase)	Type of fairness
[NeZhBa04]	e-mail and receipt	Yes	Offline	Low	High	4	Strong
[NeZhChGo05]	Digital product and receipt	Yes	Offline	Low	High	4	Strong
[AsShWa98]	Signatures on a contract	Yes	Offline	Low	High	4	Strong
[AsScWa97]	Payment and digital product	Yes	Offline	Low	High	4	Weak or Strong
[Jakobsson95]	Payment and digital product	No	N/A	N/A	High	Not specified	Fairness is not ensured
[BuPf90]	Payment and digital product	Yes	Inline	Very high	Low	7	Strong
[ZhMaMa06]	Payment and a product (digital or physical)	Yes	Online	High	Medium	7 + physical delivery and collection	Strong
[RaRaNa05]	Payment and digital product	Yes	Offline	Low	High	4	Strong
[ZhShMeAs06]	Two digital documents	Yes	Offline	Low	High	4	Strong
[DeChPh07]	Payment and digital product	Yes	Online	High	Medium	7	Strong
[RaRaNa00]	Payment and digital product	Yes	Inline	Very high	Low	6	Strong

Table 1: Comparison of fair exchange protocols

The best type of fair exchange protocols is the one that is based on offline TTP (which is also called optimistic fair exchange protocols) because the load on the TTP is low and hence its efficiency is high. However, the way in which the current optimistic fair exchange protocols work is to let the parties exchange their encrypted items and if they are satisfied then they exchange the decryption keys to decrypt the encrypted items. So, using this way the number of messages needed will be at least

four messages. Therefore, a technique is needed to reduce the number of messages. This technique is originated in this thesis.

Chapter Four

4. Concepts and Assumptions

4.1. Introduction

In this chapter, some of the cryptographic concepts will be discussed and the assumptions of the proposed protocols will be presented.

4.2. Cryptographic concepts

The protocols in this thesis rely on cryptography as the underlying security model. This thesis is not about cryptography but this section has been included to show some of the background required.

4.2.1. Symmetric-key Cryptography

The symmetric-key cryptography has different names such as secret-key cryptography and conventional cryptography [Stallings99]. The idea of the symmetric-key cryptography is to use one key that is used by a sender and a receiver of a message to do the encryption and the decryptions. That is, the sender and the receiver will agree on a symmetric-key to be used by both of them. The sender encrypts a message using the agreed symmetric-key and then sends the encrypted message to the receiver. On receiving the encrypted message, the receiver decrypts it using the same key used to encrypt the message i.e. the symmetric-key.

4.2.2. Public-key Cryptography

The public-key cryptography is sometimes referred to as asymmetric-key cryptography. The idea of the public-key cryptography [Schneier96] is based on having two keys that are mathematically related to each other. One of the two keys is called a public key and the other is called a private key. The public key can be known to anyone whereas the private key is only known to the party who the key belongs to. When using public-key cryptography, the sender first gets the public key of the receiver (the public keys of parties are publicly available) then second encrypts the message using this public key and then finally sends the encrypted message to the receiver. On receiving the encrypted message, the receiver uses their private key to decrypt the encrypted message. Many algorithms are available for public-key cryptography but only the RSA will be discussed as it is used in this thesis.

4.2.2.1. RSA

RSA [RiShAd78] refers to its inventors Ronald **R**ivest, Adi **S**hamir, and Leonard **A**dleman. RSA is an algorithm for public-key cryptography. Each party has their own public and private RSA keys. The public key is known to anyone whereas the private key is known only to the party who owns it. The generation of RSA public and private keys [Stallings99] is as follows, a party:

- selects two large random prime numbers p and q
- calculates $n = p * q$
- calculates $\varphi(n) = (p-1) * (q-1)$
- selects an integer e that satisfies: $1 < e < \varphi(n)$ and e is relatively prime to $\varphi(n)$
- calculate $d = e^{-1} \bmod \varphi(n)$
- the RSA public key $pk = (e,n)$
- the RSA private key $sk = (d,n)$

The encryption using RSA is performed as follows [RiShAd78]. The message M to be encrypted using RSA has to be less than n i.e. $(M < n)$. If the size of M is

greater than n then M is divided in blocks and each block is less than n. Therefore, for each block the following is computed:

$$C = M^e \pmod{n}$$

With regard to the decryption using RSA, to decrypt the encrypted message C, the following is computed [RiShAd78]:

$$M = C^d \pmod{n}$$

4.2.3. Hash Function

The hash function is a function that takes an input of any length and it returns a fixed length output [Schneier96]. The output of the hash function is called the hash value. The hash function is called one-way hash function if it computationally infeasible to get the original input from the hash value [Schneier96]. That is, when the hash value, say H, is computed for a message M, then it is not possible to get M from the hash value H. The hash function is called strong-collision-resistance hash function [Schneier96] if it is computationally infeasible to have the same hash values for different messages. That is, when we have two different messages *M1* and *M2* and the hash values for the two messages are *H1* and *H2*, respectively, then $H1 \neq H2$.

4.2.4. Digital signature

Digital signature is a simulation of the handwritten signature. The digital signature is a development of the public-key cryptography [Stallings99]. That is, for generating a digital signature a private key is needed whereas for verifying the signature the public key is needed. As stated in the public-key cryptography section, each party has a private key and a public key that are mathematically related to each other. To encrypt a message a sender uses the public key of the receiving party whereas, for the receiver of the encrypted message, the private key is used to decrypt the message. On the other hand, for digital signature, the sender uses their private key

to digitally sign the message and sends it to the receiver. On receiving the message, the receiver uses the public key of the sender to verify that the message is indeed signed by the sender; this is because the private key is only kept by its owner whereas the public key is available to anyone. Many algorithms of digital signature are available but only RSA signature will be presented because it is used in this thesis.

4.2.4.1. RSA Signature

The RSA public key, $pk = (e,n)$, and the private key, $sk = (d,n)$, are used in RSA signature. To generate the RSA signature [RiShAd78], the hash value of a message to be signed is computed then the hash value is encrypted using the private key (sk) of the signer. Then the message is sent to a receiver along with the signature. That is, the signature generation by the sender for a message M is as follows:

$$\text{Sig (M)} = (h(M))^{d} \pmod{n}$$

To verify the signature [RiShAd78]: on receiving the message and the signature, the receiver uses the public key, pk, of the signer to decrypt the singed hash value. Then the receiver computes the hash value of the message. The two hash values (the one computed by the receiver and the one decrypted using the public key of the signer) are compared. If they are compared then the signature is verified correctly. The operation of decrypting the signature is as follows, H refers to the hash value of the signed message:

$$H = (Sig(M))^{e} \pmod{n}$$

4.3. Assumptions

The following are the assumptions used in all the proposed protocols:

- TTP is trusted by all parties and it will not collude with any other party

- The work in this thesis is not concern about how to match the description of D with the one in D-Cert; rather it assumes that a function with the CA will do it
- All parties use the same algorithms for performing encryption, decryption, signing, and computing the hash value
- Each party *x* (*x* ∈ c, m, t, ca, cb) has an RSA key pair (public *pkx* and private *skx*). The *pkx* is certified by a recognised certificate authority and is known to all other parties
- The communication channels to be used in the pre-exchange phase of all the proposed protocols are secure
- The communication channel between TTP and C is resilient as is also the communication channel between TTP and M. A resilient channel means that all sent messages will be received by their intended recipients [RaRaNa05]
- The payment used in the protocols is in the form of a payment order that is issued and signed by a customer's bank and it specifies the amount of payment to be paid, the payee and the payer
- Double spending of the same payment is assumed to be detected and therefore will not occur
- C and M have already negotiated the items to be exchanged (i.e. the digital product and the amount of the payment) before the protocol starts. Therefore, the proposed protocols are neither negotiation systems nor payment systems. They concern the actual exchange of a payment for a digital product between C and M
- C and M will agree on the TTP to be used in both the pre-exchange phase and for dispute resolution before they start the protocol
- Parties involved in the protocols will behave rationally
- All messages included in the proposed protocols include *Nonce* and a transaction ID to prevent replay attacks and also to ensure the refreshment of all messages. But the *Nonce* and the transaction ID are omitted for simplicity
- All messages include a time of a transaction to be used for disputes resolution where applicable. But the time is omitted for simplicity

4.4. Summary

This chapter has covered the cryptographic concepts that will be used in this thesis. Additionally, the assumptions have been presented.

Chapter Five

5. Enforcing Customer Honesty Protocol

5.1 Introduction

This chapter starts by presenting the technique that is originated in this thesis. Then, it presents the Enforcing Customer Honesty (ECH) protocol that is based on the technique proposed in this thesis. In this chapter the ECH protocol will be discussed and analysed.

The approach proposed in this thesis can be described as follows. There is a trustworthy party (called the first party) who is encouraged to be honest and the other party (called the second party) is enforced to be honest. Figure 7 shows a general diagram for the proposed approach. The approach consists of only three messages to be exchanged between the two parties. The trustworthy party will start the exchange by sending the first message (E-M1). When the second party (the one who is enforced to be honest) receives the first message, they test the trustworthiness of the first party by checking all the items included in the first message (i.e. E-M1). The items in E-M1 are encrypted but the second party will be able to check whether they are correct or not using certificates. If the second party finds that the first party is indeed trustworthy (i.e. all items in E-M1 are correct) then they send the second message (E-M2) to the first party. The items included in E-M2 are encrypted using encryption keys that are known to the first party. Therefore, on receiving E-M2 the first party will be able to decrypt E-M2 (hence the second party is enforced to be honest by sending correct E-M2). After correctly decrypting the second party's items, the first party will send the third message (E-M3) to the second party. The third message includes the decryption

key to be used to decrypt the encrypted items received in the first message (E-M1). If anything went wrong, the TTP will be involved to resolve disputes.

Using this approach the number of messages needed to exchange the items of the two parties are only three messages.

This approach is applied to have a trustworthy merchant and then enforce the customer to be honest which result in the ECH protocol. Then, the approach is applied to have a trustworthy customer and then enforce the merchant to be honest which result in the EMH protocol.

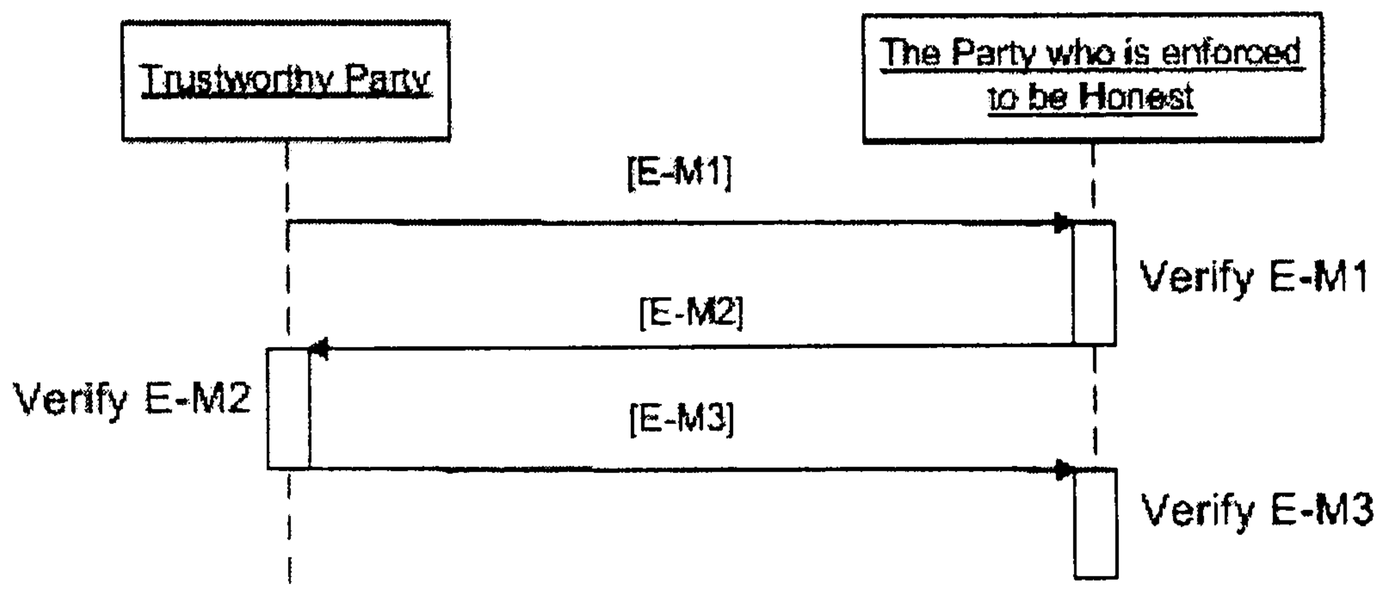

Figure 7: The proposed approach

5.2 ECH Protocol Description

5.2.1 Description

The Enforcing Customer Honesty (ECH) protocol is an optimistic fair exchange protocol for exchanging a digital product (D) for a payment. The basic idea of the ECH protocol is to have one trustworthy party (Merchant -M) and enforce the other party (Customer -C) to be honest.

The trustworthiness of M is governed by two factors, the digital product certificate (*D-Cert*) issued by the Certificate Authority (CA) and the shared public key

certificate (*C.mt*) issued by the TTP. Therefore, the digital product that will be sent by M is certified by the CA; and the public key to be used by M to encrypt the key that encrypts the digital product is certified by the TTP.

The enforcement of C to be honest is governed by letting C encrypt the payment using a key that is sent to C by M. This is to let M be able to decrypt the payment without the help of C. Hence, C will send a correct encrypted payment because M will be able to decrypt it as soon as they receive it.

The ECH protocol is like M saying to C, this is the encrypted digital product; and I am trustworthy as these certificates (*D-Cert* and *C.mt*) show BUT be honest with me by sending the correct payment to be able to receive the decryption key for the digital product. By this means M is honest by having the certificates from the CA and the TTP and also C is enforced to be honest.

The ECH protocol consists of two phases. The pre-exchange phase and the exchange phase. In the pre-exchange phase, the merchant gets the digital product and the certificates (from trusted authorities) to be used in the exchange phase. In the exchange phase, the customer and the merchant exchange their items. The items of parties are sent using messages. It is not necessarily that the exchange phase comes immediately after the pre-exchange phase.

5.2.2 Pre-exchange Phase

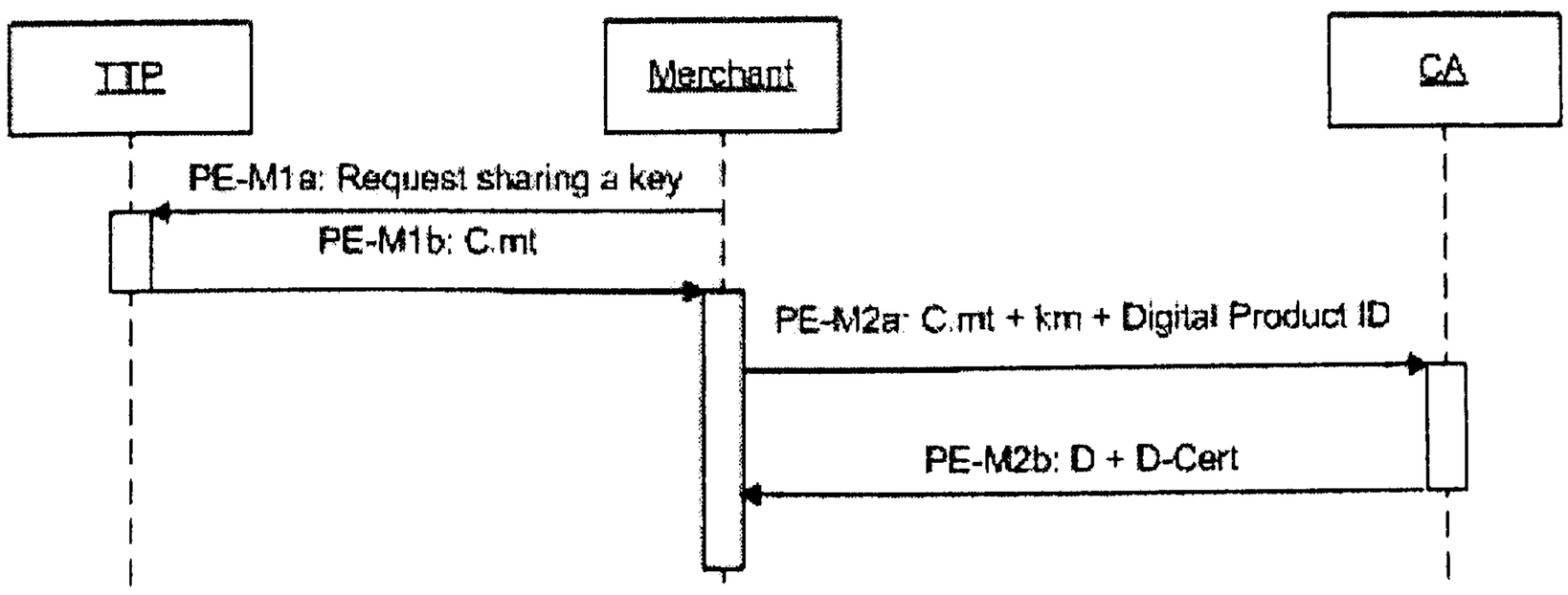

Figure 8: ECH Pre-exchange Phase

In the pre-exchange phase (Figure 8), M needs to get the certificate *C.mt* of the shared public key (*pkmt*) from the TTP to be used to encrypt the key that is used to encrypt D. To get the shared public key certificate, M needs to request it from the TTP then the TTP will issue it (that is in messages PE-M1a and PE-M1b of Figure 8). M also needs to get the digital product (D) and its certificate *D-Cert* from the CA (that is in messages PE-M2a and PE-M2b of Figure 8), (the CA can be thought of as the producer of the digital product).

In this protocol, there is a shared public key between M and the TTP. The keys of the parties are as follows:

- Each party *x* (*x* ∈ M, C, TTP and CA) has its own RSA public (*pkx*) and private (*skx*) keys.
    - The CA's public key is denoted as *pkca = (eca, nca)* and its corresponding private key is denoted as *skca = (dca, nca)*
    - The TTP's public key is denoted as *pkt = (et, nt)* and its corresponding private key is denoted as *skt = (dt, nt)*
    - M's public key is denoted as *pkm = (em, nm)* and its corresponding private key is denoted as *skm = (dm, nm)*
    - C's public key is denoted as *pkc = (ec, nc)* and its corresponding private key is denoted as *skc = (dc, nc)*
- The shared RSA public key between M and TTP is denoted as *pkmt = (emt, nmt)* and its corresponding private key is denoted as *skmt = (dmt, nmt)*

5.2.3 The Exchange Phase

It is assumed that the exchange phase will take place after C finds the required digital product (D) with M. It is also assumed that this phase will take place after C and M agree on the digital product and negotiated the price. Hence this phase is about the actual exchange of payment and digital product D.

There are only three messages to be exchanged between M and C in the exchange phase (Figure 9). These three messages are as follows:

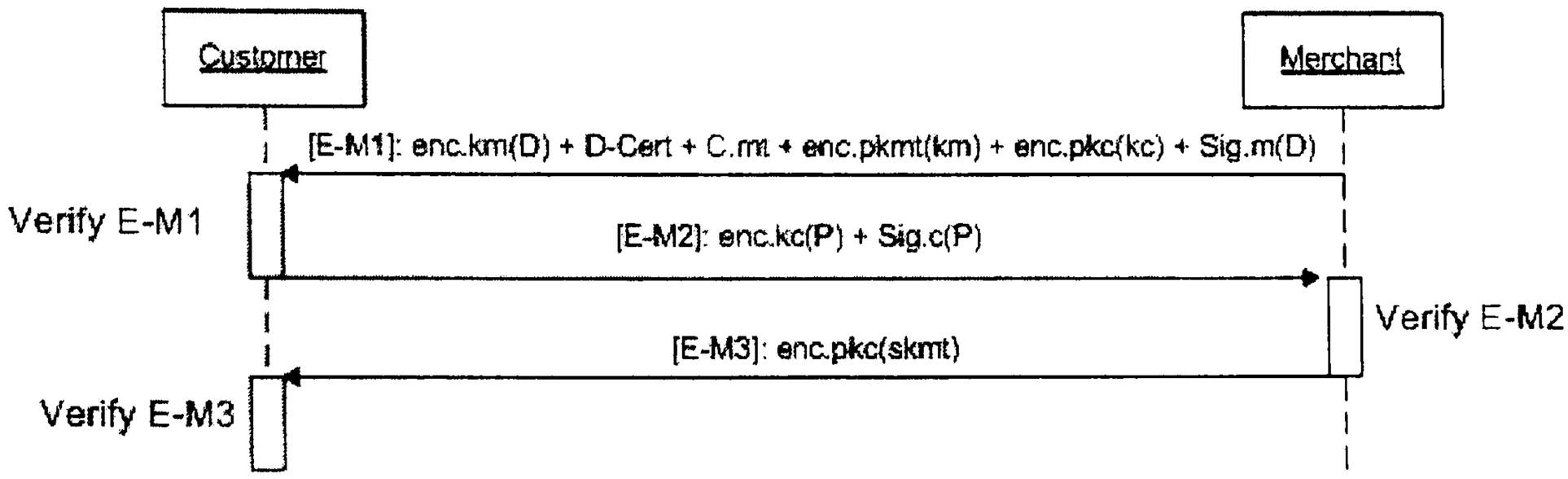

Figure 9: ECH Exchange Phase

[E-M1] **M → C:** ***enc.km(D) + D-Cert + C.mt + enc.pkmt(km) + enc.pkc(kc) + Sig.m(D)***

The first message is sent to C by M and contains the following:

- ***enc.km(D)***: the digital product D that is encrypted with the key *km* (note that *km* is generated by M)
- ***D-Cert***: the digital product certificate that is issued by the CA
- ***C.mt***: the shared public key certificate that is issued by the TTP
- ***enc.pkmt(km)***: the key *km* (that is used to encrypt D) is encrypted using the shared public key *pkmt* (*pkmt* is certified in *C.mt*)
- ***enc.pkc(kc)***: a key *kc* that will be used by C to encrypt the payment. *kc* is encrypted using the public key of C (*pkc*). The reason for encrypting *kc* using *pkc* is to prevent any other party from gaining *kc* as it will be used to encrypt the payment by C (note that *kc* is generated by M)
- ***Sig.m(D)***: M's signature on D. This signature can serve as non-repudiation of origin which allows C to be sure that D is sent by M. M's signature on D is the encryption of the hash value of D using M's private key *skm*

[E-M2] **C → M:** ***enc.kc(P) + Sig.c(P)***

On receiving message *E-M1* from M, C checks the correctness of *enc.km(D)*, *enc.pkmt(km),* Sig.m(D), *D-Cert* and *C.mt*. The correctness of *D-Cert* can be checked by verifying CA's signature on *D-Cert* and the correctness of *C.mt* can be checked by verifying TTP's signature on *C.mt*.

To check the correctness of D, C needs to check two things; the digital product D itself and the encrypted D with *km* i.e. *enc.km(D)*. Firstly, to check the correctness of D, C needs to get the hash value of D (calculated as *HD*) by decrypting *Sig.m(D)* contained in message *E-M1* using M's public key *pkm* (the public keys of all parties are publicly available) and then compare it with hash value of D (*hD*) contained in *D-Cert* i.e. to check the following:

$$HD \stackrel{?}{=} hD$$

If they are the same then C can be sure that M signed the correct D.

Secondly, to check the correctness of the encrypted D i.e. *enc.km(D)*, C computes the hash value of *enc.km(D)* (calculated as *HeD*) and then compare it with the hash value of encrypted D with *km* i.e. *heD* which is contained in *D-Cert* (note that it is assumed that C will use the same hash algorithm used by the CA to compute the hash value) i.e. to check the following:

$$HeD \stackrel{?}{=} heD$$

If they are the same then C can be sure that M encrypted D using *km* and not another key.

C also needs to check the correctness of *km* which is used to encrypt D. To do so, C computes the hash value of *enc.pkmt(km)* (calculated as *HeKm*) and then compares it with *heKm* that is included in *D-Cert*, so C will check the flowing:

$$HeKm \stackrel{?}{=} heKm$$

If they compare then C can be sure that the encrypted key is *km* and not another key. The point here is to make sure that M is honest by sending the key used to encrypt D.

If all comparisons are correct then, at this point, C will have the following fact. The encrypted D is correct (i.e. it is the one certified in *D-Cert*) and it is indeed encrypted with *km*. In addition, the key encrypted using the shared public key *pkmt* is indeed *km* and not another key. The shared public key *pkmt* used to encrypt *km* is certified by the TTP. Therefore, once C has got the private key of the shared public key then C will be able to get D (by first decrypting *enc.pkmt(km)* to get *km* and then decrypting *enc.km(D)* using *km*).

At this point C must be sure that the encrypted D matches their requirements, otherwise they will be at risk if they send message *E-M2* to M as M already has the decryption key for the payment.

Now, it is C's choice to complete the exchange or abort the protocol. If C wants to exchange the payment for D then they send the following:

- ***enc.kc(P)*:** C first needs to get *kc* by decrypting *enc.pkc(kc)* included in *E-M1* using their private key *skc* as *kc* encrypted using C's public key. Once C got *kc* then they encrypt the payment (P) using the key kc
- ***Sig.c(P)***: C's signature on P. This signature can serve as non-repudiation of receipt which allows M to be sure that C has received the encrypted D (it also can serve as non-repudiation of origin of the P)

Note that if C decides to abort the transaction after receiving message *E-M1* and before sending message *E-M2* to M then neither C nor M lose anything. But once C sends a correct message in *E-M2* to M then the transaction must be completed and the protocol will guarantee that the exchange of P and D will be fair.

[E-M3]* M → C: *enc.pkc(skmt)

On receiving message *E-M2*, M decrypts the payment using the key *kc* (*kc* is originated by M and sent to C in *E-M1* to allow C to encrypt the payment) and then checks whether it is correct. If C encrypted the payment using a different key then M ignores the transaction and aborts the protocol. If however the payment is correct then M sends the decryption key *skmt* (it is encrypted using C's public key to prevent any other party from gaining *skmt*) to C to be able to decrypt *km* and then decrypt D.

It is clear that if the payment is incorrect then M will not send *skmt* to C because it is C's responsibility to send the correct payment to be able to receive the decryption key in message *E-M3*. Using this method this protocol enforces C to be honest and hence to send the correct payment to M. If C sends incorrect payment, then M can ask C to re-send the correct payment to be able to receive the decryption key, but it depends on M if they want to do so.

5.2.4 Dispute Resolution

All disputes, if any, will come from C since M will not need to raise disputes because they receive the encrypted payment (and decrypt it using the key *kc*) before they send the decryption key to C. Therefore, the weakest link in this exchange is C as they have to send the correct payment in order to receive the decryption key for the encrypted D that they received in message *E-M1*.

Thus, if C has a dispute, the following messages are sent (Figure 10):

[DR-M1]* C → TTP: *D-Cert + C.mt + Sig.m(D) + enc.pkt(kc) + enc.kc(P)

In the case where C has a dispute, they need to send to the TTP the following: *(D-Cert + C.mt + Sig.m(D) + enc.pkt(kc) + enc.kc(P))*

[DR-M2]* TTP → M: *enc.kc(P) + enc.pkm(kc)

On receiving message *DR-M1*, the TTP checks the correctness of *D-Cert* and *C.mt* by checking their signatures. It decrypts *enc.pkt(kc)* using its private key *skt* to get *kc* then uses *kc* to decrypt the payment. The hash included in *Sig.m(D)* is compared with the one in *D-Cert* (i.e. hD). If all of these are correct then the TTP checks the amount of the payment against the price field that is in *D-Cert*. Also, the TTP checks if the payee in the payment matches the merchant's name (note, the TTP may need to contact the customer's bank for verifying the payment if necessary). If the TTP finds the payment is correct then it forwards to M the payment (encrypted using *kc*) and *kc* that decrypts the payment (*kc* is sent encrypted using M's public key *pkm* to prevent any other party from gaining *kc*).

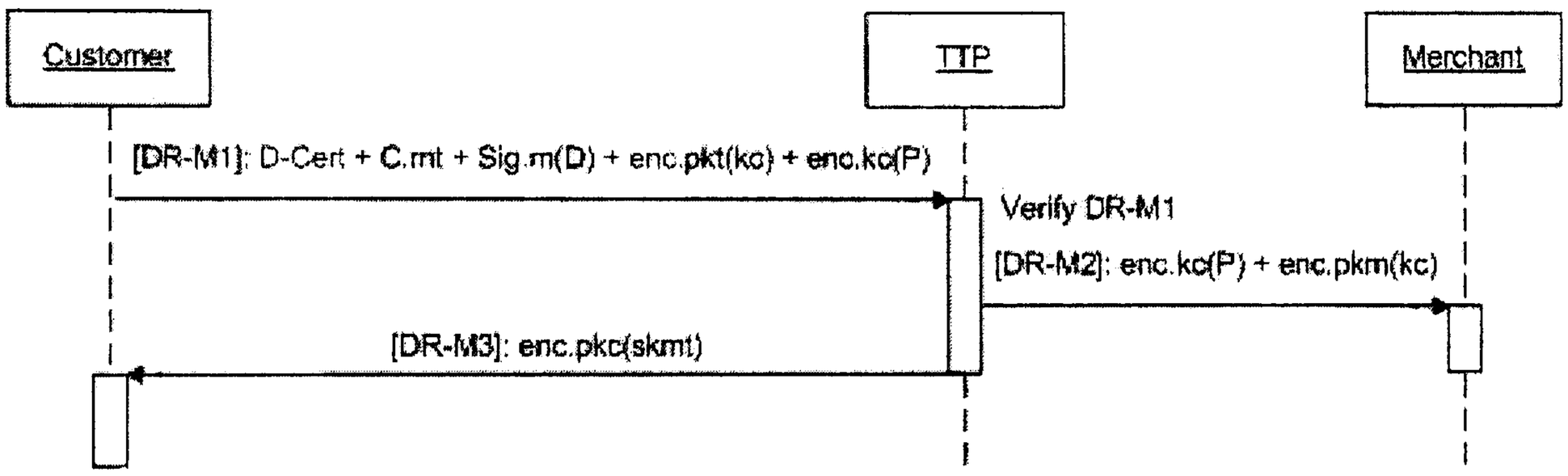

Figure 10: ECH Dispute Resolution

The reasons for forwarding the encrypted payment and *kc* to M (note that M is not the one who contacted the TTP asking for resolution) are as follows:

- C may have sent to M incorrect payment in message *E-M2* (in the exchange phase)
- C may have not sent the payment at all; i.e. C contacted the TTP before sending *E-M2* to M
- C may have encrypted the payment in *E-M2* using different *kc*; therefore, the TTP forwards the key *kc* that decrypts the payment

Otherwise, if the TTP found that the amount of the payment does not match the price in *D-Cert* then it sends an abort message to C. If the *DR-M1* is incorrect then the TTP will not contact M at all (i.e. the TTP will not send DR-M2 to M).

[DR-M3] **TTP → C:** ***enc.pkc(skmt)***

OR

TTP → C: aborts;

This is the same process for message *DR-M2* above. If the TTP found that *DR-M1* is correct then it sends to C the decryption key *skmt* that is encrypted using C's public key *pkc*. Otherwise, if *DR-M1* is incorrect then the TTP sends an abort message to C.

It is clear that if either C has not sent the correct payment to M in *E-M2*, C has not sent the payment at all or C encrypted the payment in *E-M2* using different *kc* then C will not get an advantage over M because the TTP will check the payment and see whether it meets the price or not. If the amount of the payment meets the price then the TTP will send the decryption key *skmt* to C and will also forward the encrypted payment and its decryption key *kc* to M to ensure fairness in all cases. Therefore, the fairness is ensured for both C and M. However, if the payment is incorrect then the TTP will reject C's request for a dispute.

As can be seen in the dispute resolution phase, the TTP does not need to have both C and M involved in order for the dispute to be resolved; rather only the disputant (C in this protocol) and the TTP will be involved. That is, the TTP does not need to contact M to verify whether or not they have received the correct payment; rather the TTP asks C to provide all needed information (in *DR-M1*) and will be able to make the resolution automatically. M will only be contacted by the TTP if the dispute has a resolution. Therefore, this will reduce the number of messages needed to resolve disputes.

5.3 ECH Protocol Analysis

In this section, the ECH protocol will be analysed. The analysis includes the ability of parties to detect the dishonesty of one another, disputes analysis, and presenting and discussing all possible scenarios.

5.3.1 Detection of Dishonesty

It is crucial to test whether or not the customer and the merchant in the ECH protocol will be able to detect if the other party is dishonest. The customer may act dishonestly by sending an incorrect E-M2 to the merchant. An incorrect E-M2 means either

(a) The payment is incorrect,

(b) The payment is encrypted with a key that M does not know (i.e. the payment is not encrypted with *kc*),

(c) The signature of C on the payment (i.e. *Sig.c(P)*) is incorrect.

So, on receiving E-M2 M will check if the payment is encrypted with *kc*, if the payment is correct, and if the signature is correct. Hence, detecting a dishonest Customer is simple.

On the other hand, the cases in which the Merchant may act dishonestly are by either:

1. sending an incorrect E-M1;
2. sending an incorrect E-M3;
3. not sending E-M3

In case 2 (i.e. sending an incorrect E-M3) there is only one possibility (by which M may act dishonestly) and this is where the decryption key sent by M to C in E-M3 is incorrect. In case 3 (i.e. not sending E-M3) there is also one possibility which

is M not sending E-M3 to C. Case 1 (i.e. sending an incorrect E-M1) has a number of possibilities. These possibilities will be studied as follows.

Message E-M1 includes the following: *enc.km(D), D-Cert, C.mt, enc.pkmt(km), enc.pkc(kc),* and *Sig.m(D)*. C does not need to check the correctness of *enc.pkc(kc)* because it includes the key *kc* that C will use to encrypt the payment if they are interested in the exchange. So, C will check the correctness of the other five items (i.e. *enc.km(D), D-Cert, C.mt, enc.pkmt(km),* and *Sig.m(D)*). M may act dishonestly by including incorrect items in E-M1. It is important to study the all possibilities in which M may act dishonestly in E-M1.

Note that *D-Cert* and *C.mt* are not included in this discussion because they will be checked first by C and if they are incorrect then C will not check the correctness of *enc.km(D), enc.pkmt(km)* and *Sig.m(D)*. However, if both *D-Cert* and *C.mt* are correct then C will check the correctness of *enc.km(D), enc.pkmt(km)* and *Sig.m(D*).

The possibilities for the items *enc.km(D), enc.pkmt(km)* and *Sig.m(D*) in E-M1 are:

A. *enc.km(D)*: there are two things involved in forming *enc.km(D)*, namely *D* and *km*. So, there are four possibilities for forming *enc.km(D)*:
   1) *D* is correct and *km* is correct. So, M is honest
   2) *D* is incorrect and *km* is incorrect. So, M is dishonest
   3) *D* is correct and *km* is incorrect. So, M is dishonest
   4) *D* is incorrect and *km* is correct. So, M is dishonest

B. *enc.pkmt(km)*: there are two things involved in forming *enc.pkmt(km)*, namely *km* and *pkmt*. So, there are four possibilities for forming *enc.pkmt(km)*:
   1) *km* is correct and *pkmt* is correct. So, M is honest
   2) *km* is incorrect and *pkmt* is incorrect. So, M is dishonest
   3) *km* is correct and *pkmt* is incorrect. So, M is dishonest
   4) *km* is incorrect and *pkmt* is correct. So, M is dishonest

C. *Sig.m(D)*: the signature of M on D means the hash value of D encrypted with M's private key. Therefore, there are two things involved in forming *Sig.m(D)*. They are *hash value of D* and *M's private key*. Therefore, there are four possibilities for forming *Sig.m(D)*:

1) *hash value of D* is correct and *M's private key* is correct. So, M is honest
2) *hash value of D* is incorrect and *M's private key* is incorrect. So, M is dishonest
3) *hash value of D* is correct and *M's private key* is incorrect. So, M is dishonest
4) *hash value of D* is incorrect and *M's private key* is correct. So, M is dishonest

Table 2 shows the probabilities of combining *enc.km(D), enc.pkmt(km),* and *Sig.m(D)* in E-M1. Symbol (√) means correct and (¬) means incorrect. The cases in Table 2 are analysed as follows (the numbers below refers to the numbers in Table 2), (Note that having one incorrect in E-M1 means that M is dishonest and C must not send the payment to M. However, all cases will be studied).

	enc.km(D)	**enc.pkmt(km)**	**Sig.m(D)**
1	√	√	√
2	¬	¬	¬
3	¬	√	√
4	√	¬	√
5	√	√	¬
6	√	¬	¬
7	¬	√	¬
8	¬	¬	√

Table 2: Possibilities of items in E-M1 of the ECH protocol

1. **enc.km(D), enc.pkmt(km), Sig.m(D):**

    - The encrypted D is the one certified in D-Cert and is encrypted with km and not another key
    - The encrypted key is km and is encrypted with the key that is certified in C.mt
    - The signed D is the same as D that is certified in D-Cert

Result: the hash values will be the same (i.e. all verifications are correct) and hence M is honest.

2. **¬ enc.km(D), ¬enc.pkmt(km), ¬Sig.m(D):**
    - The encrypted D is different from the one certified in D-Cert. This has the following possibilities:
        - D is different from the one certified in D-Cert and km is also different from the one certified in D-Cert
        - D is the same as certified in D-Cert but km is different from the one certified in D-Cert
        - D is different from the one certified in D-Cert and km is the one certified in D-Cert
    - The encrypted key (*km*) is not the one certified in D-Cert. This has the following possibilities:
        - km is different from the one certified in D-Cert and pkmt is different from the one certified in C.mt
        - km is the same as the one certified in D-Cert but pkmt is different from the one certified in C.mt
        - km is different from the one certified in D-Cert and pkmt is the one certified in C.mt
    - The signed D is not the same as D that is certified in D-Cert. This has the following possibilities:
        - The hash of D is incorrect and M's private key is incorrect
        - The hash of D is correct but M's private key is incorrect
        - The hash of D is incorrect and M's private key is correct

Result: it is clear that C will easily detect the problems in this E-M1 by different ways. First, C can detect that M is dishonest when C computes the hash value for enc.km(D) and compares it with the hash value of encrypted D (heD) in D-Cert. Second, C can detect that M is dishonest when C computes the hash value for enc.pkmt(km) and compares it with the hash value of encrypted km (heKm) in D-Cert. Third, when C compares the hash value of the D, that is in Sig.m(D), with the hash value of D (hD) in D-Cert. Therefore, C can detect that M is dishonest.

3. **¬ enc.km(D), enc.pkmt(km), Sig.m(D):**
    - The encrypted D is different from the one certified in D-Cert. This has three possibilities (these possibilities are the same ones mentioned in case number 2 above)
    - The encrypted key is km and is encrypted with the key that is certified in C.mt
    - The signed D is the same as D that is certified in D-Cert

Result: C can detect that M is dishonest when C computes the hash value for enc.km(D) and compares it with the hash value of encrypted D (heD) in D-Cert.

4. **enc.km(D), ¬enc.pkmt(km), Sig.m(D):**
    - The encrypted D is the one certified in D-Cert and is encrypted with km and not another key
    - The encrypted key is not the one certified in D-Cert. This has three possibilities(these possibilities are the same ones mentioned in case number 2 above)
    - The signed D is the same as D that is certified in D-Cert

Result: C can detect that M is dishonest when C computes the hash value for enc.pkmt(km) and compares it with the hash value of encrypted km (heKm) in D-Cert.

5. **enc.km(D), enc.pkmt(km), ¬Sig.m(D):**
    - The encrypted D is the one certified in D-Cert and is encrypted with km and not another key

- The encrypted key is km and is encrypted with the key that is certified in C.mt
- The signed D is not the same as D that is certified in D-Cert. This has three possibilities (these possibilities are the same ones in case number 2 above)

Result: C can detect that the signed D is different from the one certified in D-Cert when C gets the hash value of D using M's public key and then comparing this hash value with the hash value of D (hD) in D-Cert. Note, having the signed D different from the one certified in D-Cert will not compromise fairness because the most important parts that C must be sure that they match D-Cert are *enc.km(D)* and *enc.pkmt(km)* because if they are compared to the ones in D-Cert then C is sure that they can get the decryption key from TTP in case M becomes dishonest. However, it is up to C if they want to proceed or not but having all verifications are correct in E-M1 but the signed D is not the same as D that is certified in D-Cert gives C a sign that M may be dishonest. This will not affect the fairness. However, C is recommended not to send E-M2 because if M did not send E-M3 then C will need to contact the TTP (i.e. for resolution) who will find *Sig.m(D)* not correct and hence will not resolve the dispute.

6. **enc.km(D), ¬enc.pkmt(km), ¬Sig.m(D):**
   - The encrypted D is the one certified in D-Cert and is encrypted with km and not another key
   - The encrypted key is not the one certified in D-Cert. This has three possibilities (these possibilities are the same ones in case number 2 above)
   - The signed D is not the same as D that is certified in D-Cert. This has three possibilities (these possibilities are the same ones in case number 2 above)

Result: C can detect that M is dishonest when (a) C computes the hash value for enc.pkmt(km) and compares it with the hash value of encrypted km (heKm) in D-Cert; and (b) C gets the hash value of D (i.e. from Sig.m(D)) using M's public key and then comparing this hash value with the hash value of D (hD) in D-Cert

7. **¬enc.km(D), enc.pkmt(km), ¬Sig.m(D):**
    - The encrypted D is different from the one certified in D-Cert. This has three possibilities (these possibilities are the same ones in case number 2 above)
    - The encrypted key is km and is encrypted with the key that is certified in C.mt
    - The signed D is not the same as D that is certified in D-Cert. This has three possibilities (these possibilities are the same ones in case number 2 above)

Result: C can detect that M is dishonest when (a) C computes the hash value for enc.km(D) and compares it with the hash value of encrypted D (heD) in D-Cert; and (b) C gets the hash value of D (i.e. from Sig.m(D)) using M's public key and then comparing this hash with the hash value of D (hD) in D-Cert

8. **¬enc.km(D), ¬enc.pkmt(km), Sig.m(D):**
    - The encrypted D is different from the one certified in D-Cert. This has three possibilities (these possibilities are the same ones in case number 2 above)
    - The encrypted key is not the one certified in D-Cert. This has three possibilities (these possibilities are the same ones in case number 2 above)
    - The signed D is the same as D that is certified in D-Cert

Result: C can detect that M is dishonest when (a) C computes the hash value for enc.km(D) and compares it with the hash value of encrypted D (heD) in D-Cert; and (b) C computes the hash for enc.pkmt(km) and compares it with the hash value of encrypted km (heKm) in D-Cert.

To sum up the cases, it is clear from the possibilities discussed in Table 2 that if M tried to cheat on C by including incorrect items in E-M1 then C will be able to detect it and refuse to exchange the payment with M. Therefore, the fairness in ensured for both C and M. However, if M behaved honestly in message E-M1 but sent incorrect E-M3 or disappeared before sending the decryption key in message E-M3

then TTP can guarantee the fairness and will send the decryption key to C if they find that C is honest in DR-M1 i.e. DR-M1 is correct.

5.3.2 Dispute Analysis

By the end of exchanging a digital product and a payment (regardless who starts sending their item) between M and C, there are three possibilities for C (in here we are talking about the normal exchange where no protocols are used):

1) C received the correct digital product
2) C received incorrect digital product
3) C did not receive the digital product at all

There are also three possibilities for M:

1) M received the correct payment
2) M received incorrect payment
3) M did not receive the payment at all

The incorrect digital product means that the received digital product is not the one that C wanted; whereas incorrect payment means that the received payment is not the same as the requested price by M.

Table 3 studies the combination of these possibilities for C and M. In Table 3, (X) means either the party (C or M) has not received the item (payment or digital product) at all or they received incorrect item; whereas (√) means the correct item is received. Note that the resolution for dispute is not specified as this discussion is concerned with exchange and resolution in general.

	C	M	Result
	Receive digital product	**Receive payment**	
1	√	√	No dispute
2	√	X	M disputes
3	X	√	C disputes
4	X	X	There are possibilities for disputes by both C and M

Table 3: Disputes Possibilities

	C	M	Result
	Receive decryption key	**Receive payment**	
1	√	√	No dispute
2	√	X	Not applicable
3	X	√	C disputes
4	X	X	No dispute / Not applicable / C's fault

Table 4: Disputes Possibilities for ECH

As can be seen in the Table 3, if M and C receive each other's items then there is no need for a dispute (case 1 in Table 3). If C received the correct digital product and M either received incorrect payment or has not received the payment at all then M will dispute (case 2 in Table 3). If M received the correct payment and C either received incorrect digital product or has not received the digital product at all then C will dispute (case 3 in Table 3). Finally, the case number 4 in Table 3 has four possibilities which are as follows:

a) if both C and M have not received anything from each other then no dispute will be made as both of them have not revealed their items

b) if both C and M have received incorrect items from each other (i.e. C received incorrect digital product and M received incorrect payment) then both C and M will dispute
c) if C received incorrect digital product and M has not received the payment at all then both C and M will dispute
d) if C has not received the digital product at all and M received incorrect payment then both C and M will dispute

For the ECH protocol (Table 4), the actual exchange between C and M is for the payment (from C) and the decryption key for the encrypted digital product (from M). This is because M sends the encrypted digital product to C and C will verify it and if satisfied then the exchange of the payment and the decryption key will take place. The order of the exchange of the payment and the decryption key in the ECH protocol is as follows. M will receive a correct payment before C receives the decryption key.

The following studies the cases presented in Table 4 which shows all possible cases for disputes for the ECH protocol. The meaning of (X) and (√) in Table 4 is the same as the ones in Table 3.

1) In case 1, both C and M receive the correct item (i.e. C receives the correct decryption key and M receives the correct payment) from each other. Hence, there is no dispute.
2) In case 2, C received a correct decryption key, and M either received incorrect payment or has not received the payment at all. This case is not applicable in the ECH protocol because C has to send a correct payment to be able to receive the correct decryption key.
3) In case 3, C has either received incorrect decryption key or not received the decryption key at all, and M received the correct payment. In this case C will dispute to TTP.
4) In case 4, there are four possibilities which are:
    a) Both C and M have not received anything from each other. So, no dispute will be made as both of them have not revealed their items. This represents the case where C received E-M1 and did not send E-M2 to M.

b) Both C and M have received incorrect items from each other. That is, C received incorrect decryption key and M received incorrect payment. This case is not applicable in the ECH protocol because C has to send a correct payment to be able to receive the correct decryption key. So, if M found that the payment is incorrect then M will not send the decryption key at all i.e. neither correct decryption key nor incorrect decryption key.

c) C received an incorrect decryption key and M has not received the payment at all. This case is not applicable in the ECH protocol because C has to send a correct payment to be able to receive the correct decryption key. So, if M has not received the payment then M will not send the decryption key at all.

d) C has not received the decryption key at all and M received incorrect payment. This case is normal because if the payment is incorrect then M will not send the decryption key to C. That is, C has to send a correct payment to be able to receive the correct decryption key from M. Therefore, if this case occurs then for C to raise a dispute to the TTP, C needs to send to the TTP the correct payment. If C sends the correct payment to the TTP then the TTP will make a resolution to both C and M. That is, M may receive two payments (the incorrect payment from C, and the correct payment from TTP) and this is the penalty that C pays for being dishonest. Of course M may choose to disregard the incorrect payment. However, if the TTP found that the payment is incorrect then C's dispute will be rejected.

The design of the ECH protocol reduces the possibilities for having disputes. Additionally, only C will raise disputes as M will not send their item unless the items from C are correct. Hence, the ECH protocol enforces C to be honest. As a result, the possibilities for disputes are reduced by preventing them occurring.

In addition to the previous cases, the following cases are studied:

- C disputes that after decrypting the digital product they have found the digital product is incorrect. This is not possible because *D-Cert* guarantees that the digital product D is correct; and if C found that D is incorrect then they should

not have sent the payment to M. So, it is C's fault for sending the payment to M if they had any doubt about D. Once C sends the payment to M then this means that they are satisfied with the digital product D. Therefore, this dispute will not happen because C knows the rules of the protocol which allow C to check the digital product before they send the payment to M; and as a result C will not put themselves at risk

- It is clear that M will not raise a dispute because M will receive the payment before they send the decryption key to C. However, the following cases are studied:
    - M claims that they have received incorrect payment from C. This will not happen because C knows that if they send incorrect payment then they will not receive the decryption key. However, if this case does happen then it is C's fault for sending the incorrect payment. Again, the idea of this protocol is to have one trustworthy party and enforce the other party to be honest if they are willing to exchange their item. Therefore, if M received incorrect payment then they will not send the decryption key i.e. M's item is not revealed and hence M does not need to dispute.
    - M claims that they have not received the decryption key to decrypt the payment. This is not applicable in this protocol because the payment is encrypted with *kc* that is sent by M; and hence M is able to decrypt the payment as soon as they get it. However, if M found that the payment is not encrypted with *kc* then M will not send the decryption key to C in E-M3.

5.3.3 Scenarios Analysis

There are different scenarios for executing the ECH protocol by C and M. These scenarios include:

a) having both C and M are behaving honestly
b) C is behaving dishonestly and M is behaving honestly
c) M is behaving dishonestly and C is behaving honestly
d) having both C and M are behaving dishonestly

The possible scenarios for executing the ECH protocol are as follows:

1. C and M are honest which result in normal execution. That is, M sends a correct E-M1, C sends a correct E-M2, and M sends a correct E-M3 (Figure 11).

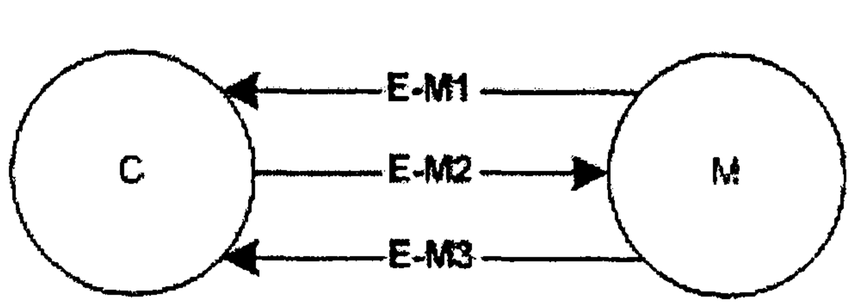

Figure 11: ECH Protocol, Scenario 1

2. After receiving E-M1, C quits the protocol because either E-M1 is incorrect or C is no longer interested in the exchange (Figure 12).

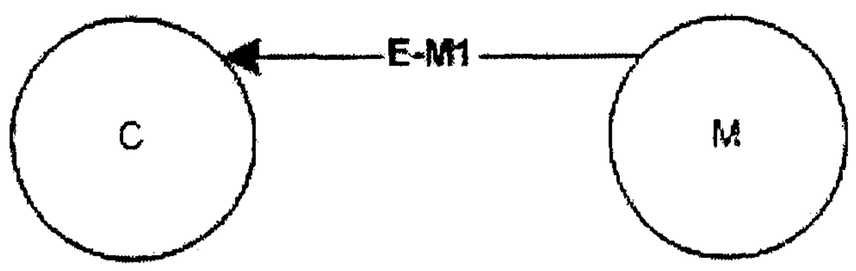

Figure 12: ECH Protocol, Scenario 2

3. After receiving E-M1, C contacts the TTP before sending E-M2 to M. However, the TTP found that DR-M1 is incorrect. Therefore, the TTP sends abort message to C. In this scenario C tries to cheat but they gained nothing (Figure 13).

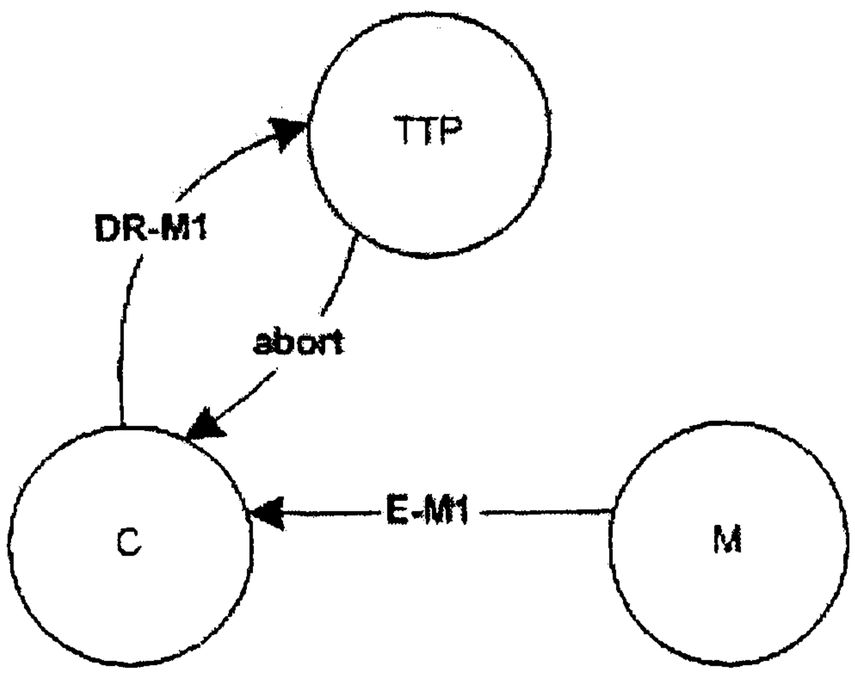

Figure 13: ECH Protocol, Scenario 3

4. The same as scenario number 3 but in here, the TTP found that DR-M1 is correct. Therefore, the TTP will make it fair for both C and M by sending the resolution to M in DR-M2 and also by sending the resolution to C in DR-M3. In this scenario C tries to cheat but the TTP makes it fair for both C and M (Figure 14).

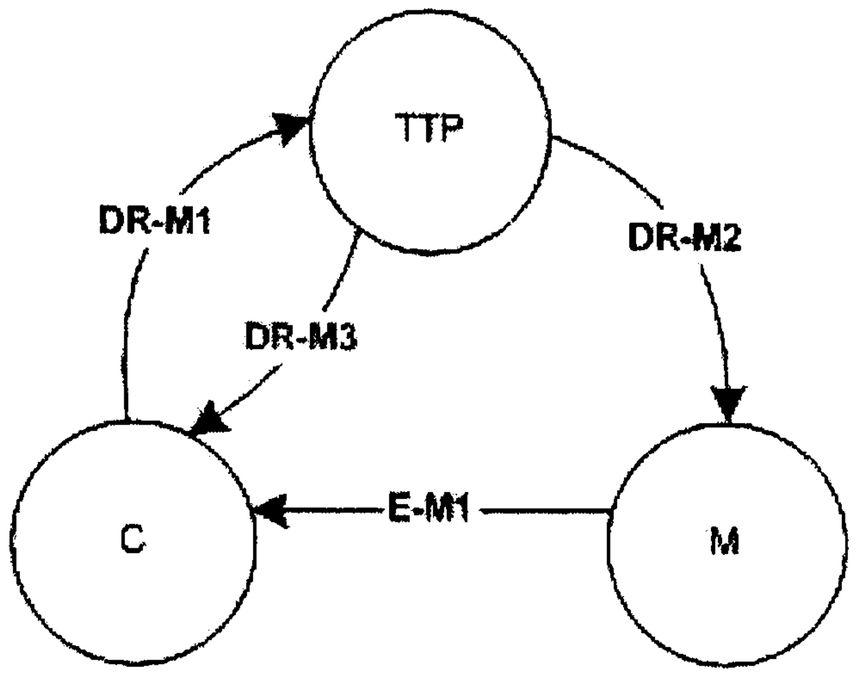

Figure 14: ECH Protocol, Scenario 4

5. After C received E-M1 from M, C found that E-M1 is correct and hence C sends E-M2 to M. C waited to receive E-M3 from M but nothing is received. Therefore, C contacted the TTP for resolution. However, the TTP found that DR-M1 is incorrect and hence the TTP sends an abort message to C. Note, there are two possibilities why M did not send E-M3 to C. These possibilities are either because M found that E-M2 is incorrect or because M is dishonest. If it is the former (where E-M2 is incorrect) then it is C's fault for sending an incorrect E-M2 to M. While, if it is the later (where E-M2 is correct but M is dishonest) then C needs to send correct DR-M1 to the TTP to be able to receive a resolution (Figure 15).

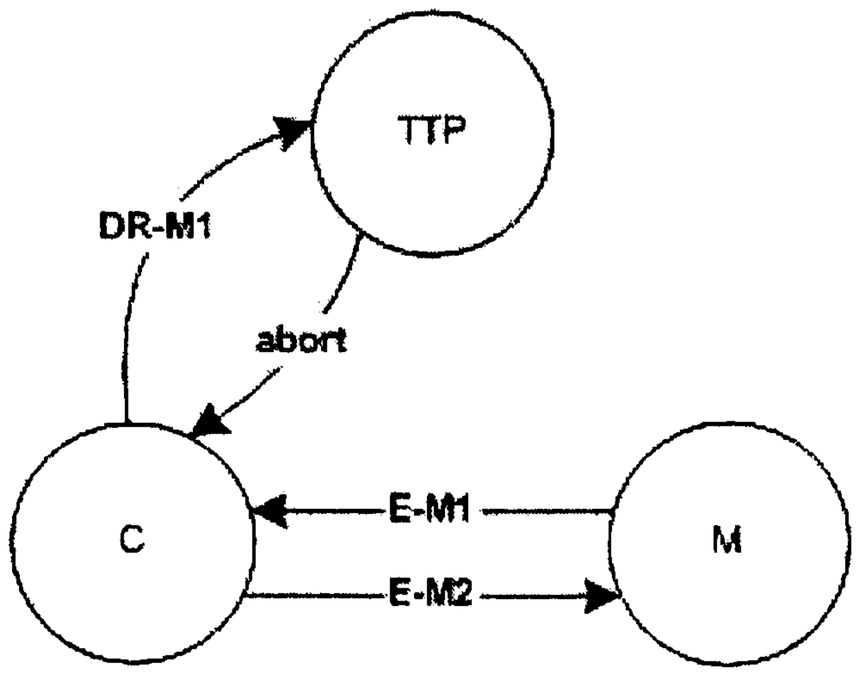

Figure 15: ECH Protocol, Scenario 5

6. The same as scenario number 5 but here, the DR-M1 that C sent to the TTP is correct. Therefore, the TTP resolves it by sending DR-M2 to M and DR-M3 to C. Again, the reason for M not sending E-M3 to C is either because E-M2 is incorrect or because M is dishonest. If it is because E-M2 is incorrect, then for C to receive the DR-M3 (which is the decryption key for the encrypted digital product) then C has to send correct DR-M1 to the TTP. When the TTP received the correct DR-M1 then the TTP will make it fair for both C and M. While if M did not send E-M3 to C because M is dishonest, then C has to send the correct DR-M1 to the TTP to resolve C's dispute. In this case (DR-M1 is correct) the TTP will send the resolution to both C and M. Therefore, M will receive the payment twice but there is an assumption in this protocol that double spending of the same payment will be detected. As a result, it will be fair for both C and M; and M will not get advantage over C (Figure 16).

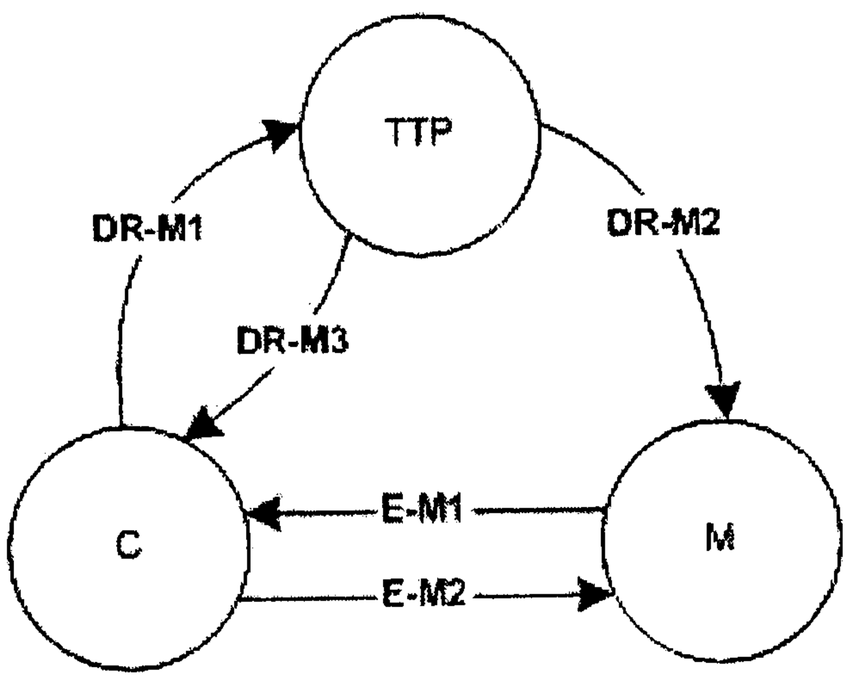

Figure 16: ECH Protocol, Scenario 6

7. After C received E-M1 from M, C found that E-M1 is correct and hence C sends E-M2 to M. Then, M found that E-M2 is correct. Hence, M sends E-M3 to C. After receiving E-M3 C contacted the TTP for resolution. However, the TTP found that DR-M1 is incorrect. Therefore, the TTP sends an abort message to C. There are two possibilities why C contacted the TTP in this scenario. The first one is because E-M3 is incorrect. The second one is because E-M3 is correct but C wants to see what they can receive from the TTP and thus trying to get an advantage over M. In both possibilities, C has to send correct DR-M1 to receive a resolution (Figure 17).

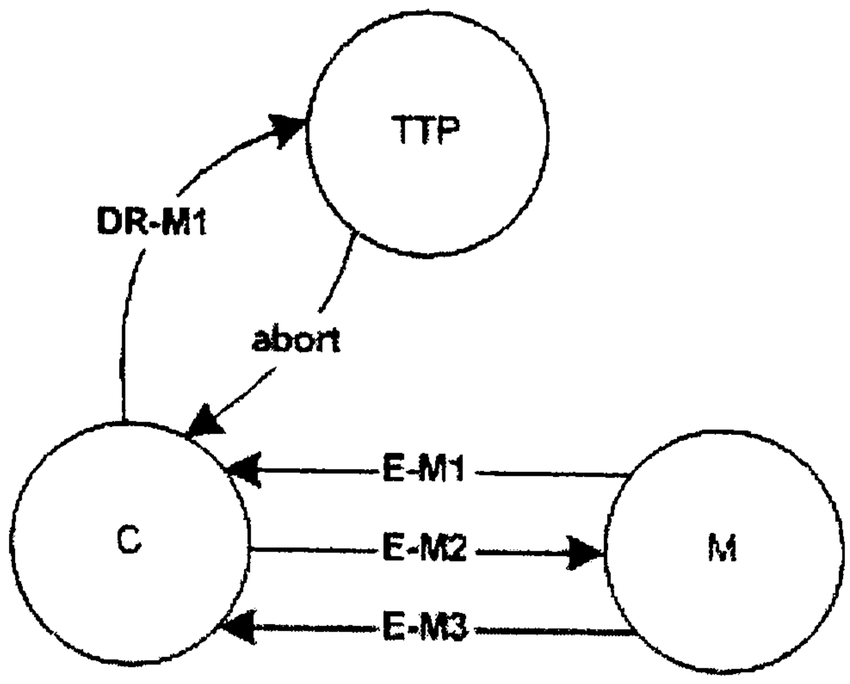

Figure 17: ECH Protocol, Scenario 7

8. The same as scenario number 7 but in here, DR-M1 that C sends to the TTP is correct. Therefore, the TTP resolves it by sending DR-M2 to M and DR-M3 to C (Figure 18).

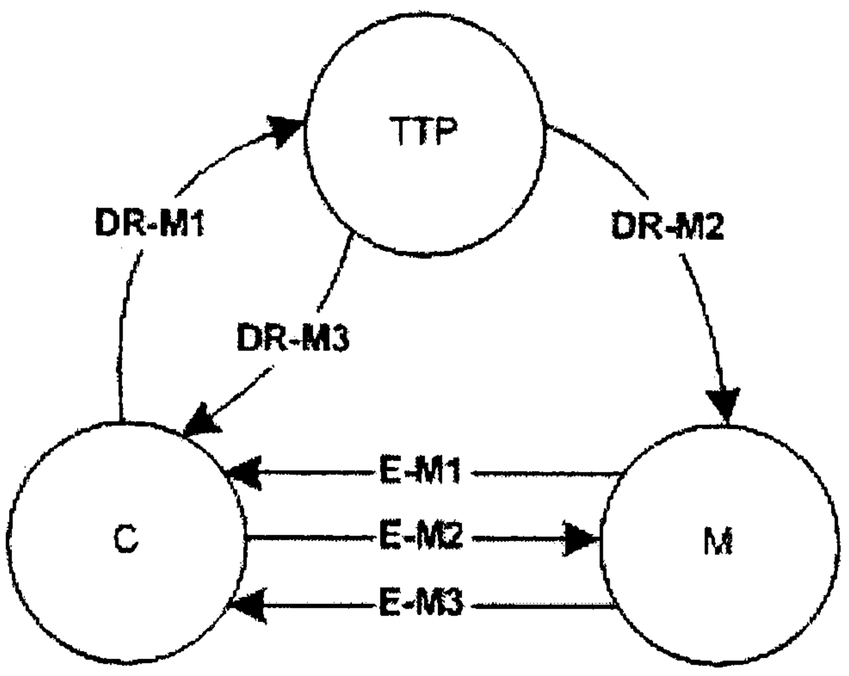

Figure 18: ECH Protocol, Scenario 8

Therefore, it is clear from the above scenarios that fairness is ensured for both C and M.

On the other hand, the following studies the scenarios where the messages (E-M1, E-M2, E-M3, DR-M1, DR-M2, and DR-M3) of the ECH protocol are sent but have not been received because of the failure in the communication channels between the parties involved in the protocols:

1) If E-M1 is not received by C then the fairness will not be compromised because no one revealed their item as C will not send E-M2 if they have not received a correct E-M1

2) If E-M2 is not received by M then M will consider that C is not interested in the exchange and hence M will not send E-M3. As a result, C will wait for E-M3 from M and as M has not received E-M2 from C then M will not send E-M3. Therefore, C will contact the TTP for resolution. However, there is a possibility of having the communication channel failed. So, this possibility will be studied later (see case 4)
3) If E-M3 is not received by C then C will consider it as M behaving dishonestly. As a result, C will contact the TTP for resolution. However, there is a possibility of having the communication channel failed. So, this possibility will be studied in the next case (see case 4)
4) If DR-M1 is not received by the TTP then C needs to re-contact the TTP asking for a resolution again
5) If DR-M2 is not received by M then there are two possibilities:
    a) If DR-M2 is not received by M (and M has not received a correct E-M2) and at the same time DR-M3 is not received by C then the fairness is not compromised
    b) If DR-M2 is not received by M but DR-M3 is received by C then the fairness is compromised if and only if M has not received a correct E-M2
6) If DR-M3 is not received by C then there are two possibilities:
    a) If DR-M3 is not received by C and at the same time DR-M2 is not received by M (and also M has not received a correct E-M2) then the fairness is not compromised
    b) If DR-M3 is not received by C but DR-M2 is received by M then the fairness is compromised if and only if C has not received a correct E-M3 i.e. when C received a correct E-M3 they contacted the TTP

As can be seen in these cases, the communication channels between C and TTP, and M and TTP should be resilient for the fairness to be ensured. To ensure fairness even in the case of communication channels failure the fault tolerance techniques need to be applied to the ECH protocol. However, this is out of the scope of this thesis. Hence, it is left as a future work.

5.4 Summary

The ECH protocol is presented and analysed in this chapter. It has been shown that the number of messages is reduced when we apply the technique of having a trustworthy merchant and enforcing the customer to be honest in the fair exchange protocols. That is, only three messages are exchanged between the customer and the merchant. Additionally, only three messages are executed in order for the dispute to be resolved.

It has been shown in this chapter that each party (customer and merchant) will be able to detect the dishonesty of the other party. If a party detects that the other party is dishonest then they will not send their item to them. All dispute possibilities for the ECH protocol have been presented. The possible scenarios of executing the ECH protocol have been discussed.

Chapter Six

6. Enforcing Merchant Honesty Protocol

6.1. Introduction

This chapter presents a fair exchange protocols that is based on having a trustworthy customer and then enforces the merchant to be honest. In this chapter the protocol will be discussed and analysed.

6.2. EMH Protocol Description

6.2.1. Description

The purpose of the Enforcing Merchant Honesty (EMH) protocol is for exchanging a digital product D for a payment. The EMH protocol enforces the merchant to act honestly. Its basic idea is to have one trustworthy party (C) and enforce the other party (M) to be honest.

The trustworthiness of C is governed by two factors; the payment certificate (P-Cert) issued by the CB; and the shared public key certificate (C.ct) issued by the TTP. Thus, the payment that will be sent by C is certified by the CB; and the public key to be used by C to encrypt the key that encrypts the payment is certified by the TTP.

The enforcement of M to be honest is governed by letting M encrypt the digital product using a key that is sent to M by C. This is to let C be able to decrypt the digital product without assistance from M. Hence, M will send a correct encrypted digital product because C will be able to decrypt it as soon they receive it.

In essence this protocol is like C saying to M: this is the encrypted payment; and I am a trustworthy as these certificates (P-Cert and C.ct) show BUT be honest with me by sending the correct digital product in order to receive the decryption key for the payment. By this means C is trustworthy by having the correct certificates from the CB and the TTP, and also M is enforced to be honest.

The EMH protocol consists of two phases. The pre-exchange phase and the exchange phase. In the pre-exchange phase, the customer gets the payment and the certificates (from trusted authorities) to be used in the exchange phase. In the exchange phase, the customer and the merchant exchange their items. It is not necessarily that the exchange phase comes immediately after the pre-exchange phase.

6.2.2. Pre-exchange Phase

In the pre-exchange phase (Figure 19), C needs to get the certificate C.ct of the shared public key from the TTP to be used to encrypt the key that is used to encrypt payment. Therefore, C needs to request the shared public key certificate from the TTP and then the TTP will send it (that is in messages PE-M1a and PE-M1b of Figure 19). C also needs to get the payment and its certificate P-Cert from the CB (that is in messages PE-M2a and PE-M2b of Figure 19).

M also needs to get a special version of digital product certificate (*DG-Cert*) from the CA. This *DG-Cert* is assumed to be publicly available, for example in M's website. Therefore, C can get it easily to help them in specifying what they want from M in the exchange phase that will be discussed later. The *DG-Cert* is unique for each digital product and it is issued once and can be used as many times as possible for the certified digital product. The content of this *DG-Cert* is shown in the Notation section.

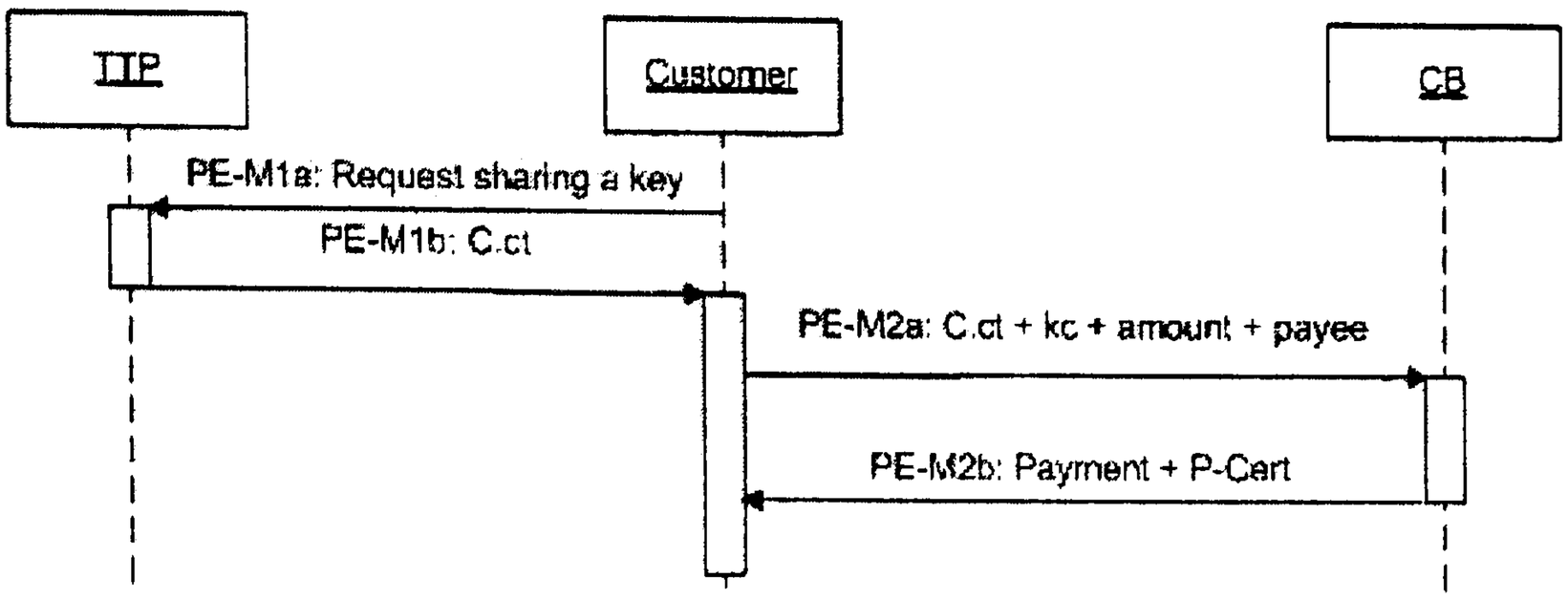

Figure 19: EMH Pre-exchange Phase

In this protocol C and TTP share a public key. The keys of the parties are as follows:

- Each party *x* (*x* ∈ C, M TTP and CB) has its own public (*pkx*) and private (*skx*) keys.
    - The CB's public key is denoted as pkcb = (ecb, ncb) and its corresponding private key is denoted as skcb = (dcb, ncb).
    - The TTP's public key is denoted as pkt = (et, nt) and its corresponding private key is denoted as skt = (dt, nt).
    - C's public key is denoted as pkc = (ec, nc) and its corresponding private key is denoted as skc = (dc, nc).
    - M's public key is denoted as pkm = (em, nm) and its corresponding private key is denoted as skm = (dm, nm)
- The shared public key between C and TTP is denoted as pkct = (ect, nct) and its corresponding private key is denoted as skct = (dct, nct)

6.2.3. The Exchange Phase

It is assumed that the exchange phase will take place after C finds the requested digital product (D) with M, and that C and M agree on the digital product and negotiated the price. Hence this phase is about the actual exchange of payment and digital product D.

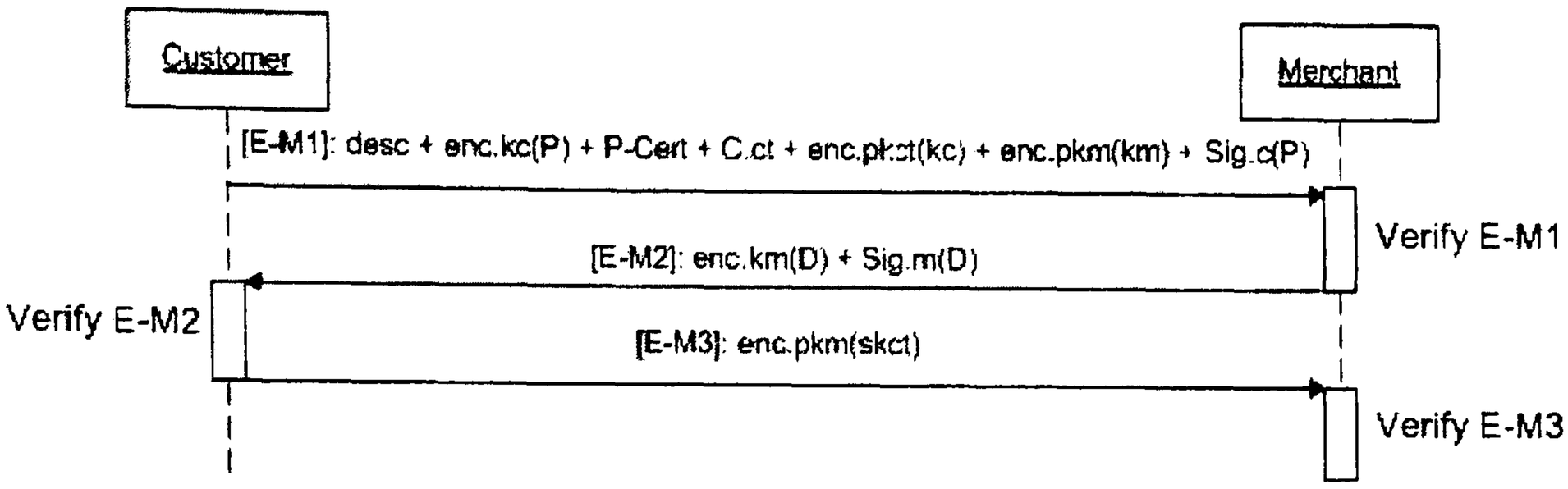

Figure 20: EMH Exchange Phase

There are only three messages to be exchanged between M and C in the exchange phase (Figure 20). These three messages are as follows:

[E-M1] **C → M: desc + enc.kc(P) + P-Cert + C.ct + enc.pkct(kc) + enc.pkm(km) + Sig.c(P)**

C sends the first message E-M1 to M which contains the following:

- **desc**: specifies what C wants from M. desc includes two fields, they are
  - description of D that C wants. The description can be the digital product ID number
  - hDG: hash value of D. C can get it from DG-Cert that is assumed to be publicly available

  Note that, **desc** is signed by C but for simplicity C's signature is omitted
- **enc.kc(P)**: the payment that is encrypted with the key *kc* (note that *kc* generated by C)
- **P-Cert**: the payment certificate that is issued by the CB
- **C.ct**: the shared public key certificate that is issued by the TTP
- **enc.pkct(kc)**: the key *kc*, used to encrypt the payment, is encrypted using the shared public key *pkct* that is certified in *C.ct*
- **enc.pkm(km)**: a key *km* that will be used by M to encrypt the digital product. *km* is encrypted using the public key of M (*pkm*). The reason for encrypting *km* using *pkm* is to prevent any other party from gaining *km* as it will be used to encrypt the digital product by M

- **Sig.c(P)**: C's signature on the payment. This signature can serve as non-repudiation of origin which allows M to be sure that the payment is sent by C. C's signature on the payment is the encryption of the hash value of the payment using C's private key skc

[E-M2] **M → C: enc.km(D) + Sig.m(D)**

On receiving message E-M1 from C, M checks the correctness of *enc.kc(P)*, *enc.pkct(kc)*, *Sig.c(P)*, *P-Cert* and *C.ct*. The correctness of *P-Cert* can be checked by verifying CB's signature on *P-Cert* and the correctness of C.ct can be checked by verifying TTP's signature on *C.ct*.

To check the correctness of the payment (*P*), M needs to check two things which are the payment itself and the encrypted payment with *kc* i.e. *enc.kc(P)*. Firstly, to check the correctness of *P*, M needs to get the hash value of payment (calculated as *HP*) by decrypting *Sig.c(P)* contained in message E-M1 using C's public key *pkc* (the public keys of all parties are publicly available) and then compare it with hash value of *P* (*hP*) contained in *P-Cert* i.e. to check the following:

$$HP \stackrel{?}{=} hP$$

If they are the same then M can be sure that C signed the correct *P*.

Secondly, to check the correctness of the encrypted payment *enc.kc(P)*, M computes the hash value of *enc.kc(P)* (calculated as *HeP*) and then compare it with the hash value of encrypted payment with *kc* i.e. *heP* which is contained in *P-Cert* (note that it is assumed that M will use the same function used by the CB to compute the hash value) i.e. to check the following:

$$HeP \stackrel{?}{=} heP$$

If they are the same then M can be sure that C encrypted the payment using *kc* and not another key.

M also needs to check the correctness of *kc* which is used to encrypt *payment*. To do so, M computed the hash value of *enc.pkct(kc)* (calculated as *HeKc*) and then compare it with *heKc* that is included in P-Cert, so M will check the flowing:

$$HeKc \stackrel{?}{=} heKc$$

If they are compared then M can be sure that the encrypted key is *kc* and not another key. The point here is to make sure that C is honest by sending the key used to encrypt the payment.

Therefore, if all comparisons are correct then, at this point, M will have the following fact. The encrypted payment is correct (i.e. it is the one certified in P-Cert) and it is indeed encrypted with *kc*. In addition, the key encrypted using the shared public key *pkct* is indeed *kc* and not another key. The shared public key *pkct* used to encrypt *kc* is certified by TTP. Therefore, once M got the private key of the shared public key then M will be able to get the payment (by first decrypting *enc.pkct(kc)* to get *kc* and then decrypting *enc.kc(P)* using *kc*).

At this point M must be sure that the encrypted payment matches their requirements, otherwise they will be at risk if they send message E-M2 to C as C already has the decryption key for D. Also, M must send D that matches C's requirement i.e. that matches *desc* included in E-M1.

Now, it is M's choice to complete the exchange or abort the protocol. If M wants to exchange D with the payment then M sends the following:

- **enc.km(D):** M first needs to get *km* by decrypting *enc.pkm(km*) included in E-M1 using their private key skm as *km* was encrypted using M's public key. Once M got *km* then they encrypt D using the key *km*
- **Sig.m(D)**: M's signature on D. This signature can serve as non-repudiation of receipt (because if M did not receive the encrypted payment then they will not send E-M2) which allows C to be sure that M has received the encrypted payment (it also serves as non-repudiation of origin)

Note that if M decides to abort the transaction after receiving message E-M1 and before sending message E-M2 to C then neither M nor C lose anything. But once M sends a correct message in E-M2 to C then the transaction must be completed and the protocol will guarantee that the exchange of D and P will be fair.

[E-M3] **C → M: enc.pkm(skct)**

On receiving message E-M2, C decrypts D using the key *km* (*km* is originated by C and sent to M in E-M1 to allow M to encrypt D) and then checks whether it is correct. If M encrypted D using different key then C ignores the transaction and aborts the protocol. If however D is correct then C sends the decryption key *skct* (it is encrypted using M's public key to prevent any other party from gaining *skct*) to M to be able to decrypt *kc* and then decrypt the payment.

It is clear that if D is incorrect then C will not send *skct* to M because it is M's responsibility to send the correct D to be able to receive the decryption key in message E-M3. Using this method this protocol enforces M to be honest and hence sends the correct D to C. If M sends incorrect D, then C can ask M to re-send the correct D to be able to receive the decryption key, but it depends on C if they want to do so.

6.2.4. Dispute Resolution

All disputes, if any, will come from M since C will not have to raise disputes as they receive the encrypted D and decrypt it using the key *km* before they send the decryption key to M. Therefore, the weakest link in this exchange is M as they have to send the correct D in order to receive the decryption key for the encrypted payment that they have received in message E-M1.

Therefore, if M has a dispute, then the following three messages are sent (Figure 21):

[DR-M1] **M → TTP: desc + P-Cert + C.ct + Sig.c(P) + enc.pkt(km) + enc.km(D)**

In case M has a dispute, they need to send to TTP the following: (*desc + P-Cert + C.ct + Sig.c(P) + enc.pkt(km) + enc.km(D)*).

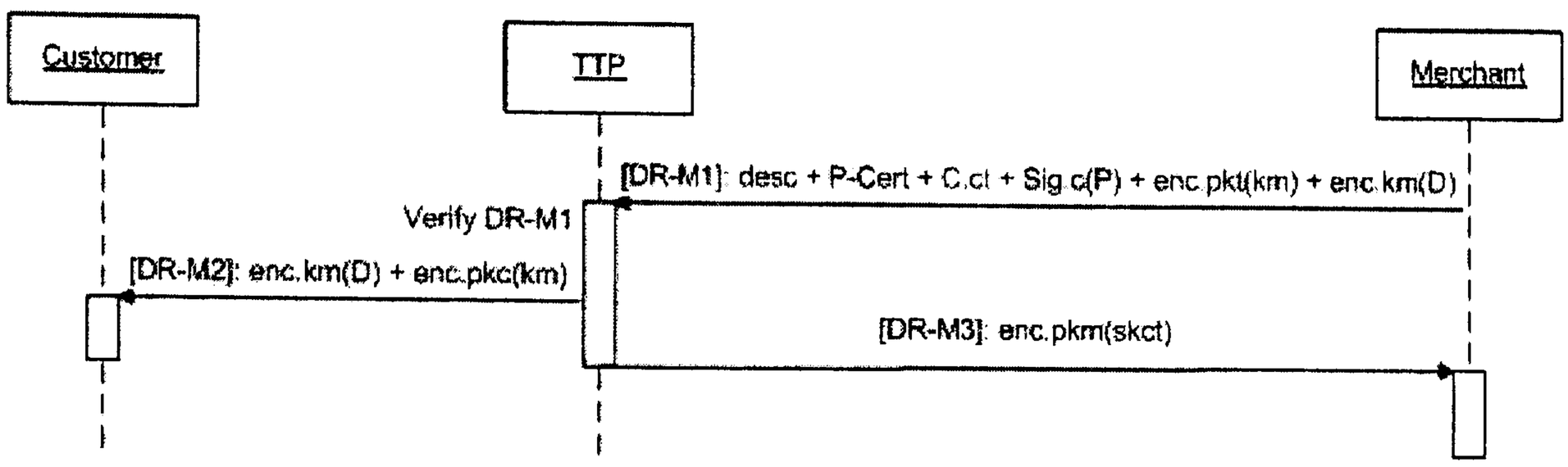

Figure 21: EMH Dispute Resolution

[DR-M2] **TTP → C: enc.km(D) + enc.pkc(km)**

On receiving message DR-M1 above, the TTP checks the correctness of *P-Cert* and *C.ct* by checking their signatures. In addition, the TTP verifies C's signature on *desc*. Then, the TTP decrypts *enc.pkt(km)* using their private key *skt* to get *km* then uses *km* to decrypt D.

If all of these are correct then TTP computes the hash value of D then compare it with hDG included in *desc*. This comparison makes TTP be sure that C ordered this D and not another D. The TTP will also compare the hash value included in *sig.c(P)* with the hash value of payment included in *P-Cert* (i.e. hP). If the TTP found the comparisons are correct then the TTP forwards D (that is encrypted using *km*) and the key km to decrypt D (*km* is sent to C encrypted using C's public key *pkc* to prevent any other party from gaining *km*) to C.

There are many reasons for forwarding the encrypted D and *km* to C. They are as follows:

- M may have sent to C incorrect D in message E-M2 (in the exchange phase)
- M may have not sent D at all; i.e. M contacted TTP before sending E-M2 to C
- M may have encrypted D in E-M2 using different *km*; therefore, the TTP forwards the key *km* that decrypts D

Otherwise, if the TTP found that the two hash values (i.e. the hash value of D with hDG and the hash value of P with hP) do not match or the other verifications are incorrect then the TTP sends an abort message to M. If the DR-M1 is incorrect then TTP will not contact C at all i.e. the TTP will not send DR-M2 to C.

[DR-M3] **TTP → M: enc.pkm(skct)**

OR

TTP → M: aborts;

The same process done for the message DR-M2 above. That is, if the TTP found that the hash value of D matches hDG included in desc and also the hash value of P matches hP and also the other verifications are correct then the TTP sends to M the decryption key *skct* that is encrypted using M's public key *pkm*. Otherwise, if the two hashes do not match then the TTP sends an abort message to M.

It is clear that if either M has not sent the correct D to C in E-M2, M has not sent D at all or M encrypted D in E-M2 using different *km* then M will not get an advantage over C because the TTP will check D and see whether it meets what C wants. If the hash value of D matches hDG then the TTP is sure that C ordered D from M and hence will send the decryption key *skct* to M and will forward the encrypted D and its decryption key *km* to C to ensure fairness in all cases. Therefore, the fairness is guaranteed for both M and C. However, if D is incorrect (e.g. the hash value of D does not match hDG) then the TTP will reject M's request for a dispute.

As can be seen in the dispute resolution phase, the TTP does not need to have both C and M to be involved in order for the dispute to be resolved; rather only the disputant (M in this protocol) and the TTP will be involved. That is, the TTP does not need to contact C to verify whether or not they have received the correct D; rather the

TTP asks M to provide all needed information (in DR-M1) and will be able to make the resolution. C will only be contacted by the TTP if the dispute has a resolution. Therefore, this will reduce the number of messages needed to resolve disputes.

6.3. EMH Protocol Analysis

In this section, the EMH protocol will be analysed. The analysis includes the ability of parties to detect the dishonesty of one another, disputes analysis, and presenting and discussing all possible scenarios.

6.3.1. Detection of Dishonesty

The same analysis that has been done for the ECH protocol will be done for the EMH protocol, to test the ways in which the customer and the merchant may act dishonestly.

There is only one way in which the merchant may act dishonestly. It is by sending incorrect E-M2. Incorrect E-M2 can be constructed by:

(a) Including incorrect digital product (either not the same as the one C wanted or a broken digital product),

(b) Encrypting the digital product with a key that C does not know of (a key that is not *km*),

(c) Including incorrect signature (i.e. *Sig.m(D)*).

On receiving message E-M2, if E-M2 is incorrect then C will easily detect that M is dishonest and then will not send E-M3 to M.

On the other hand, the cases in which the customer may act dishonestly are by :

1. sending an incorrect E-M1;
2. sending an incorrect E-M3;
3. not sending E-M3

In case 2 (i.e. sending an incorrect E-M3) there is only one possibility (by which C may act dishonestly) and this is where the decryption key sent by C to M in E-M3 is incorrect. In case 3 (i.e. not sending E-M3) there is also one possibility which is C not sending E-M3 to M. Case 1 (i.e. sending an incorrect E-M1) has a number of possibilities. These possibilities will be studied as follows.

Message E-M1 includes the following: *enc.kc(P), P-Cert, C.ct, enc.pkct(kc), enc.pkm(km),* and *Sig.c(P)*. M does not need to check the correctness of *enc.pkm(km)* because it includes the key *km* that M will use to encrypt the digital product D if they are interested in the exchange. So, M will check the correctness of the other five items (which are *enc.kc(P), P-Cert, C.ct, enc.pkct(kc),* and *Sig.c(P)*). C may act dishonestly by including incorrect items in E-M1. It is important to study the all possibilities in which C may act dishonestly in E-M1.

Note that *P-Cert* and *C.ct* are not included in this discussion because they will be checked first by M and if they are incorrect then M will not check the correctness of *enc.kc(P), enc.pkct(kc)* and *Sig.c(P)*. However, if both *P-Cert* and *C.ct* are correct then M will check the correctness of *enc.kc(P), enc.pkct(kc)* and *Sig.c(P)*.

The possibilities for the items *enc.kc(P), enc.pkct(kc)* and *Sig.c(P)* in E-M1 are:

A. *enc.kc(P)*: there are two things involved in forming *enc.kc(P)*. They are *P* and *kc*. So, four possibilities are available for forming *enc.kc(P)*:

1) *P* is correct and *kc* is correct. So, C is honest
2) *P* is incorrect and *kc* is incorrect. So, C is dishonest
3) *P* is correct and *kc* is incorrect. So, C is dishonest
4) *P* is incorrect and *kc* is correct. So, C is dishonest

B. *enc.pkct(kc)*: there are two things involved in forming *enc.pkct(kc)*. They are *kc* and *pkct*. So, four possibilities are available for forming *enc.pkct(kc)*:

1) *kc* is correct and *pkct* is correct. So, C is honest
2) *kc* is incorrect and *pkct* is incorrect. So, C is dishonest
3) *kc* is correct and *pkct* is incorrect. So, C is dishonest
4) *kc* is incorrect and *pkct* is correct. So, C is dishonest

C. *Sig.c(P)*: this signature means the hash value of payment (*P*) encrypted with C's private key. Therefore, there are two things involved in forming *Sig.c(P)*. They are *hash value of P* and *C's private key*. Therefore, there are four possibilities for forming *Sig.c(P)*:

1) *hash value of P* is correct and *C's private key* is correct. So, C is honest
2) *hash value of P* is incorrect and *C's private key* is incorrect. So, C is dishonest

3) *hash value of P* is correct and *C's private key* is incorrect. So, C is dishonest
4) *hash value of P* is incorrect and *C's private key* is correct. So, C is dishonest

	enc.kc(P)	enc.pkct(kc)	Sig.c(P)
1	√	√	√
2	¬	¬	¬
3	¬	√	√
4	√	¬	√
5	√	√	¬
6	√	¬	¬
7	¬	√	¬
8	¬	¬	√

Table 5: Possibilities of items in E-M1 of the EMH protocol

Table 5 shows the possibilities of combining *enc.kc(P)*, *enc.pkct(kc)*, and *Sig.c(P)* in E-M1. Symbol (√) means correct and (¬) means incorrect. The cases in Table 5 are analysed as follows (the numbers below refers to the numbers in Table 5), (Note that having one incorrect in E-M1 means that C is dishonest and M must not send the digital product to C. However, all cases will be studied).

1. **enc.kc(P), enc.pkct(kc), Sig.c(P):**
   - The encrypted payment is the one certified in P-Cert and is encrypted with kc and not another key
   - The encrypted key is kc and is encrypted with the key that is certified in C.ct
   - The signed payment is the same as payment that is certified in P-Cert

Result: the hash values will be the same (i.e. all the verifications are correct) and hence C is honest.

2. **¬ enc.kc(P), ¬enc.pkct(kc), ¬Sig.c(P):**

- The encrypted payment is different from the one certified in P-Cert. This has the following possibilities:
  - Payment is different from the one certified in P-Cert and kc is also different from the one certified in P-Cert
  - payment is the same as certified in P-Cert but kc is different from the one certified in P-Cert
  - payment is different from the one certified in P-Cert and kc is the one certified in P-Cert
- The encrypted key is not the one certified in P-Cert. This has the following possibilities:
  - kc is different from the one certified in P-Cert and pkct is different from the one certified in C.ct
  - kc is the same as the one certified in P-Cert but pkct is different from the one certified in C.ct
  - kc is different from the one certified in P-Cert and pkct is the one certified in C.ct
- The signed payment is not the same as payment that is certified in P-Cert. This has the following possibilities:
  - The hash value of the payment is incorrect and C's *private key* is incorrect
  - The hash value of the payment is correct but C's *private key* is incorrect
  - The hash value of the payment is incorrect and C's *private key* is correct

Result: it is clear that M will easily detect the problems in this E-M1 by different ways. First, M can detect that C is dishonest when M computes the hash value for enc.kc(P) and compares it with the hash value of encrypted payment (heP) included in P-Cert. Second, M can detect that C is dishonest when M computes the hash value for enc.pkct(kc) and compares it with the hash value of encrypted kc (heKc) included in P-Cert. Third, when M compares the hash value of the payment, that is in Sig.c(P), with the hash value of payment (hP) included in P-Cert. Therefore, M can detect that C is dishonest.

3. **¬enc.kc(P), enc.pkct(kc), Sig.c(P):**
    - The encrypted payment is different from the one certified in P-Cert. This has three possibilities (the possibilities are the same as the ones in case number 2 above)
    - The encrypted key is kc and is encrypted with the key that is certified in C.ct
    - The signed payment is the same as payment that is certified in P-Cert

Result: M can detect that C is dishonest when M computes the hash value for enc.kc(P) and compares it with the hash value of encrypted payment (heP) included in P-Cert.

4. **enc.kc(P), ¬enc.pkct(kc), Sig.c(P):**
    - The encrypted payment is the one certified in P-Cert and is encrypted with kc and not another key
    - The encrypted key is not the one certified in P-Cert. This has three possibilities (the possibilities are the same as the ones in case number 2 above)
    - The signed payment is the same as payment that is certified in P-Cert

Result: M can detect that C is dishonest when M computes the hash value for enc.pkct(kc) and compares it with the hash value of encrypted kc (heKc) included in P-Cert.

5. **enc.kc(P), enc.pkct(kc), ¬Sig.c(P):**
    - The encrypted payment is the one certified in P-Cert and is encrypted with kc and not another key
    - The encrypted key is kc and is encrypted with the key that is certified in C.ct
    - The signed payment is not the same as payment that is certified in P-Cert. This has three possibilities (the possibilities are the same as the ones in case number 2 above)

Result: M can detect that the signed payment is different from the one certified in P-Cert when M gets the hash value of payment using C's public key and then comparing this hash with the hash value of payment (hP) included in P-Cert. Note, having the signed payment different from the one certified in P-Cert will not compromise fairness because the most important parts that M must be sure that they match P-Cert are *enc.kc(P)* and *enc.pkct(kc)* because if they are compared to the ones in P-Cert then M is sure that they can get the decryption key from TTP in case C becomes dishonest. However, it is up to M if they want to proceed or not but having all verifications are correct in E-M1 but the signed payment is not the same as payment that is certified in P-Cert gives M a sign that C may be dishonest. This will not affect the fairness. However, if *Sig.c(P)* is incorrect then in case of dispute then M needs to send *Sig.c(P)* to the TTP but when the TTP finds it incorrect then the TTP will not resolve the dispute. Therefore, it is better not to send E-M2 to C if *Sig.c(P)* is incorrect.

6. enc.kc(P), ¬enc.pkct(kc), ¬Sig.c(P):

- The encrypted payment is the one certified in P-Cert and is encrypted with kc and not another key
- The encrypted key is not the one certified in P-Cert. This has three possibilities (the possibilities are the same as the ones in case number 2 above)
- The signed payment is not the same as payment that is certified in P-Cert. This has three possibilities (the possibilities are the same as the ones in case number 2 above)

Result: M can detect that C is dishonest when (a) M computes the hash value for enc.pkct(kc) and compares it with the hash value of encrypted kc (heKc) included in P-Cert; and (b) M gets the hash value of payment (i.e. from Sig.c(P)) using C's public key and then comparing this hash with the hash value of payment (hP) included in P-Cert

7. **¬enc.kc(P), enc.pkct(kc), ¬Sig.c(P):**
   - The encrypted payment is different from the one certified in P-Cert. This has three possibilities (the possibilities are the same as the ones in case number 2 above)
   - The encrypted key is kc and is encrypted with the key that is certified in C.ct
   - The signed payment is not the same as payment that is certified in P-Cert. This has three possibilities (the possibilities are the same as the ones in case number 2 above)

Result: M can detect that C is dishonest when (a) M computes the hash value for enc.kc(P) and compares it with the hash value of encrypted payment (heP) included in P-Cert; and (b) M gets the hash value of payment (i.e. from Sig.c(P)) using C's public key and then comparing this hash with the hash value of payment (hP) included in P-Cert

8. **¬enc.kc(P), ¬enc.pkct(kc), Sig.c(P):**
   - The encrypted payment is different from the one certified in P-Cert. This has three possibilities (the possibilities are the same as the ones in case number 2 above)
   - The encrypted key is not the one certified in P-Cert. This has three possibilities (the possibilities are the same as the ones in case number 2 above)
   - The signed payment is the same as payment that is certified in P-Cert

Result: M can detect that C is dishonest when (a) M computes the hash value for enc.kc(P) and compares it with the hash value of encrypted payment (heP) included in P-Cert; and (b) M computes the hash value for enc.pkct(kc) and compares it with the hash value of encrypted kc (heKc) included in P-Cert.

To sum up the cases appear in Table 5, it is clear from the possibilities discussed in Table 5 that if C tried to cheat on M by including incorrect items in E-M1 then M will be able to detect it and refuse to exchange the digital product with C.

Therefore, the fairness is ensured for both C and M. However, if C behaved honestly in E-M1 but disappeared before sending the decryption key in E-M3 or sent incorrect E-M3 then the TTP can guarantee the fairness and will send the decryption key to M if they find that M is honest.

6.3.2. Dispute Analysis

For the EMH protocol, the exchange between C and M is for the digital product (from M) and the decryption key for the encrypted payment (from C). This is because C sends the encrypted payment to M in message E-M1 and M will verify it and if satisfied then the exchange of the digital product (by M) and the decryption key (by C) will take place. The order of the exchange of the digital product and the decryption key in the EMH protocol is as follows. C will receive a correct digital product before M receives the decryption key.

	C	M	Result
	Receive digital product	**Receive decryption key**	
1	√	√	No dispute
2	X	√	Not applicable
3	√	X	M disputes
4	X	X	No dispute / Not applicable in EMH protocol / M's fault

Table 6: Disputes possibilities for EMH

Table 6 shows all possible cases for disputes for the EMH protocol. Each case is discussed below. In Table 6, (X) means either the party (C or M) has not received the item (digital product or decryption key) at all or they received incorrect item; whereas (√) means the correct item is received.

1) In case 1, both C and M receive the correct items (i.e. C receives the correct digital product and M receives the correct decryption key) from each other. Hence, there is no dispute.
2) In case 2, C has either received incorrect digital product or not received the digital product at all, and M received the correct decryption key. This case is not applicable in the EMH protocol because M has to send a correct digital product to be able to receive the correct decryption key
3) In case 3, C received a correct digital product, and M either received incorrect decryption key or has not received the decryption key at all. In this case M will dispute to TTP
4) In case 4, there are four possibilities which are:
    a) Both C and M have not received anything from each other. So, no dispute will be made as both of them have not revealed their items. This represents the case where M received E-M1 and did not send E-M2 to C
    b) Both C and M have received incorrect items from each other. That is, C received incorrect digital product and M received incorrect decryption key. This case is not applicable in the EMH protocol because M has to send a correct digital product to be able to receive the correct decryption key. So, if C found that the digital product is incorrect then C will not send the decryption key at all i.e. neither correct decryption key nor incorrect decryption key
    c) C received an incorrect digital product and M has not received the decryption key at all. This case is normal because if the digital product is incorrect then C will not send the decryption key to M. That is, M has to send a correct digital product to be able to receive the correct decryption key from C. Therefore, if this case occurs then for M to raise a dispute to the TTP, M needs to send to the TTP the correct digital product. If M sends the correct digital product to the TTP then the TTP will make a resolution to both C and M. That is, C will received two digital products (the incorrect digital product from M, and the correct digital product from TTP) and this is the penalty that M pays for being dishonest. However, if the TTP found that the digital product is incorrect then M's dispute will be rejected. Note that the incorrect digital product that C may have received from M means either a digital product that is not valid (not working software for example) or a

digital product that is not the one C wants (C wants a piece of music for artist X and received a piece of music for artist Y)

d) C has not received the digital product at all and M received incorrect decryption key. This case is not applicable in the EMH protocol because M has to send a correct digital product to be able to receive the correct decryption key. So, if C has not received the correct digital product then C will not send the decryption key at all

The design of the EMH protocol reduces the possibilities for having disputes. Additionally, only M will raise disputes as C will not send the decryption key unless the correct digital product is received from M. Hence, the EMH protocol enforces M to be honest. As a result, the possibilities for disputes are reduced by preventing them occurring.

In addition to the previous cases, the following cases are studied:

- M disputes that they have received incorrect payment. This is not possible because *P-Cert* guarantees that the payment is correct; and if M found that *payment* is incorrect or not the same as they requested then they should have not sent the digital product to C. So, it is M's fault if they send the digital product to C if they have any doubt about the correctness of the payment. Once M sends the digital product to C then this means that they are satisfied with the payment. Therefore, this dispute will not happen because M knows the rules of the protocol which allow M to check the payment before they send the digital product to C; and as a result M will not put themselves at risk
- It is clear that C will not raise a dispute because C will receive the digital product before they send the decryption key to M. However, the following cases are studied:
    - C claims that they have received incorrect digital product from M. This will not happen because M knows that if they send incorrect digital product then they will not receive the decryption key. Again, the idea of this protocol is to have one trustworthy party and enforce the other party to be honest if they are willing to exchange their item. Therefore,

if C received incorrect digital product then they will not send the decryption key to M i.e. C's item is not revealed and hence C does not need to dispute.

- C claims that they have not received the decryption key to decrypt digital product. This is not applicable in this protocol because the digital product is encrypted with *km* that is sent by C; and hence C is able to decrypt the digital product as soon as they get it. However, if C found that D is not encrypted with *km* then C will not send the decryption key to M in E-M3.

6.3.3. Scenarios Analysis

There are different scenarios for executing the EMH protocol by C and M. These scenarios include:

a) having both C and M are behaving honestly
b) C is behaving dishonestly and M is behaving honestly
c) M is behaving dishonestly and C is behaving honestly
d) having both C and M are behaving dishonestly

The possible scenarios for executing the EMH protocol are as follows:

1. C and M are honest which result in normal execution. That is, C sends a correct E-M1, M sends a correct E-M2, and C sends a correct E-M3 (Figure 22).

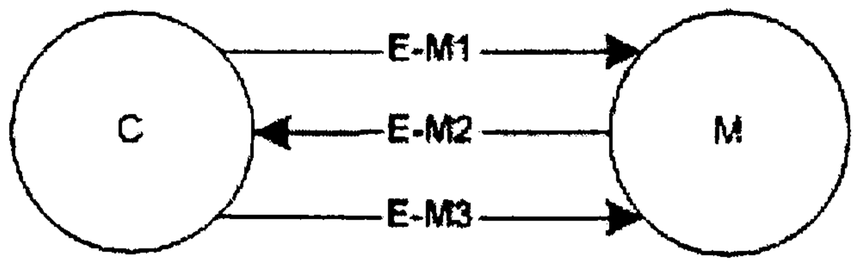

Figure 22: EMH Protocol, Scenario 1

2. After receiving E-M1, M quits the protocol because either E-M1 is incorrect or M is no longer interested in the exchange (Figure 23).

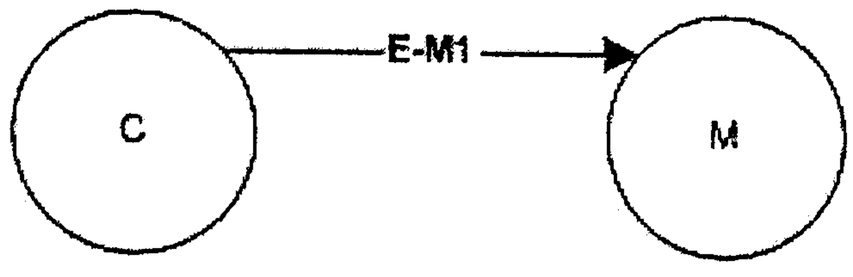

Figure 23: EMH Protocol, Scenario 2

3. After receiving E-M1, M contacts the TTP before sending E-M2 to C. However, the TTP found that DR-M1 is incorrect. Therefore, the TTP sends abort message to M. In this scenario M tries to cheat but they gained nothing (Figure 24)

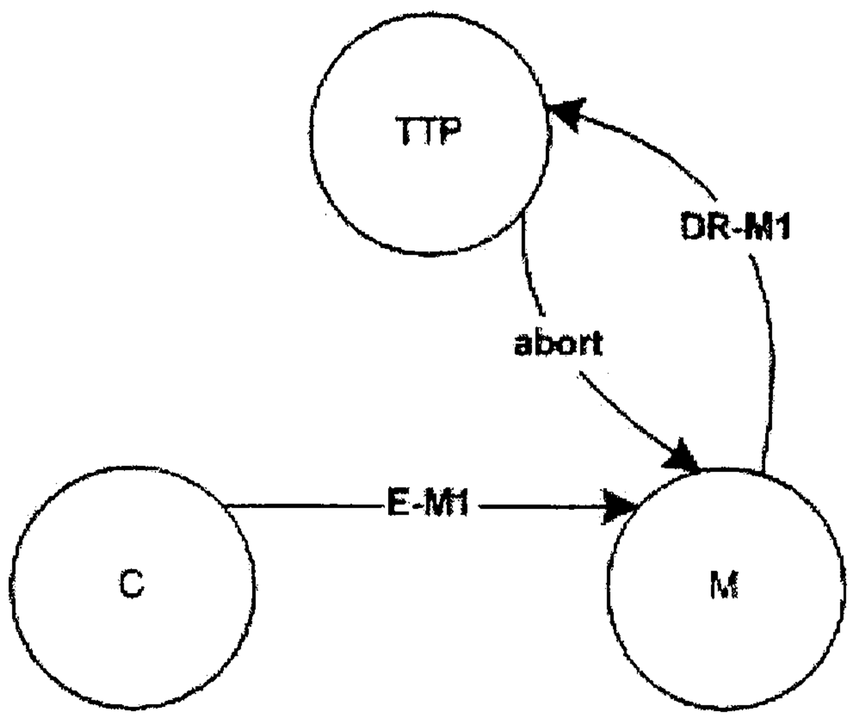

Figure 24: EMH Protocol, Scenario 3

4. The same as scenario number 3 but in here, the TTP found that DR-M1 is correct. Therefore, the TTP will make it fair for both C and M by sending the resolution to C in DR-M2 and also by sending the resolution to M in DR-M3. In this scenario M tries to cheat but TTP makes it fair for both C and M (Figure 25)

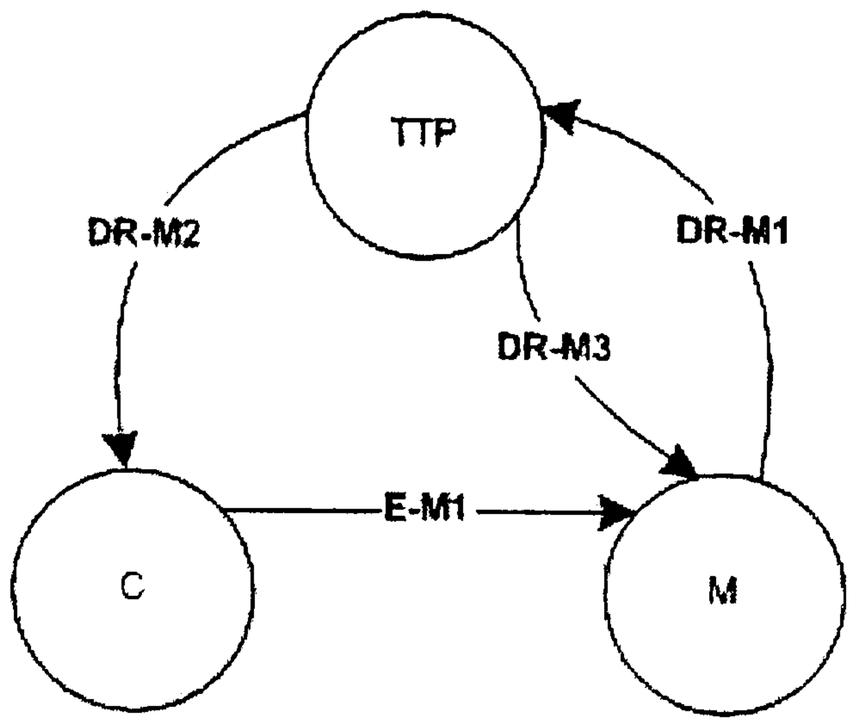

Figure 25: EMH Protocol, Scenario 4

5. After M received E-M1 from C, M found that E-M1 is correct and hence M sends E-M2 to C. M waited to receive E-M3 from C but nothing is received. Therefore, M contacted the TTP for resolution. However, the TTP found that DR-M1 is incorrect and hence the TTP sends an abort message to M. Note, there are two possibilities why C did not send E-M3 to M. These possibilities are either because C found that E-M2 is incorrect or because C is dishonest. If it is the former (where E-M2 is incorrect) then it is M's fault for sending an incorrect message to C. While, if it is the later (where E-M2 is correct but C is dishonest) then M needs to send correct DR-M1 to the TTP to be able to receive a resolution (Figure 26)

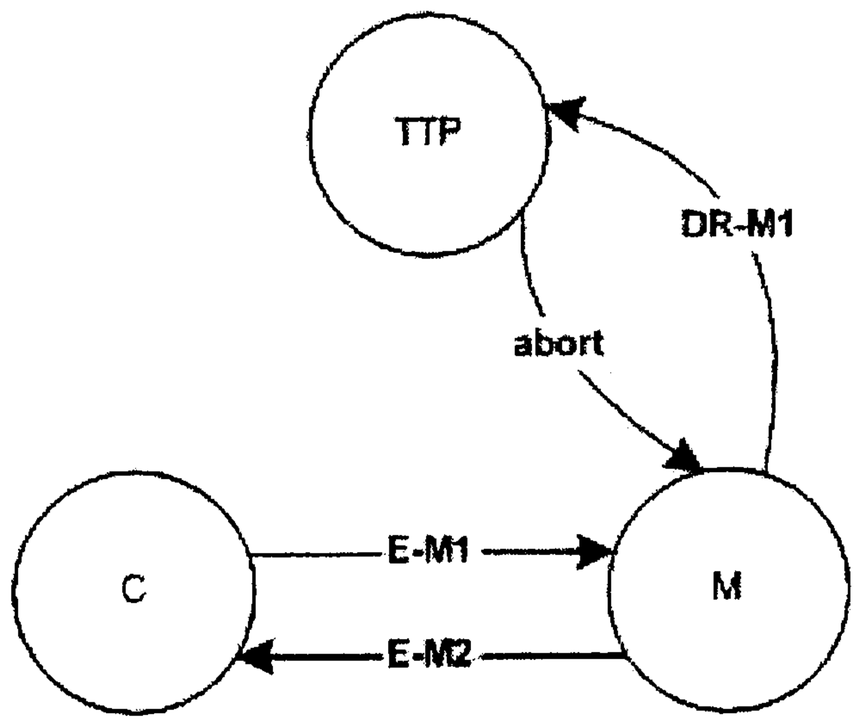

Figure 26: EMH Protocol, Scenario 5

6. The same as scenario number 5 but here, DR-M1 that M sent to the TTP is correct. Therefore, the TTP resolves it by sending DR-M2 to C and DR-M3 to M. Again, the reason for C not sending E-M3 to M is either because E-M2 is

incorrect or because C is dishonest. If it is because E-M2 is incorrect, then for M to receive the DR-M3 (which is the decryption key for the encrypted payment) then M has to send correct DR-M1 to the TTP. When the TTP received the correct DR-M1 then the TTP will make it fair for both C and M. While if C did not send E-M3 to M because C is dishonest, then M has to send the correct DR-M1 to the TTP for the dispute to be resolved (Figure 27).

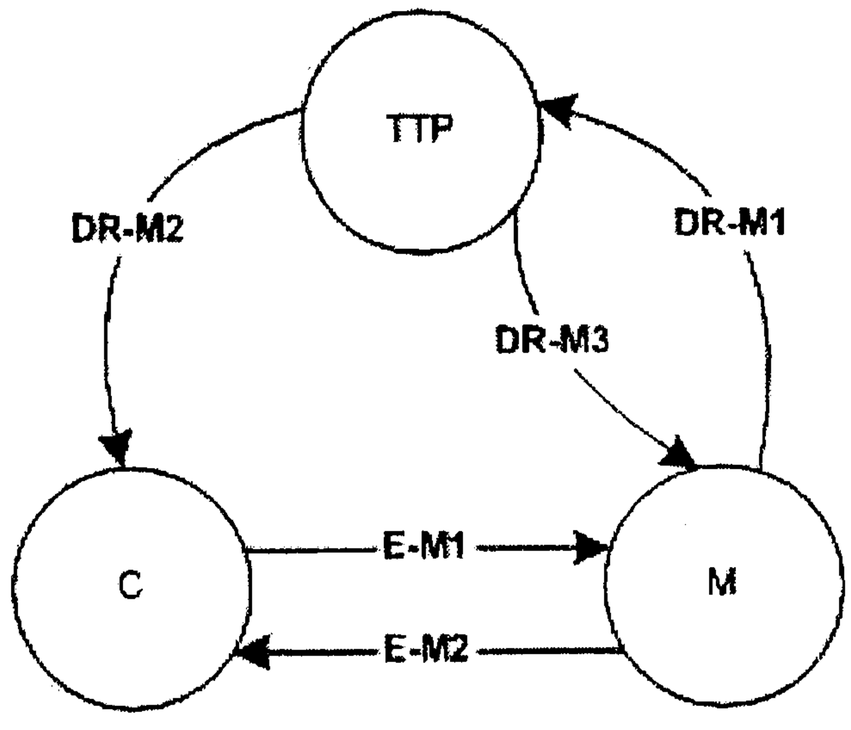

Figure 27: EMH Protocol, Scenario 6

7. After M received E-M1 from C, M found that E-M1 is correct and hence M sends E-M2 to C. Then, C found that E-M2 is correct. Hence, C sends E-M3 to M. After receiving E-M3, M contacted the TTP for resolution. However, the TTP found that DR-M1 is incorrect. Therefore, the TTP sends an abort message to M. There are two possibilities why M contacted the TTP in this scenario. The first one is because E-M3 is incorrect. The second one is because E-M3 is correct but M wants to see what they can receive from the TTP and thus trying to get an advantage over C. In both possibilities, M has to send correct DR-M1 to receive a resolution (Figure 28)

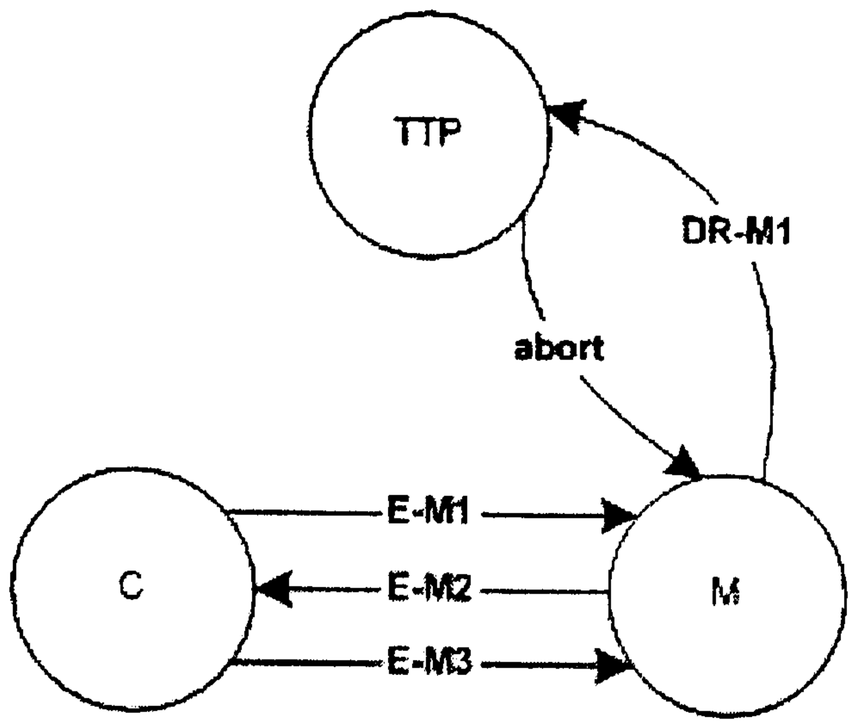

Figure 28: EMH Protocol, Scenario 7

8. The same as scenario number 7 but in here, DR-M1 that M sends to the TTP is correct. Therefore, the TTP resolve it by sending DR-M2 to C and DR-M3 to M (Figure 29)

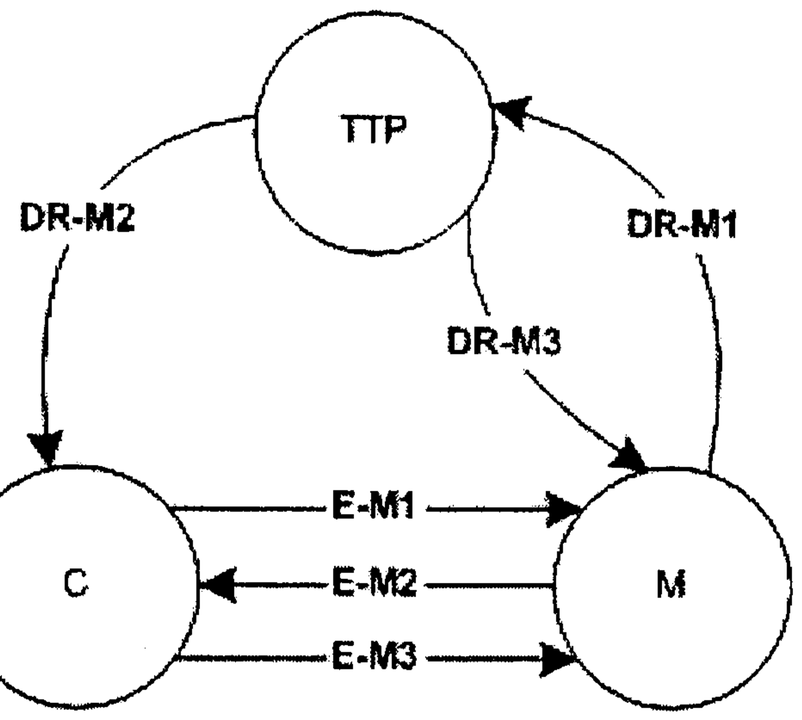

Figure 29: EMH Protocol, Scenario 8

Therefore, it is clear from the above scenarios that fairness is ensured for both C and M.

On the other hand, the following studies the scenarios where the messages (E-M1, E-M2, E-M3, DR-M1, DR-M2, and DR-M3) of the EMH protocol are sent but have not been received because of the failure in the communication channels between the parties involved in the protocols:

1) If E-M1 is not received by M then the fairness will not be compromised because no one revealed their item as M will not send E-M2

2) If E-M2 is not received by C then C will consider that M is not interested in the exchange and hence C will not send E-M3. As a result, M will wait for E-M3 from C and as if C has not received E-M2 from M then C will not send E-M3. Therefore, M will contact the TTP for resolution. However, there is a possibility of having failure in the communication channel. So, this possibility will be studied later (see case 4)
3) If E-M3 is not received by M then M will consider it as C behaving dishonestly. As a result, M will contact the TTP for resolution. However, there is a possibility of having the communication channel failed. So, this possibility will be studied in the next case (see case 4)
4) If DR-M1 is not received by the TTP then M needs to re-contact the TTP asking for a resolution again
5) If DR-M2 is not received by C then there are two possibilities:
    a) If DR-M2 is not received by C (and also C has not received a correct E-M2) and at the same time DR-M3 is not received by M then the fairness is not compromised
    b) If DR-M2 is not received by C but DR-M3 is received by M then the fairness is compromised if and only if C has not received a correct E-M2
6) If DR-M3 is not received by M then there are two possibilities:
    a) If DR-M3 is not received by M and at the same time DR-M2 is not received by C (and also C has not received a correct E-M2) then the fairness is not compromised
    b) If DR-M3 is not received by M but DR-M2 is received by C then the fairness is compromised if and only if M has not received a correct E-M3 i.e. when M received a correct E-M3 they contacted the TTP

As can be seen in these cases, the communication channels between C and TTP, and, M and TTP should be resilient for the fairness to be ensured. To ensure fairness even in the case of communication channels failure the fault tolerance techniques need to be applied to the EMH protocol. However, this is out of the scope of this thesis. Hence, it is left as a future work.

6.4. Summary

The EMH protocol is presented and analysed in this chapter. It has been shown that the number of messages is reduced when we apply the technique of having a trustworthy customer and enforcing the merchant to be honest in the fair exchange protocols. That is, only three messages are exchanged between the customer and the merchant.

It has been shown in this chapter that each party (customer and merchant) will be able to detect the dishonesty of the other party. If a party detects that the other party is dishonest then they will not send their item to them. All dispute possibilities for the EMH protocol have been presented. The possible scenarios of executing the EMH protocol have been discussed.

This chapter presented the EMH protocol where the merchant is enforced to be honest; whereas the previous chapter presented the ECH protocol where the customer is enforced to be honest. The same ideas that are applied in the ECH protocol and the EMH protocol were used in order to propose a fair exchange protocol that enforces both parties (customer and merchant) to be honest. However, it has been found that enforcing the honesty of the both parties is not possible in the optimistic fair exchange. This is because the party that is enforced to be honest is the party who reveals their unencrypted item first (or the item is encrypted with a key that the other party knows of). That is, in the ECH protocol the customer is the first party who reveals the payment that is encrypted with a key that the merchant knows of. Therefore, the customer is enforced to be honest in the ECH protocol. In the EMH protocol, however, the merchant is the first party who reveals the digital product that is encrypted with a key that the customer knows of. Therefore, the merchant is enforced to be honest in the EMH protocol. As a result, enforcing both parties to be honest (in the optimistic approach) is not possible as only one party has to reveal their item first.

Chapter Seven

7. Encouraging Customer and Merchant Honesty Protocol

7.1 Introduction

This chapter presents a fair exchange protocols that is based on having two trustworthy parties i.e. customer and merchant. The ideas used in the ECH and EMH protocols have been combined to produce a protocol that encourages (rather that enforces as explained in the previous chapter) the customer and merchant to be honest. In this chapter, this protocol will be discussed and analysed.

7.2 ECMH Protocol Description

7.2.1 Description

The Encouraging Customer and Merchant Honesty (ECMH) protocol is an optimistic fair exchange for exchanging a digital product D with a payment. The basic idea of the ECMH protocol is to provide the parties (C and M) with certificates that let the parties test the trustworthiness of one another. Therefore, there is no point for a party to be dishonest. Hence, the ECMH encourages the parties to be honest.

The trustworthiness of C is governed by two things which are the payment certificate (P-Cert) issued by the CB and the public key certificate (C.ct) issued by the TTP. Therefore, the payment that will be sent by C is certified by the CB; and the public key to be used by C to encrypt the key used to encrypt this payment is certified by the TTP. The trustworthiness of M is also governed by two things which are the digital product certificate (D-Cert) issued by the CA and the public key certificate (C.mt) issued by the TTP. Therefore, the digital product that will be sent by M is certified by the CA; and the public key to be used by M to encrypt the key used to encrypt this digital product is certified by the TTP. Therefore, this protocol encourages both C and M to be honest by sending correct items as each party will be able to detect if the received item is incorrect.

The ECMH protocol is like C and M exchanging their encrypted items (payment and digital product) and their certificates. These encrypted items and their certificates will test the trustworthy of each party. If the parties found that the other party is trustworthy then they will complete the exchange otherwise they abort it. Hence, there is no point of being dishonest.

The ECMH protocol consists of two phases. The pre-exchange phase and the exchange phase. In the pre-exchange phase, the merchant gets the digital product and the certificates (from trusted authorities) to be used in the exchange phase. Additionally, the customer gets the payment and the certificates (from trusted authorities) to be used in the exchange phase. In the exchange phase, the customer and the merchant exchange their items. It is not necessarily that the exchange phase comes immediately after the pre-exchange phase.

7.2.2 Pre-exchange phase

In the pre-exchange phase (Figure 31), C needs to get the certificate C.ct of the shared public key from the TTP to be used to encrypt the key used to encrypt the payment (that is in messages PE-M1a and PE-M1b of Figure 31). C also needs to get the payment and its certificate P-Cert from the CB (that is in messages PE-M2a and

PE-M2b of Figure 31). Also in the pre-exchange phase (Figure 30) M needs to get the certificate C.mt of the shared public key from the TTP to be used to encrypt the key used to encrypt D (that is in messages PE-M1a and PE-M1b of Figure 30). M also needs to get the digital product (D) and its certificate D-Cert from the CA (that is in messages PE-M2a and PE-M2b of Figure 30), (the CA can be thought of as the producer of the digital product).

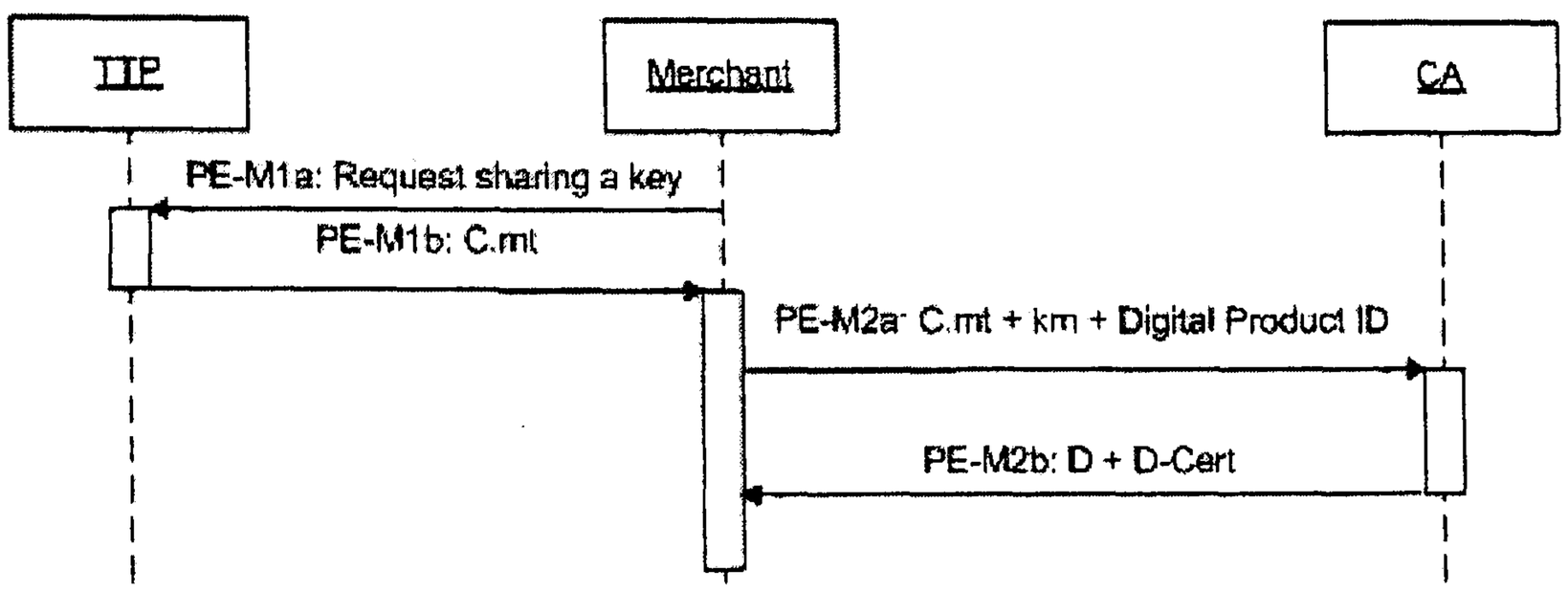

Figure 30: ECMH Pre-exchange Phase (1)

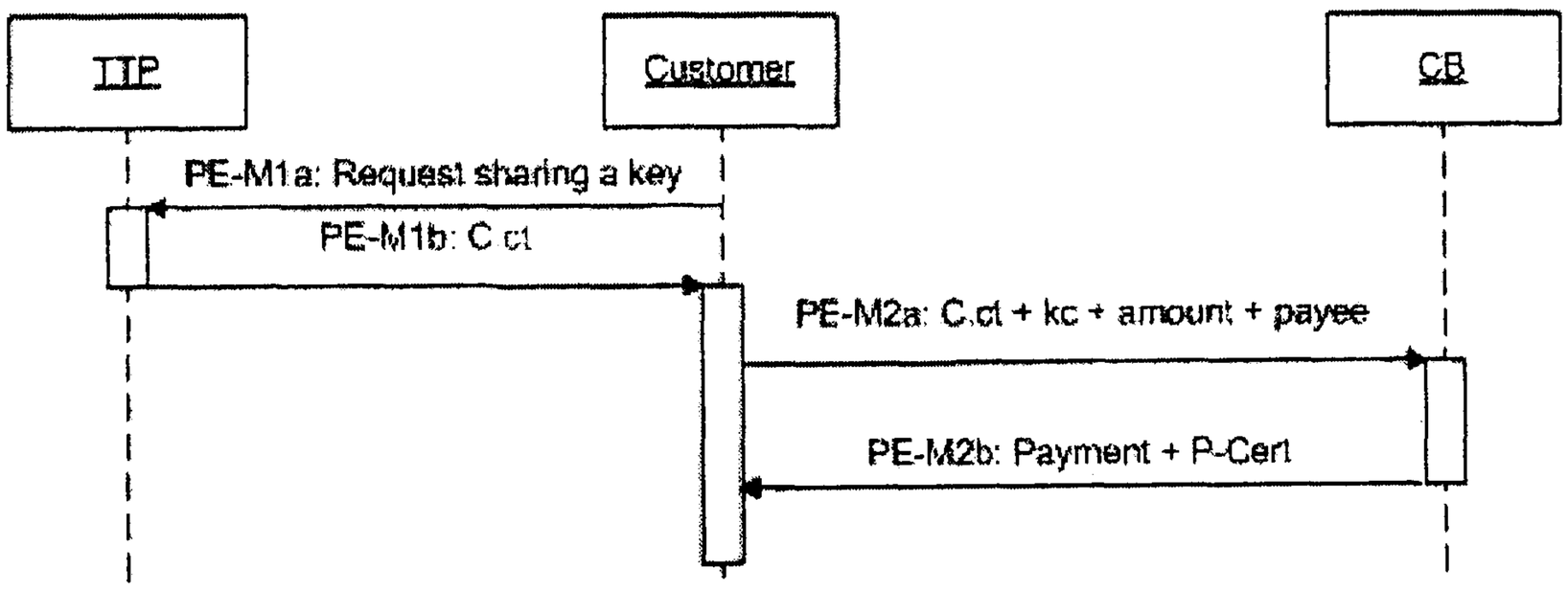

Figure 31: ECMH Pre-exchange Phase (2)

In this protocol, there are two public keys to be shared. The first one is shared between the TTP and C. The other one is shared between the TTP and M. The way in which these keys are shared is as follows.

- Each party x ($x \in$ C, M, CB, CA and TTP) has its own public (pkx) and private (skx) keys.
  - The CB's public key is denoted as pkcb = (ecb, ncb) and its corresponding private key is denoted as skcb = (dcb, ncb).

- The CA's public key is denoted as pkca = (eca, nca) and its corresponding private key is denoted as skca = (dca, nca).
- The TTP's public key is denoted as pkt = (et, nt) and its corresponding private key is denoted as skt = (dt, nt).
- C's public key is denoted as pkc = (ec, nc) and its corresponding private key is denoted as skc = (dc, nc)
  - M's public key is denoted as pkm = (em, nm) and its corresponding private key is denoted as skm = (dm, nm).

- The shared public key between C and the TTP is denoted as pkct = (ect, nct) and its corresponding private key is denoted as skct = (dct, nct).
- The shared public key between M and the TTP is denoted as pkmt = (emt, nmt) and its corresponding private key is denoted as skmt = (dmt, nmt).

7.2.3 The Exchange Phase

It is assumed that the exchange phase will take place after C finds the wanted digital product (D) with M. It is also assumed that this phase will take place after C and M agree on the digital product and negotiated the price. Hence this phase is about the actual exchange of payment and digital product D.

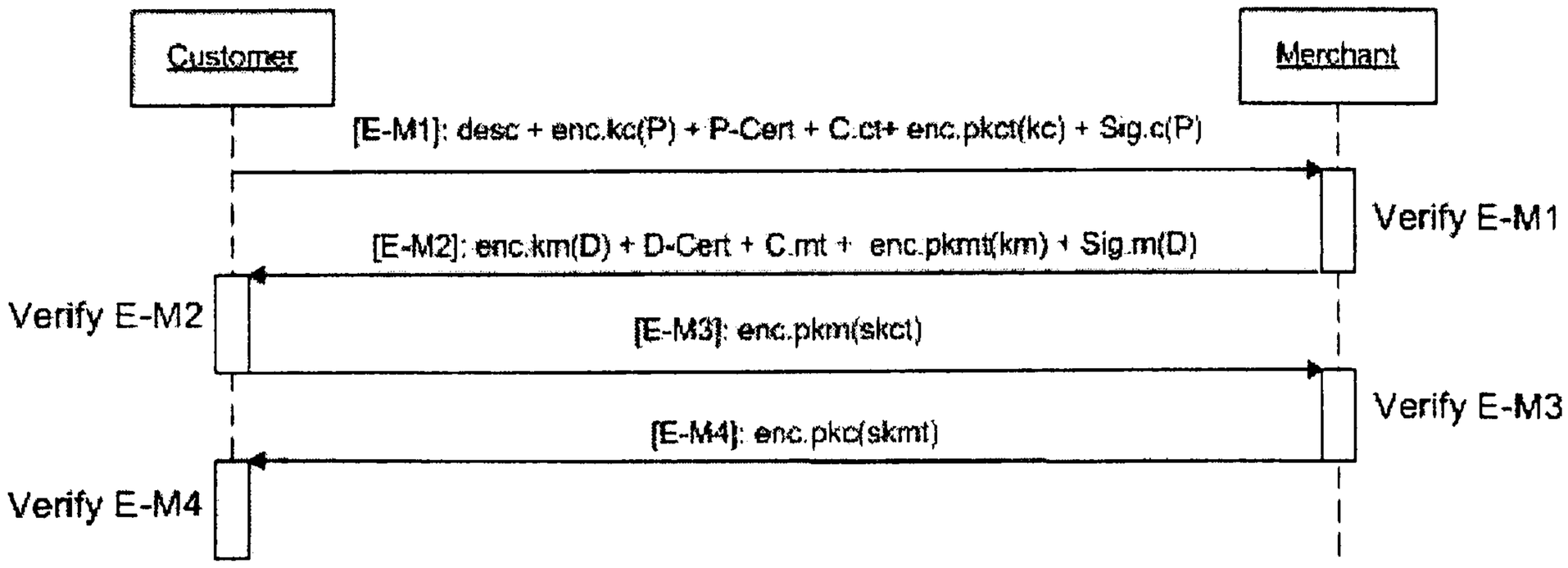

Figure 32: ECMH Exchange Phase

There are four messages to be exchanged between M and C in the exchange phase (Figure 32). These four messages are as follows:

[E-M1] **C → M: desc + enc.kc(P) + P-Cert + C.ct+ enc.pkct(kc) + Sig.c(P)**

C sends to M the first message E-M1 which contains the following:

- **desc**: specifies what C wants from M i.e. description of D that C wants (the description can be the digital product ID)
- **enc.kc(P)**: the payment that is encrypted with the key *kc*. *kc* is generated by C
- **P-Cert**: the payment certificate that is issued by the CB
- **C.ct**: the shared public key certificate that is issued by the TTP
- **enc.pkct(kc)**: the key *kc* (that is used to encrypt the payment) is encrypted using the shared public key *pkct* that is certified in *C.ct*
- **Sig.c(P)**: C's signature on the payment. This signature can serve as non-repudiation of origin which allows M to be sure that the payment is sent by C. C's signature on payment is the encryption of the hash value of payment using C's private key (skc)

[E-M2] **M → C: enc.km(D) + D-Cert + C.mt + enc.pkmt(km) + Sig.m(D)**

On receiving message E-M1 from C, M checks the correctness of *enc.kc(P)*, *enc.pkct(kc)*, *Sig.c(P)*, *P-Cert* and *C.ct*. The correctness of *P-Cert* can be checked by verifying the CB's signature on *P-Cert* and the correctness of *C.ct* can be checked by verifying the TTP's signature on *C.ct*.

To check that the encrypted payment is correct, M needs to check three things (1) the amount field in P-Cert against the price field in D-Cert that M has. This is to make sure that the payment meets the asked price; (2) the payment itself; and (3) the encrypted payment with *kc* i.e. *enc.kc(P)*.

To check the correctness of the payment, M needs to get the hash value of payment (calculated as *HP*) by decrypting *Sig.c(P)* using C's public key *pkc* (the public keys of all parties are publicly available) and then compare it with hash value of the payment (*hP*) that is included in *P-Cert*. That is, to check the following:

$$HP \stackrel{?}{=} hP$$

If they are the same then M can be sure that C signed the correct payment.

To check the correctness of the encrypted payment *enc.kc(P)*, M computes the hash value of *enc.kc(P)* (calculated as *HeP*) and then compare it with the hash value of encrypted payment with *kc* i.e. *heP* which is included in *P-Cert* (note that it is assumed that M will use the same function used by the CB to compute the hash value). That is, to check the following:

$$HeP \stackrel{?}{=} heP$$

If they are the same then M can be sure that C encrypted the payment using *kc* and not another key.

M also needs to check the correctness of *kc* which is used to encrypt the payment. To do so, M computes the hash value of *enc.pkct(kc)* (calculated as *HeKc*) and then compare it with *heKc* that is included in P-Cert, so M will check the flowing:

$$HeKc \stackrel{?}{=} heKc$$

If they are the same then M can be sure that the encrypted key is *kc* and not another key. The point here is to make sure that C is honest by sending the key used to encrypt the payment.

Therefore, if all comparisons are correct then, at this point, M will have the following fact. The encrypted payment is correct (i.e. it is the one certified in *P-Cert*) and it is indeed encrypted with *kc*. In addition, the encrypted key in *enc.pkct(kc)* is indeed *kc* and not another key. The shared public key *pkct* used to encrypt *kc* is certified by TTP. Therefore, once M got the private key (*skct*) of the shared public key then M will be able to get the payment (by first decrypting *enc.pkct(kc)* to get *kc* and then decrypting *enc.kc(P)* using *kc*).

Now, it is M's choice to complete the exchange or abort the protocol. If M wants to exchange D for the payment then M sends (in E-M2) the following to C:

- **enc.km(D)**: the digital product D that is encrypted with the key *km* (*km* is generated by M). This D must be the one described by C in E-M1 in *desc*

- **D-Cert:** the digital product certificate that is issued by the CA
- **C.mt:** the shared public key certificate that is issued by the TTP
- **enc.pkmt(km):** the key *km*, that is used to encrypt D, encrypted using the shared public key *pkmt* that is certified in *C.mt*
- **Sig.m(D):** M's signature on D. This signature can serve as non-repudiation of origin which allows C to be sure that D is sent by M. M's signature on D is the encryption of the hash value of D using M's private key skm

Note that if M decides to abort the transaction after receiving message E-M1 and before sending message E-M2 to C then neither M nor C lose anything.

[E-M3] **C → M: enc.pkm(skct)**

On receiving message E-M2 from M, C checks the correctness of *enc.km(D), enc.pkmt(km), Sig.m(D), D-Cert* and *C.mt*. The correctness of *D-Cert* can be checked by verifying the CA's signature on *D-Cert* and the correctness of *C.mt* can be checked by verifying the TTP's signature on *C.mt*.

To check the correctness of D, C needs to check two things which are the digital product *D* itself and the encrypted D with *km* i.e. *enc.km(D)*. Firstly, to check the correctness of D, C needs to get the hash value of D (calculated as *HD*) by decrypting *Sig.m(D)* contained in message E-M2 using M's public key *pkm* (the public keys of all parties are publicly available) and then compare it with hash value of D (*hD*) contained in *D-Cert*. That is, to check the following:

$$HD \stackrel{?}{=} hD$$

If they are the same then C can be sure that M signed the correct D.

Secondly, to check the correctness of the encrypted D *enc.km(D)*, C computes the hash value of *enc.pkmt(D)* (calculated as *HeD*) and then compares it with the hash value of encrypted D with *km* i.e. *heD* which is contained in *D-Cert* (note that it is

assumed that C will use the same function used by the CA to compute the hash value) i.e. to check the following:

$$HeD \stackrel{?}{=} heD$$

If they are the same then C can be sure that M encrypted D using *km* and not another key.

C also needs to check the correctness of *km* which is used to encrypt *D*. To do so, C computes the hash value of *enc.pkmt(km)* (calculated as *HeKm*) and then compares it with *heKm* that is included in D-Cert, so C will check the flowing:

$$HeKm \stackrel{?}{=} heKm$$

If they are compared then C can be sure that the encrypted key is *km* and not another key. The point here is to make sure that M is honest by sending the key used to encrypt D.

Therefore, if all comparisons are correct then, at this point, C will have the following fact. The encrypted D is correct (i.e. it is the one certified in *D-Cert*) and it is indeed encrypted with *km*. In addition, the encrypted key in *enc.pkmt(km)* is indeed *km* and not another key. The shared public key *pkmt* used to encrypt *km* is certified by TTP. Therefore, once C got the private key (*skmt*) of the shared public key then C will be able to get D (by first decrypting *enc.pkmt(km)* to get *km* and then decrypting *enc.km(D)* using *km*).

Note that C must be sure that the encrypted D matches their requirements as explained earlier, otherwise C will be at risk if they send a correct E-M3 to M because when C sends to M the decryption key then this means that they are satisfied with E-M2 and hence M will be able to decrypt the payment.

Now, it is C's choice to complete the exchange or abort the protocol. If C wants to exchange the payment for D then C sends to M the decryption key *skct* encrypted using M's public key *pkm* to allow M be able to decrypt the encrypted payment.

Note that if C decides to abort the transaction after receiving message E-M2 and before sending message E-M3 to M then neither C nor M lose anything. But once C sends a correct E-M3 to M then the transaction must be completed and the protocol will guarantee that the exchange of payment and D will be fair if the sent items are as they described i.e. the payment matches the price in D-Cert and also the digital product matches *desc* that appears in message E-M1.

[E-M4] **M → C: enc.pkc(skmt)**

On receiving message E-M3, M decrypts *enc.pkm(skct)* using M's private key *skm* to get the private key *skct.* Once M got *skct* then they decrypt *enc.pkct(kc)* to get *kc* that can be used to decrypt the encrypted payment received in E-M1.

If M managed to get the payment correctly then M sends to C in E-M4 the decryption key *skmt* that is encrypted using C's public key.

On receiving message E-M4, C decrypts *enc.pkc(skmt)* using C's private key *skc* to get the private key *skmt*. Once C got *skmt* then they decrypt *enc.pkmt(km)* to get *km* that can be used to decrypt the encrypted D received in E-M2.

If C managed to get D correctly then the protocol finishes and the fair exchange of payment and digital product is ensured. If, however, C has any dispute then they can contact the TTP for resolution (the dispute resolution will be discussed in the following section).

7.2.4 Dispute resolution

All disputes requests, if any, will come from C since M will not need to raise disputes as they get the decryption key of the encrypted payment and decrypt it before they send the decryption key of the digital product to C. Thus, if C has a dispute, the following messages are sent (see Figure 33):

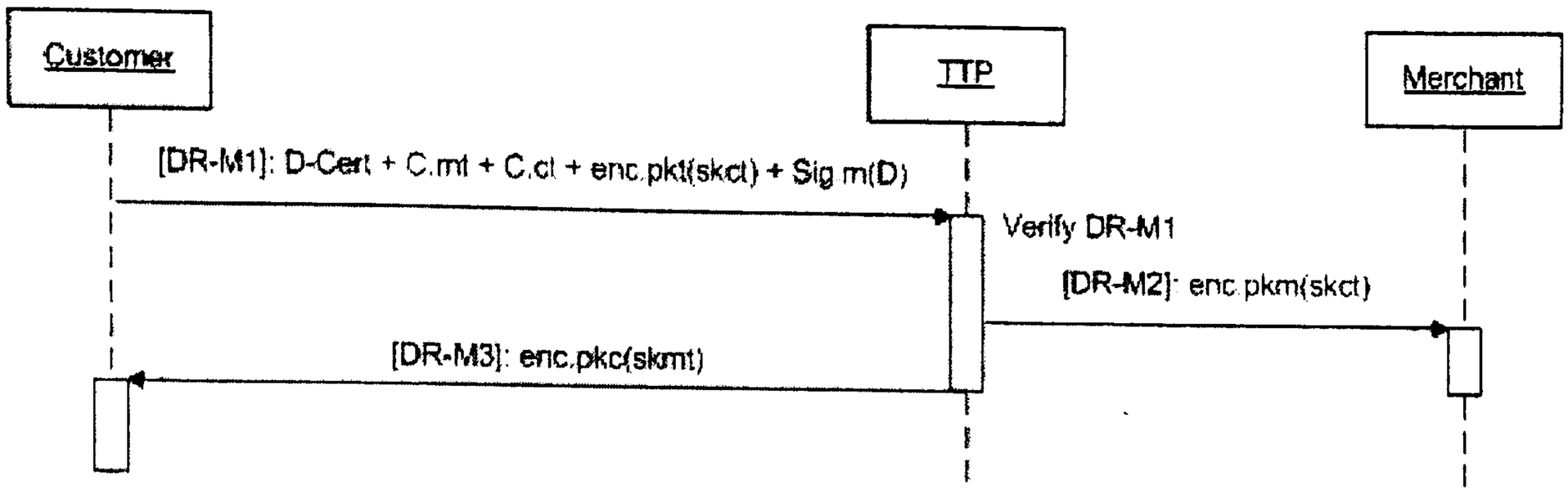

Figure 33: ECMH Dispute Resolution

[DR-M1] **C → TTP: D-Cert + C.mt + C.ct + enc.pkt(skct) + Sig.m(D)**

In case C has a dispute, they need to send to the TTP the following: *D-Cert*, *C.mt, C.ct, enc.pkt(skct)* and M's signature on D that has been received in message E-M2 of the exchange phase.

[DR-M2] **TTP → M: enc.pkm(skct)**

On receiving message *DR-M1* above, the TTP will check the correctness of *D-Cert*, *C.mt, C.ct* by checking their signatures. If they are correct then the TTP will decrypt the signature of M on D. That is, TTP decrypts *Sig.m(D)* to get the hash value of D included in the signature and then compares it with the hash value of D (*hD*) which is included in *D-Cert*. If the TTP managed to decrypt *Sig.m(D)* correctly and the two hashes are the same then the TTP is sure that M was satisfied with the payment that C sent to them in message E-M1 of the exchange phase. This is because no other party can sign D as it needs M's private key which is only held by M. If M was not satisfied then they would not send *Sig.m(D)* to C in message E-M2. In other words, M will send message E-M2 (which includes *Sig.m(D)*) only if they are sure that the payment sent by C is correct. If TTP found that the certificates and the signature of M is correct then TTP sends to M the decryption key *skct* (encrypted with M's public key) to be used to get *kc* that decrypts the encrypted payment. The reason for sending the decryption key *skct* to M (as M is not the one who raises the dispute) is because C may have not sent the decryption key to M in message E-M3 or has sent incorrect decryption key.

Otherwise, if the TTP found that either the certificates are incorrect or the signature of M is incorrect then the TTP sends an abort message to C and nothing will be sent to M.

[DR-M3] **TTP → C: enc.pkc(skmt)**

OR

TTP → C: aborts;

This is the same process for message *DR-M2* above, if the TTP found that *Sig.m(D)* is correct then the TTP sends to C the decryption key *skmt* (encrypted with C's public key) to be used to get *km* that decrypts the encrypted D. Otherwise if *Sig.m(D)* is incorrect then the TTP sends an abort message to C.

It is clear that if either C has sent incorrect decryption key *skct* to M or C has not sent the decryption key at all in message E-M3 then C will not get an advantage over M because the TTP will check *DR-M1* that C send to the TTP in order to check the signature of M. If the signature is correct then the TTP will send the decryption keys to both parties (C and M) to ensure fairness. Therefore, the fairness is ensured for both C and M. However, if the signature of M is incorrect then the TTP will reject C's request for a dispute.

As can be seen in the dispute resolution phase, the TTP does not need to have both C and M to be involved in order for the dispute to be resolved; rather only the disputant (C in this protocol) and the TTP will be involved. That is, the TTP does not need to contact M to verify whether or not they have received the correct decryption key; rather the TTP asks C to provide all evidences and finally will make the resolution. M will only be contacted by the TTP if the dispute has a resolution. Therefore, this will reduce the number of messages needed to resolve disputes.

7.3 ECMH Protocol Analysis

In this section, the ECMH protocol will be analysed. The analysis includes the ability of parties to detect the dishonesty of one another, disputes analysis, and presenting and discussing all possible scenarios.

7.3.1 Detection of Dishonesty

The same analysis that has been done for the ECH protocol and the EMH protocol will be done for the ECMH protocol.

The ways in which the customer may act dishonestly are by:

1. sending an incorrect E-M1; and
2. sending an incorrect E-M3

Incorrect E-M3 can be constructed by sending incorrect decryption key (*skct*). However, there are different ways in which incorrect E-M1 can be constructed. These are exactly the same possibilities mentioned for message E-M1 for the EMH protocol; see Table 5 in the EMH Protocol Chapter. Therefore, to avoid repetition these possibilities will not be discussed again.

On the other hand, the ways in which the merchant may act dishonestly are by:

1. sending an incorrect E-M2;
2. sending an incorrect E-M4; and
3. not sending E-M4

In case 2, there is only one possibility by which E-M4 can be incorrect. This possibility is by sending an incorrect decryption key (*skmt*). In case 3 there is also one possibility which is not sending E-M4 to C. However, there are different ways in which incorrect E-M2 can be constructed. These are exactly the same possibilities

mentioned for message E-M1 for the ECH protocol; see Table 2 in the ECH Protocol Chapter. Therefore, to avoid repetition these possibilities will not be discussed again.

Therefore, if C tried to act dishonestly then M will be able to detect it. Also, if M tried to act dishonestly then C will be able to detect it. As a result, the fairness is ensured for both C and M in the ECMH protocol.

7.3.2 Dispute Analysis

For the ECMH, the actual exchange between C and M is for the decryption key in E-M3 (from C) and the decryption key in E-M4 (from M). This is because C and M exchanged their encrypted items in messages E-M1 and E-M2. Therefore, after receiving the first message (E-M1) M will verify it and if satisfied they will send the second message (E-M2) to C. If C is satisfied with the second message then the exchange of the decryption keys between C and M will take place. The order of the exchange of the decryption keys in the ECMH protocol is as follows. M will receive a correct decryption key from C before C receives the correct decryption key from M.

	C	**M**	**Result**
	Receive decryption key (E-M4)	**Receive decryption key (E-M3)**	
1	√	√	No dispute
2	√	X	Not applicable
3	X	√	C disputes
4	X	X	No dispute / Not applicable in ECMH protocol / C's fault

Table 7: Disputes possibilities for ECMH

Table 7 shows all possible cases for disputes for the ECMH protocol. Each case is discussed below. In Table 7, (X) means either the party (C or M) has not

received the item (decryption key) at all or they received incorrect item; whereas (√) means the correct item is received.

1) In case 1, both C and M receive the correct item (i.e. C receives the correct decryption key and M receives the correct decryption key) from each other. Hence, there is no dispute.
2) In case 2, C received a correct decryption key, and M either received incorrect decryption key or has not received the decryption key at all. This case is not applicable in the ECMH protocol because C has to send a correct decryption key to M to be able to receive the correct decryption key from M.
3) In case 3, C has either received incorrect decryption key or not received the decryption key at all, and M received the correct decryption key. In this case C will dispute to the TTP.
4) In case 4, there are four possibilities which are:
    a) Both C and M have not received anything from each other. So, no dispute will be made as both of them have not revealed their items (the decryption keys). This represents the case where C received E-M2 and did not send E-M3 to M or the case where C sends E-M1 to M but M does not send E-M2 to C.
    b) Both C and M have received incorrect items from each other. That is, C received incorrect decryption key and M received incorrect decryption key. This case is not applicable in the ECMH protocol because C has to send a correct decryption key to be able to receive the correct decryption key from M. So, if M found that the decryption key is incorrect then M will not send to C the decryption key at all.
    c) C received an incorrect decryption key and M has not received the decryption key at all. This case is not applicable in the ECMH protocol because C has to send a correct decryption key to M to be able to receive the correct decryption key from M. So, if M has not received the decryption key then M will not send the decryption key at all.
    d) C has not received the decryption key at all and M received incorrect decryption key. This case is normal to occur because if C sent incorrect decryption key then M will not send their decryption key to C. Therefore, if this case occurs then for C to raise a dispute to the TTP, C

needs to send to the TTP the correct DR-M1 (see message DR-M1 of the dispute resolution phase of the ECMH protocol). If C sends the correct DR-M1 to the TTP then the TTP will make a resolution to both C and M. However, if the TTP found that DR-M1 is incorrect then C's dispute will be rejected

The design of the ECMH protocol reduces the possibilities for having disputes. Additionally, only C will raise disputes as M will not send their item unless the items from C are correct. As a result, the possibilities for disputes are reduced by preventing them occurring.

In addition to the previous cases, the following cases are studied:

- C disputes that they have received incorrect digital product. This is not possible scenario because *D-Cert* guarantees that the digital product is correct; and if C found that the digital product is incorrect or not the same as they wanted then they should have not sent to M the decryption key in E-M3. So, it is C's fault to send to M the decryption key if they have any doubt about the digital product. Once C sends to M the decryption key then this means that they are satisfied with the digital product. Therefore, this scenario will not happen because C knows the rules of the protocol which allow C to check the digital product before they send the decryption key to M; and as a result C will not put themselves at risk
- It is clear that M will not raise a dispute because M will receive from C the decryption key *skct* and get the payment before they send the decryption key *skmt* to C. However, the following scenarios are studied:
    - M claims that they have received incorrect payment from C. This will not occur because if M received incorrect payment then they will not send the encrypted digital product in message E-M2 i.e. if M found that the payment is incorrect then they would not send the encrypted digital product to C and hence no one will get advantage over the other party. However, once M sends message E-M2 to C then this means that they are satisfied with the encrypted payment

- M claims that they have not received the decryption key *skct*. This is not applicable in this protocol because if the decryption key is not received then no party is hurt and the fairness is not compromised. The reason for not receiving the decryption key *skct* may be because C is not satisfied with the encrypted digital product or C is no longer interested in the exchange
- M claims that they have received incorrect decryption key *skct* from C. This is not applicable in this protocol because if the decryption key is incorrect then no party is hurt and the fairness is not compromised as if the *skct* is incorrect then M will not send their decryption key (*skmt*) to C

7.3.3 Scenarios Analysis

There are different scenarios for executing the ECMH protocol by C and M. These scenarios include:

a) having both C and M are behaving honestly
b) C is behaving dishonestly and M is behaving honestly
c) M is behaving dishonestly and C is behaving honestly
d) having both C and M are behaving dishonestly

The possible scenarios for executing the ECMH protocol are as follows:

1. C and M are honest which result in normal execution. That is, C sends a correct E-M1, M sends a correct E-M2, C sends a correct E-M3, and M sends a correct E-M4 (Figure 34)

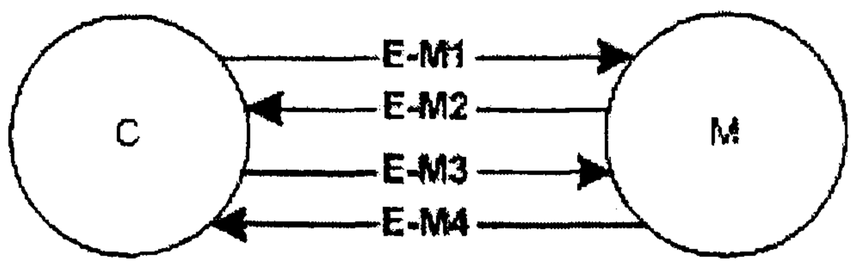

Figure 34: ECMH Protocol, Scenario 1

2. After receiving E-M1, M quits the protocol because either E-M1 is incorrect or M is no longer interested in the exchange (Figure 35)

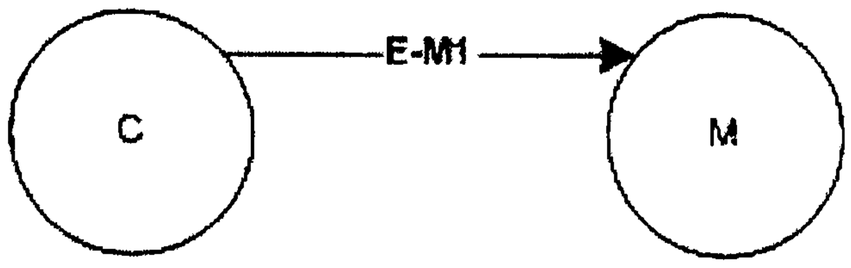

Figure 35: ECMH Protocol, Scenario 2

3. After receiving E-M1, M found that E-M1 is correct and hence they sent E-M2 to C. However, C quits the protocol after receiving E-M2 because either E-M2 is incorrect or C is no longer interested in the exchange (Figure 36)

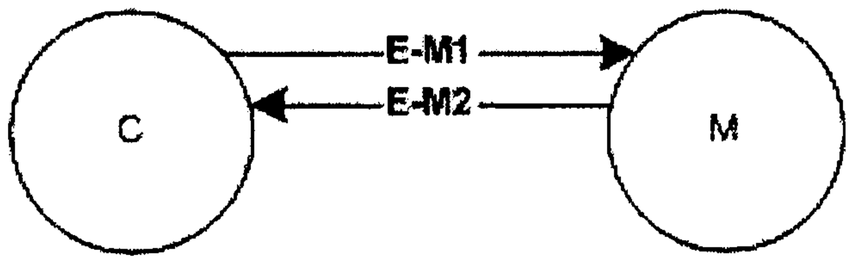

Figure 36: ECMH Protocol, Scenario 3

4. After receiving E-M2, C contacts the TTP before sending E-M3 to M. However, the TTP found that DR-M1 is incorrect. Therefore, the TTP sends abort message to C. In this scenario C tries to cheat but they gained nothing (Figure 37)

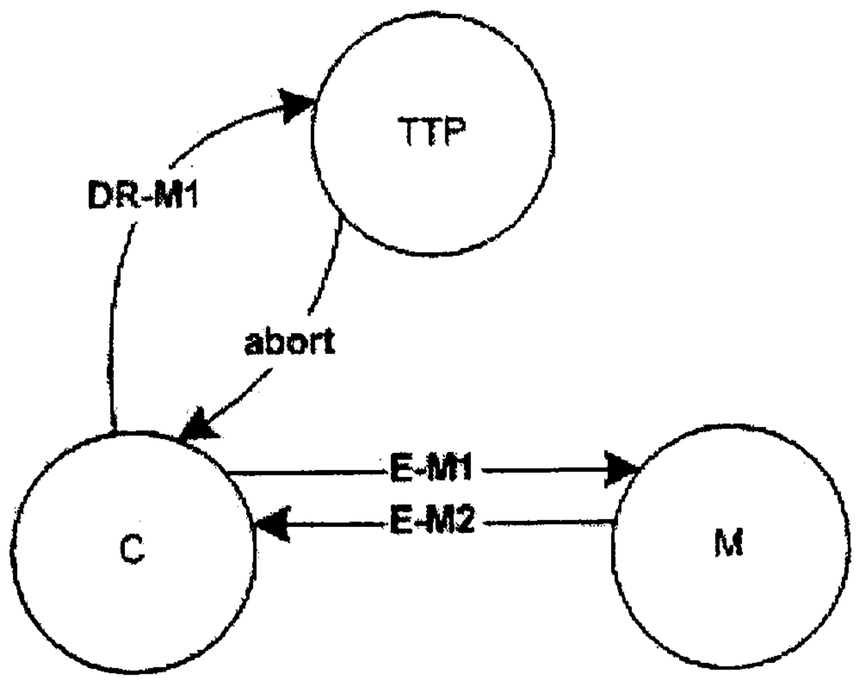

Figure 37: ECMH Protocol, Scenario 4

5. The same as scenario number 4 but in here, the TTP found that DR-M1 is correct. Therefore, the TTP will make it fair for both C and M by sending the

resolution to M in DR-M2 and also by sending the resolution to C in DR-M3. In this scenario C tries to cheat but the TTP makes it fair for both C and M (Figure 38)

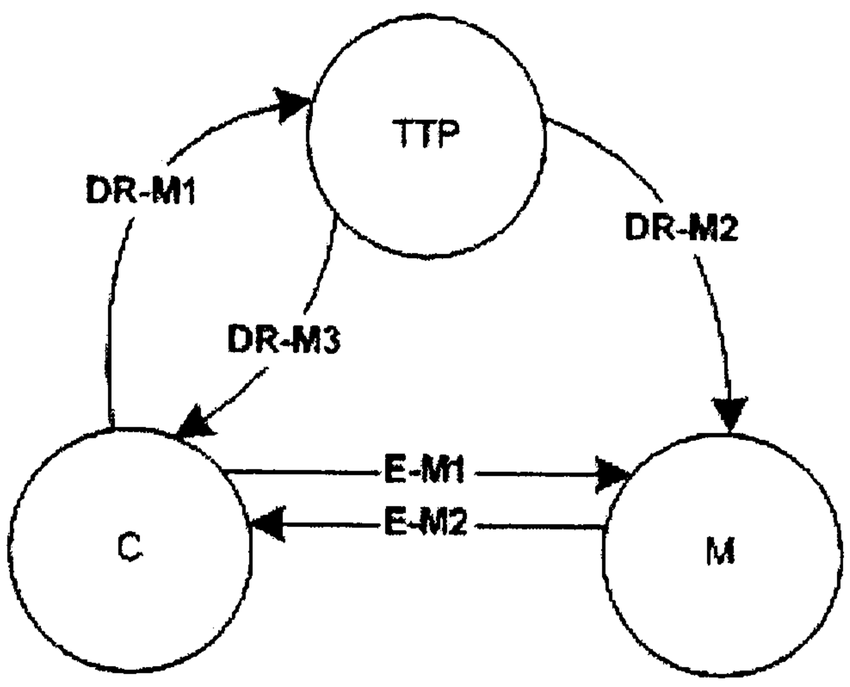

Figure 38: ECMH Protocol, Scenario 5

6. After M received E-M1 from C, M found that E-M1 is correct and hence M sends E-M2 to C. Then, C validated E-M2 and found it correct as well. C is interested in the exchange and hence C sent E-M3 to M. C waited to receive E-M4 from M but nothing is received. Therefore, C contacted the TTP for resolution. However, the TTP found that DR-M1 is incorrect and hence the TTP sends an abort message to C. Note, there are two possibilities why M did not send E-M4 to C. These possibilities are either because M found that E-M3 is incorrect or because M is dishonest. If it is the former (where E-M3 is incorrect) then it is C's fault to send incorrect message to M. While, if it is the later (where E-M3 is correct but M is dishonest) then C needs to send correct DR-M1 to the TTP to be able to receive a resolution (Figure 39)

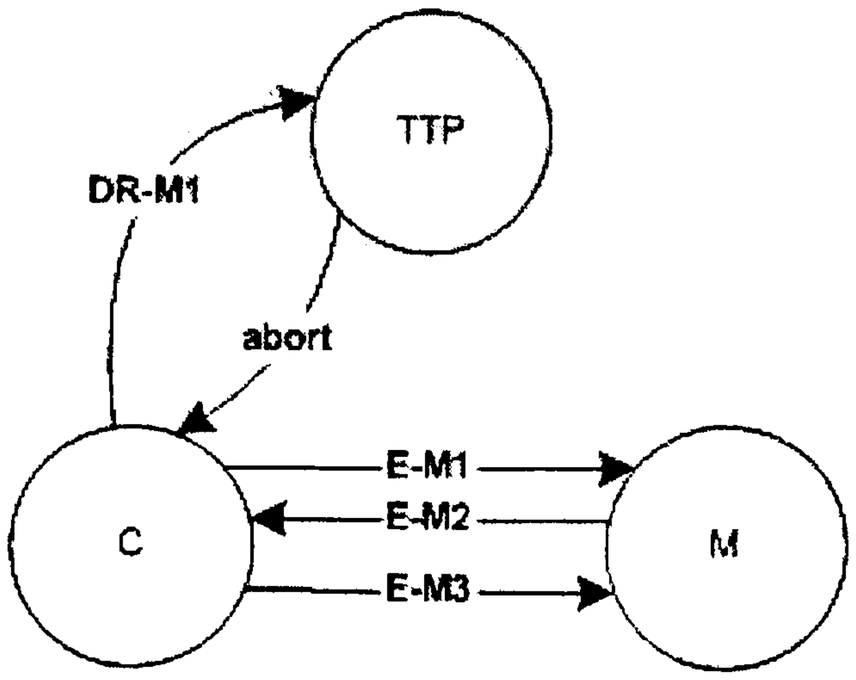

Figure 39: ECMH Protocol, Scenario 6

7. The same as scenario number 6 but in here, DR-M1 that C sent to the TTP is correct. Therefore, the TTP resolves it by sending DR-M2 to M and DR-M3 to C. Again, the reason for M not sending E-M4 to C is either because E-M3 is incorrect or because M is dishonest. When the TTP received the correct DR-M1 then the TTP will make it fair for both C and M. If it is the case where M is dishonest (by not sending E-M4) then M will receive the same decryption key twice (one in E-M3 and the other in DR-M2). If, however, M did not send E-M4 because E-M3 is incorrect then the TTP will ensure fairness for both C and M (Figure 40)

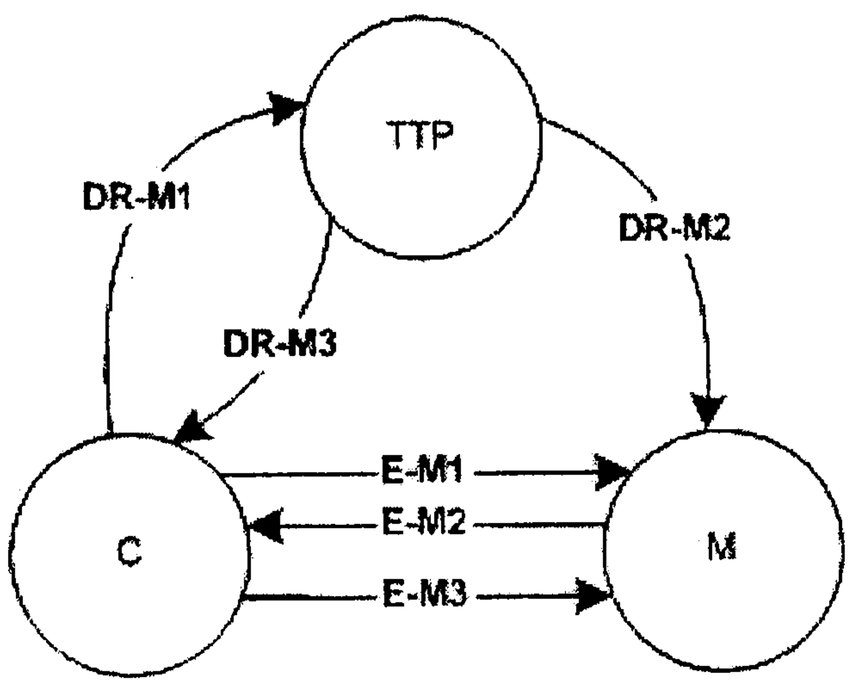

Figure 40: ECMH Protocol, Scenario 7

8. After M received E-M1 from C, M found that E-M1 is correct and hence M sent E-M2 to C. Then, C found that E-M2 is correct. Hence, C sent E-M3 to M. When M found that E-M3 is correct, they sent E-M4 to C. After receiving E-M4 C contacted the TTP. However, the TTP found that DR-M1 is incorrect. Therefore, the TTP sends an abort message to C. There are two possibilities why C contacted the TTP in this scenario. The first one is because E-M4 is incorrect. The second one is because E-M4 is correct but C wants to see what they can receive from the TTP and thus trying to get an advantage over M. In both possibilities, C has to send correct DR-M1 to receive a resolution (Figure 41)

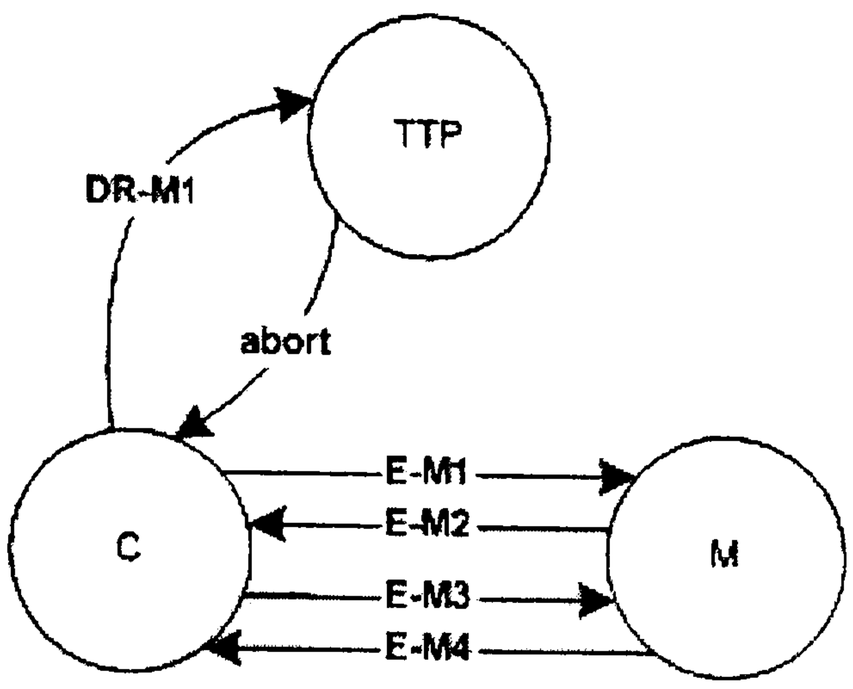

Figure 41: ECMH Protocol, Scenario 8

9. The same as scenario number 8 but in here, DR-M1 that C sends to the TTP is correct. Therefore, the TTP resolves it by sending DR-M2 to M and DR-M3 to C. Hence, the fairness is ensured for both C and M (Figure 42)

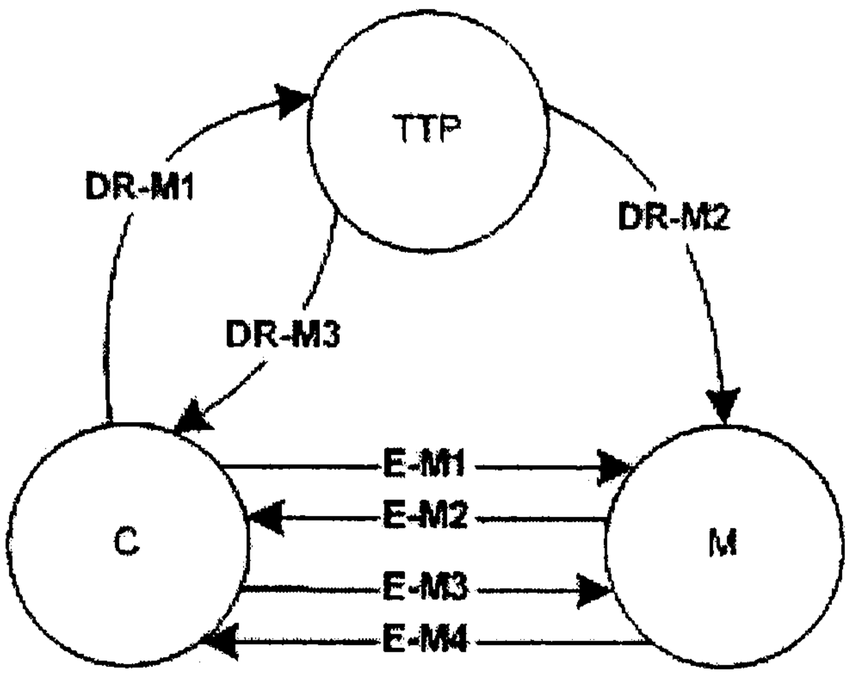

Figure 42: ECMH Protocol, Scenario 9

Note that there are other scenarios which include M sending dispute requests to the TTP but the TTP will find that DR-M1 is incorrect as M has not got the key *skct*. Therefore, if this kind of scenario occurs then the TTP will reject the dispute request.

Therefore, it is clear from the above scenarios that fairness is ensured for both C and M.

On the other hand, the following studies the scenarios where the messages (E-M1, E-M2, E-M3, E-M4, DR-M1, DR-M2, and DR-M3) of the ECMH protocol are

sent but have not been received because of the failure in the communication channels between the parties involved in the protocol:

1) If E-M1 is not received by M then the fairness will not be compromised because no one revealed their item
2) If E-M2 is not received by C then the fairness will not be compromised because no one revealed their item
3) If E-M3 is not received by M then M will consider that C is not interested in the exchange and hence M will not send E-M4. As a result, C will wait for E-M4 from M and as M has not received E-M3 from C then M will not send E-M4. Therefore, C will contact the TTP for resolution. However, there is a possibility of having failure in the communication channel. So, this possibility will be studied later (see case 5)
4) If E-M4 is not received by C then C will consider it as M behaving dishonestly. As a result, C will contact TTP for resolution. However, there is a possibility of having the communication channel failed. So, this possibility will be studied in the next step (see case 5)
5) If DR-M1 is not received by the TTP then C needs to re-contact the TTP asking for a resolution again
6) If DR-M2 is not received by M then there are two possibilities:
    a) If DR-M2 is not received by M (and also E-M3 is not received by M) and at the same time DR-M3 is not received by C then the fairness is not compromised
    b) If DR-M2 is not received by M but DR-M3 is received by C then the fairness is compromised if and only if M has not received a correct E-M3
7) If DR-M3 is not received by C then there are two possibilities:
    a) If DR-M3 is not received by C and at the same time E-M3 and DR-M2 are not received by M then the fairness is not compromised
    b) If DR-M3 is not received by C but either E-M3 or DR-M2 are received by M then the fairness is compromised if and only if C has not received a correct E-M4 i.e. when C received a correct E-M4 they contacted the TTP

As can be seen in these cases, the communication channels between C and TTP, and M and TTP should be resilient for the fairness to be ensured. To ensure

fairness even in the case of communication channels failure the fault tolerance techniques need to be applied to the ECMH protocol. However, this is out of the scope of this thesis. Hence, it is left as a future work.

7.4 Summary

The ECMH protocol is presented and analysed in this chapter. The number of messages needed for the exchange of items in this protocol is four messages.

It has been shown in this chapter that each party (customer and merchant) will be able to detect the dishonesty of the other party. If a party detects that the other party is dishonest then they will not send their item to them. All dispute possibilities for the ECMH protocol have been presented. The possible scenarios of executing the ECMH protocol have been discussed.

Chapter Eight

8. Protocols Comparisons

8.1. Introduction

In this chapter the ECH, EMH, and ECMH protocols will first be compared against each other and then compared with other relevant protocols in the literature.

8.2. Comparisons of the Three Protocols

In this section, the ECH, EMH, and ECMH protocols will be compared against each other.

8.2.1. ECH Protocol vs. EMH Protocol

The number of messages in the exchange phase of the ECH protocol and the EMH protocol are three messages. Therefore, the two protocols have the same number of messages in the exchange phase.

The way in which products (digital or physical) and payments are exchanged online is that a customer sends the payment to a merchant and then the merchant checks the correctness of the payment. If the payment is correct then the merchant sends the product to the customer. This way is applied in the EMH protocol. However, the way it is applied in the ECH protocol is that a merchant sends an encrypted digital product to the customer and then the customer checks the correctness of the encrypted

digital product. If the customer is satisfied with the digital product then they send the payment to the merchant.

Each way has advantages and disadvantages. The advantage of the way it is applied in the EMH protocol is that it follows the conventional exchange order where the customer sends money first and then the merchant sends the product. Therefore, customers and merchants in the EMH protocol will not feel that there are changes in the way they conduct business. Whereas they will notice a difference in the ECH protocol as the merchant will start the exchange by sending the digital product. Additionally, in the EMH protocol, the merchant receives the first message (E-M1) which contains the encrypted payment and then the merchant may decide not to complete the exchange i.e. not to send the second message (E-M2) to the customer. However, in the ECH protocol the customer receives the first message (E-M1) which contains the encrypted digital product and then the customer may decide not to complete the exchange i.e. not to send the second message (E-M2) to the merchant. Therefore, if the ECH protocol and the EMH protocol are compared in this respect (i.e. the party who receives E-M1 decides not to complete the exchange because either they are not interested in the exchange or E-M1 is incorrect), then the EMH protocol would be better in that the size of payment is smaller than the size of digital product.

The EMH protocol puts more responsibility on merchant to find a digital product that matches customer's requirements. This is the way in which merchant is enforced to be honest and hence sends the correct digital product to customer in order to receive the decryption key to decrypt the encrypted payment. The ECH protocol, on the other hand, puts more responsibility on customer to send a correct payment in order to receive the decryption key to decrypt the encrypted digital product. This is the way in which customer is enforced to be honest in the ECH protocol.

In the case of disputes, in the EMH protocol the party to raise disputes, if any, is the merchant whereas in the ECH protocol the party to raise disputes, if any, is the customer. The number of messages to be executed in case of dispute is three messages which are the same for both the ECH protocol and the EMH protocol.

The total size of messages (i.e. DR-M1, DR-M2, and DR-M3) in the dispute resolution phase for the EMH protocol is bigger than the size of messages (i.e. DR-M1, DR-M2, and DR-M3) in the dispute resolution phase for the ECH protocol. The reason for this is that in the EMH protocol the encrypted digital product is included in both DR-M1 and DR-M2; while, in the ECH protocol the encrypted payment is the one included in both DR-M1 and DR-M2. Therefore, it is clear that the size of digital product is bigger than the size of payment (especially if the size of digital product is very big).

The ECH protocol enforces the customer to be honest whereas the EMH protocol enforces the merchant to be honest. Therefore, the idea of enforcing a party to be honest has been applied in both protocols. Both protocols ensure strong fairness.

Regarding the number of modular exponentiations (which are the most expensive operations [NeZhChGo05]), both the ECH and the EMH protocols have the same number of modular exponentiations in the exchange phase with 11 modular exponentiations; and also in the dispute resolution phase with 7 modular exponentiations.

8.2.2. ECH and EMH Protocols vs. ECMH Protocol

The number of messages to be exchanged between merchant and customer in both the ECH protocol and the EMH protocol is only three messages whereas there are four messages in the ECMH protocol.

In the case of disputes, three messages are sent to resolve disputes in all the three protocols. However, the items (payment and digital product) are not included in DR-M1, DR-M2, and DR-M3 of the ECMH protocol; whereas these items are included in the ECH protocol (the payment is included) and the EMH protocol (the digital product is included). Therefore, the total size of messages in the dispute resolution phase in the ECMH protocol is smaller than the ones in the ECH and the EMH protocols.

The number of modular exponentiations in the exchange phase of the ECMH protocol is 14 whereas the number of modular exponentiations in the dispute resolution phase of the ECMH protocol is 9. Therefore, both the ECH and the EMH protocols have lower numbers of modular exponentiations in both the exchange and dispute resolution phases.

The ECH protocol and the EMH protocol are more efficient than the ECMH protocol because of many reasons. Firstly, the ECH protocol and the EMH protocol have fewer messages in the exchange phase. Secondly, the ECMH protocol has more encryptions and certificates. That is, four certificates are included in the ECMH protocol in the exchange phase. These certificates are P-Cert, C.ct, D-Cert and C.mt whereas only two certificates are included in the ECH protocol and three certificates are included in the EMH protocol. Finally, the number of modular exponentiations in the exchange phase is greater in the ECMH protocol than the one in the ECH and the EMH protocols. Also, the number of modular exponentiations in the dispute resolution phase is greater in the ECMH protocol than the one in the ECH and the EMH protocols.

The Table 8 summarizes the comparisons between the three protocols presented in this thesis.

	ECH Protocol	**EMH Protocol**	**ECMH Protocol**
# messages in exchange phase	3	3	4
# messages in dispute phase	3	3	3
Starts the exchange	M	C	C
Enforced to be Honest	C	M	N/A
Party who raises disputes	C	M	C
# of modular exponentiations (exchange phase)	11	11	14
# of modular exponentiations (dispute resolution phase)	7	7	9

Table 8: Comparisons between ECH, EMH, and ECMH protocols

8.2.3. Comparing the Timing of the Three Protocols

In this section, the timing of executing the ECH, EMH, and ECMH protocols will be compared. The timing numbers are in milliseconds and the numbers are the mean of running the protocols 10 times.

The specification of the computer used to perform and measure these timings are as follows. The processor is Intel Pentium 4, 2.4 GHz, and 496 MB of RAM.

For each protocol, three scenarios have been carried out. The first one is to have a digital product (D) that of size 28 KB, the second one is to have a digital product (D) that of size 2.2 MB, and the third one is to have a digital product (D) that of size 10 MB.

Tables 9, 10, and 11 represent the time comparisons tables for the ECH, the EMH, and the ECMH protocols, respectively. SD in Tables 9, 10, and 11 refers to Standard Deviation. Note, some of the standard deviations in Tables 9, 10 and 11 are larger than the mean because the distribution of the values is skewed and the values given are in milliseconds. Timings on a PC are not precise hence the distribution of values will probably be skewed.

	Scenario 1: digital product size= 28 KB	**Scenario 2: digital product size= 2.2 MB**	**Scenario 3: digital product size= 10 MB**
E-M1 Construction	40 (SD=9.4)	312.3 (SD=17.8)	1310.9 (SD=261.7)
E-M1 Verification	10 (SD=4.7)	50.3 (SD=0.4)	199.3 (SD=10.1)
E-M2 Construction	28.1 (SD=4.2)	29 (SD=3.1)	32 (SD=6.3)
E-M3 Construction	17 (SD=8.2)	26 (SD=33.7)	54.1 (SD=62.7)
Payment decryption	2 (SD=4.2)	3 (SD=4.8)	4 (SD=6.9)
Digital product decryption	164.1 (SD=17.7)	557.8(SD=18.2)	1394 (SD=59.4)
Total	**261.2**	**978.4**	**2994.3**

Table 9: Timing of the ECH Protocol

	Scenario 1: digital product size= 28 KB	**Scenario 2: digital product size= 2.2 MB**	**Scenario 3: digital product size= 10 MB**
E-M1 Construction	32 (SD=4.4)	42 (SD=4.4)	38 (SD=4.5)
E-M1 Verification	18 (SD=8.3)	18 (SD=4.4)	18.2 (SD=4.6)
E-M2 Construction	38.2 (SD=4.6)	364.8 (SD=20.9)	1359.8 (SD=136.4)
E-M3 Construction	11 (SD=2.2)	12 (SD=4.4)	24 (SD=20.7)
Payment decryption	140.2 (SD=0.4)	138 (SD=4.4)	136 (SD=5.4)
Digital product decryption	8 (SD=4.4)	248.6 (SD=38.1)	1009.6 (SD=16.3)
Total	**247.4**	**823.4**	**2585.6**

Table 10: Timing of the EMH Protocol

In the tables (9, 10 and 11) the time gets bigger if a message includes the digital product. That is, the time needed to construct the message E-M1 of the ECH protocol (see table 9) when the digital product size equals 2.2 MB is greater than the time needed when the digital product size equals 28 KB.

The time is roughly the same when a message does not include the digital product. Examples from the tables (9, 10, and 11) are as follows. First, the timings of E-M2 construction, E-M3 construction, and payment decryption in table 9. Second, the timings of E-M1 construction, E-M1 verification, E-M3 construction, payment decryption in table 10. Finally, the timings of E-M1 construction, E-M1 verification, E-M3 construction, E-M4 construction, and payment decryption in table 11.

	Scenario 1: digital product size= 28 KB	**Scenario 2: digital product size= 2.2 MB**	**Scenario 3: digital product size= 10 MB**
E-M1 Construction	24 (SD=5.1)	21 (SD=3.1)	22 (SD=4.2)
E-M1 Verification	12 (SD=4.2)	14 (SD=5.1)	13 (SD=4.8)
E-M2 Construction	28 (SD=6.3)	346.6 (SD=20.7)	1369.9 (SD=269.1)
E-M2 Verification	13.1 (SD=4.9)	51.1 (SD=3.1)	202.3 (SD=8.2)
E-M3 Construction	31.1 (SD=26.9)	20 (SD=22.1)	36 (SD=41.9)
E-M4 Construction	13 (SD=4.8)	15 (SD=9.7)	21.3 (SD=12.2)
Payment decryption	178.2 (SD=25.6)	185.3 (SD=56.2)	157.1 (SD=4.9)
Digital product decryption	157.3 (SD=9.4)	591 (SD=51.2)	1570.4 (SD=172.6)
Total	**456.7**	**1244**	**3392**

Table 11: Timing of the ECMH Protocol

It is worth mentioning that the time needed to decrypt the same item is different in each protocol. That is, the time needed to decrypt the payment in the ECH protocol (see table 9) is less than the time needed to decrypt the payment in the EMH protocol (see table 10). The reason for this is that the merchant in the ECH protocol has the decryption key that decrypts the payment (i.e. the merchant has *kc*). Therefore, on receiving the encrypted payment in message E-M2, the merchant decrypt it using the key *kc*. On the other hand, in the EMH protocol the merchant receives the encrypted *kc* (that is encrypted with *pkct*) along with the encrypted payment (that is encrypted with *kc*) in the message E-M1. Therefore, in order to decrypt the payment the merchant needs to first decrypt *kc* with *skct* (that is received in E-M3) then they

use *kc* to decrypt the payment. So, two decryptions are needed to get the payment in the EMH protocol whereas only one is needed in the ECH protocol; and this is the reason that the time needed for decrypting the payment in the ECH protocol is less than the one needed in the EMH protocol.

The same reason applies for the time needed to decrypt the digital product in the ECH protocol and the EMH protocol. That is, two decryptions are needed to get the digital product in the ECH protocol whereas only one is needed to get it in the EMH protocol. Hence, the time needed to decrypt the digital product in the EMH protocol is less than the time needed in the ECH protocol.

For the ECMH protocol, there is a need for two decryptions to get the digital product and also two decryptions to get the payment. The reason for this is that the merchant and the customer encrypt their items with *km* and *kc*, respectively. Then they encrypt *km* and *kc* with the shared keys *pkmt* and *pkct*, respectively. Therefore, the receiver first needs to decrypt the key (*km* or *kc*) then they decrypt the item.

From the above discussion, it is clear that in both the ECH protocol and the EMH protocol there is one item that requires one decryption (payment in the ECH and digital product in the EMH) and the other item requires two decryptions (digital product in the ECH protocol and payment in the EMH protocol). However, both items (payment and digital product) require two decryptions in the ECMH protocol. Hence, the total time required for the ECMH protocol is greater than the time required in the ECH and the EMH protocols. This reflects how the idea of enforcing a party to be honest helped in proposing more efficient protocols.

8.2.4. Cost functions for the three protocols

In this section a cost function will be devised for each of the ECH, EMH, and ECMH protocols. The cost functions to be devised concern with the time needed to form/construct the messages of the exchange phases of the protocols.

For the ECH protocol:

There are three messages in the exchange phase of the protocol. Therefore, a cost function will be devised for each message then a general cost function will be devised for the all messages of the protocol.

E-M1:

The message E-M1 includes six items, namely, *enc.km(D), D-Cert, C.mt, enc.pkmt(km), enc.pkc(kc), Sig.m(D)*. The items *D-Cert and C.mt* are received from CA and TTP, respectively, in the pre-exchange phase. Therefore, M will not need to construct them. As a result, M will need to construct the other four items (namely, *enc.km(D), enc.pkmt(km), enc.pkc(kc), Sig.m(D)*). So, the cost function for the E-M1 is the sum of the time needed to construct these four items:

$$f_{(\text{E-M1})} = \text{enc.km(D)} + \text{enc.pkmt(km)} + \text{enc.pkc(kc)} + \text{Sig.m(D)}$$

The time needed for constructing *enc.km(D)* will change according to the size of the digital product (D) whereas the time needed to construct the other three items will remain roughly the same because the size of the keys (i.e. km, pkmt, pkc, kc and skm that is used to sign the digital product in Sig.m(D)) will remain the same in any message even if the keys change. Therefore, the cost function for E-M1 is:

$$f_{(\text{E-M1})} = \text{enc.km(D)} + c1 + c2 + c3$$

where c1, c2 and c3 are constants that represent the time needed to construct enc.pkmt(km), enc.pkc(kc), and Sig.m(D), respectively.

E-M2:

The message E-M2 includes two items, namely, *enc.kc(P)* and *Sig.c(P)*. Therefore, the cost function for the E-M2 is the sum of the time needed to construct these two items:

$$f_{(E\text{-}M2)} = \text{enc.kc(P)} + \text{Sig.c(P)}$$

The payment normally includes specific information (such as the names of the payer and payee, and the amount) which has roughly the same size even if the information included in the payment is different i.e. if there is a payment of £100 from Mike to Sally and another payment of £20 from Louise to John then the size of the two payments is roughly the same. Therefore, the time needed to construct the two items included in E-M2 will remain roughly the same in any message. Therefore, the cost function for E-M2 is:

$$f_{(E\text{-}M2)} = c4 + c5$$

where c4 and c5 are constants that represent the time needed to construct enc.kc(P) and Sig.c(P), respectively.

E-M3:

The message E-M3 includes one item which is *enc.pkc(skmt)*. Therefore, the cost function for E-M3 is:

$$f_{(E\text{-}M3)} = \text{enc.pkc(skmt)}$$

The time needed to construct E-M3 will remain roughly the same even if the keys (i.e. pkc and skmt) change because the size of these keys will be the same. Therefore, the cost function for E-M3 is:

$$f_{(E\text{-}M3)} = c6$$

where c6 is a constant that represents the time needed to construct enc.pkc(skmt).

The cost function for the exchange phase of the ECH protocol:

The cost function for the exchange phase of the ECH protocol is the sum of the cost functions for E-M1, E-M2 and E-M3. That is:

$$f(\text{ECH}) = f(\text{E-M1}) + f(\text{E-M2}) + f(\text{E-M3})$$

$$f(\text{ECH}) = (\text{enc.km(D)} + \text{c1} + \text{c2} + \text{c3}) + (\text{c4} + \text{c5}) + (\text{c6})$$

$$f(\text{ECH}) = \text{enc.km(D)} + \text{c1} + \text{c2} + \text{c3} + \text{c4} + \text{c5} + \text{c6}$$

$$f(\text{ECH}) = \text{enc.km(D)} + \text{C} \qquad \text{(where C is the sum of all constants)}$$

The time needed to construct the messages in the exchange phase of the ECH will increase significantly if the size of the digital product gets bigger. Otherwise, the time will remain roughly the same if the size of the digital product remains the same. The time needed to construct the messages of the exchange phase of the ECH protocol in three scenarios was given in Table 9.

For the EMH protocol:

The cost function for each message will be devised then the cost function for the EMH protocol will be the sum of these cost functions.

E-M1:

$$f(\text{E-M1}) = \text{desc} + \text{enc.kc(P)} + \text{enc.pkct(kc)} + \text{enc.pkm(km)} + \text{Sig.c(P)}$$

Note that P-Cert and C.ct are not included because they were constructed in the pre-exchange phase by CB and TTP, respectively. The time needed to construct the items included in $f(\text{E-M1})$ will remain roughly the same as the size of these items will not change. Therefore, this cost function will be as follows:

$$f_{(E\text{-}M1)} = c1 + c2 + c3 + c4 + c5$$

where c1, c2, c3, c4 and c5 are constants that represent the time needed to construct desc, enc.kc(P), enc.pkct(kc), enc.pkm(km), Sig.c(P), respectively.

E-M2:

$$f_{(E\text{-}M2)} = enc.km(D) + Sig.m(D)$$

The time needed to construct enc.km(D) will change according to the size of the digital product whereas the time needed to construct Sig.m(D) will remain roughly the same. Therefore, this cost function can be rewritten as follows:

$$f_{(E\text{-}M2)} = enc.km(D) + c6$$

where c6 is a constant that represents the time needed to construct Sig.m(D).

E-M3:

$$f_{(E\text{-}M3)} = enc.pkm(skct)$$

The time needed to construct E-M3 will remain roughly the same even if the keys (i.e. pkm and skct) change because the size of these keys will be the same. Therefore, the cost function for E-M3 is:

$$f_{(E\text{-}M3)} = c7$$

where c7 is a constant that represents the time needed to construct enc.pkm(skct).

The cost function for the exchange phase of the EMH protocol:

The cost function for the exchange phase of the EMH protocol is the sum of the cost functions for E-M1, E-M2 and E-M3. That is:

$$f_{(EMH)} = f_{(E\text{-}M1)} + f_{(E\text{-}M2)} + f_{(E\text{-}M3)}$$
$$f_{(EMH)} = (c1 + c2 + c3 + c4 + c5) + (enc.km(D) + c6) + (c7)$$
$$f_{(EMH)} = c1 + c2 + c3 + c4 + c5 + enc.km(D) + c6 + c7$$
$$f_{(EMH)} = enc.km(D) + C \qquad \text{(where C is the sum of all constants)}$$

Therefore, the cost function for constructing the messages in the exchange phase of the EMH protocol is very similar of the one for the ECH protocol. In both of them the time gets bigger when the digital product gets bigger. The time needed to construct the messages of the exchange phase of the EMH protocol in three scenarios was given in Table 10.

For the ECMH protocol:

The cost function for each message will be devised then the cost function for the ECMH protocol will be the sum of these cost functions.

E-M1:

$$f_{(E\text{-}M1)} = \text{desc} + \text{enc.kc(P)} + \text{enc.pkct(kc)} + \text{Sig.c(P)}$$

This cost function will be rewritten as follows:

$$f_{(E\text{-}M1)} = c1 + c2 + c3 + c4$$

where c1, c2, c3, and c4 are constants that represent the time needed to construct desc, enc.kc(P), enc.pkct(kc), Sig.c(P), respectively. The reasons for having these constants are the same as in E-M1 of the EMH protocol.

E-M2:

$$f_{(E\text{-}M2)} = \text{enc.km(D)} + \text{enc.pkmt(km)} + \text{Sig.m(D)}$$

This cost function will be rewritten as follows:

$$f_{(E\text{-}M2)} = \text{enc.km(D)} + c5 + c6$$

where c5 and c6 are constants that represent the time needed to construct enc.pkmt(km) and Sig.m(D) respectively.

E-M3:

$$f_{(E\text{-}M3)} = \text{enc.pkm(skct)}$$

This cost function will be rewritten as follows:

$$f_{(E\text{-}M3)} = c7$$

where c7 is a constant that represents the time needed to construct enc.pkm(skct).

E-M4:

$$f_{(E\text{-}M4)} = \text{enc.pkc(skmt)}$$

This cost function will be rewritten as follows:

$$f_{(E\text{-}M4)} = c8$$

where c8 is a constant that represents the time needed to construct enc.pkc(skmt).

Cost function for the exchange phase of the ECMH protocol:

The cost function for the exchange phase of the ECMH protocol is the sum of the cost functions for E-M1, E-M2, E-M3 and E-M4. That is:

$$f_{(ECMH)} = f_{(E\text{-}M1)} + f_{(E\text{-}M2)} + f_{(E\text{-}M3)} + f_{(E\text{-}M4)}$$
$$f_{(ECMH)} = (c1 + c2 + c3 + c4) + (\text{enc.km(D)} + c5 + c6) + (c7) + (c8)$$
$$f_{(ECMH)} = c1 + c2 + c3 + c4 + \text{enc.km(D)} + c5 + c6 + c7 + c8$$
$$f_{(ECMH)} = \text{enc.km(D)} + C \quad \text{(where C is the sum of all constants)}$$

The time needed to construct the messages of the exchange phase of the ECMH protocol in three scenarios was given in Table 11.

8.3. Comparisons with other Protocols

In this section, the ECH, EMH, and ECMH protocols will be compared with other protocols in the literature.

To have a fair comparison, the three protocols will only be compared to those protocols have the same characteristics. In other words, the ECH, EMH, and ECMH protocols will be compared to fair exchange protocols that are based on RSA [RiShAd78] and for exchanging digital products and payments (or for exchanging two digital products). Therefore, the ECH, EMH, and ECMH protocols will be compared against Ray et al protocol [RaRaNa05] (denoted as Ray protocol), Zhang et al protocol [ZhShMeAs06] (denoted as Zha protocol), Devane et al protocol [DeChPh07] (denoted as Devane protocol), and Ray et al protocol [RaRaNa00] (denoted as R-R-N protocol).

The comparisons will be made on different criteria. (1) number of messages in both the exchange and dispute resolution phases, (2) whether or not the TTP needs to hold a copy of an item to be exchanged, (3) whether or not all parties (M and C) will be involved to allow the TTP to resolve any disputes, and (4) number of modular exponentiations in both the exchange and dispute resolution phases.

The Ray protocol paper [RaRaNa05] did not give detail for the dispute resolution phase so the number of messages in the phase and the number of modular exponentiations had to be calculated manually. In addition, the numbers of modular exponentiations for Zha's protocol [ZhShMeAs06], R-R-N protocol [RaRaNa00], and Devane protocol [DeChPh07] have also been calculated manually.

As can be seen in Table 12, the ECH protocol and the EMH protocol have the lowest number of messages (among other protocols) needed to be exchanged between customer and merchant in the exchange phase. The ECMH protocol, Zha protocol, and Ray protocol has four messages; whereas R-R-N protocol and Devane protocol require six message and seven messages, respectively.

With regard to dispute resolution, all protocols apart from Ray protocol have the same numbers of messages needed in the dispute resolution phase.

Ray's protocol and R-R-N protocol let the TTP hold merchant's item before the exchange between customer and merchant takes place. Therefore, this requires more storage and security assurance in the TTP side.

Ray's protocol requires both parties (customer and merchant) to be contacted by the TTP in case one party raises a dispute; whereas in the other protocols only the disputant and the TTP will be involved. Involving both parties in dispute resolution would require more messages to be sent and hence more load on the communication channels.

With regard to the number of modular exponentiations, the ECH, EMH, and ECMH protocols have the lowest number of modular exponentiations needed to generate and verify messages in the exchange phase. On the other hand, the number of modular exponentiations for the ECH, the EMH protocols and other protocols in the dispute resolution phase is roughly the same. The ECMH protocol has the highest number of modular exponentiations in the dispute resolution phase.

It is clear from the comparison presented in Table 12 how the idea of enforcing honesty in fair exchange protocol reduces both the number of messages and the number of modular exponentiations. As a result, it helped in having more efficient fair exchange protocols.

	R-R-N [RaRaNa00]	Devane [DeChPh07]	Ray [RaRaNa05]	Zha [ZhShMeAs06]	ECH protocol	EMH Protocol	ECMH protocol
# messages in the exchange phase	6	7	4	4	3	3	4
# messages in dispute resolution phase	Not specified	Not specified	3 to 5	3	3	3	3
TTP type	Inline	Online	Offline	Offline	Offline	Offline	Offline
TTP hold item	Yes	No	Yes	No	No	No	. No
Both parties are involved in dispute resolution	Not specified	Not specified	Yes	No	No	No	No
# of modular exponentiations in the exchange phase	20	28	27	20	11	11	14
# of modular exponentiations in the dispute resolution phase	Not specified	Not specified	5 to 6	6	7	7	9

Table 12: Protocols Comparisons

8.4. Summary

This chapter presented the comparisons between the ECH, EMH, and ECMH protocols. The three protocols were also compared with other relevant protocols from the literature.

Chapter Nine

9. Model Checking

9.1. Introduction

This chapter introduces the notion of formal verification of systems and in particular model checking. Model checking will be used to model the ECH, EMH, and ECMH protocols to formally verify the fairness property.

9.2. Formal analysis (verification)

Formal analysis of systems is very important for detecting unexpected flaws in the system design. Many techniques can be used to do the formal verifications, including manual proofs, theorem proving, and model checking [AnHaLoSu06b, WaHiBaWh00]. The problem with manual proofs is that it is time consuming, slow and error prone [AnHaLoSu06a, WaHiBaWh00]. Theorem proving sometimes requires human involvement, when a failure is detected by the theorem prover then it will not necessarily provide a detailed description of the source of that failure [AnHaLoSu06a, WaHiBaWh00]. Model checking provides fully automated method for validating systems.

9.3. Model checking

Model checking is a formal verification technique which is fully automated. It is used to verify whether or not a property of a finite state system holds [HeTyWiWo96].

Model checking has many advantages over the other formal verification techniques such as manual proofs and theorem proving [AnHaLoSu05a, WaHiBaWh00, and RayRay00b]. Firstly, it is fully automated and hence is fast, not time consuming, and not error-prone. Secondly, in the case where a property does not hold then a model checker can provide a counterexample that specifies the reasons.

The tasks of a model checker can be summarised as follows [ClGrPe99]. First, the model (system) needs to be specified. Second, property to be verified needs to be specified (the properties represent the system requirements). Then, the model checker will automatically verify whether or not the property holds by checking all possible behaviours of the system. If the property does not hold then the model checker will provide a counterexample that specifies the reasons why the property does not hold. In addition to verifying properties of systems, some model checkers provide a facility for randomly simulating a system run.

Many model checkers are available. Some of which are FDR (Failures Divergences Refinement) [FDR99], SMV (Symbolic Model Verifier) [SMV08], Mocha [AlHeMaQaRaTa98], VeriSoft [VeriSoft08], and SPIN [Spin08].

Many e-commerce protocols have been verified using these model checkers. Ray and Ray [RayRay00b] used the FDR model checker to verify the fairness property. Wang et al [WaHiBaWh00] used two model checkers for their formal verifications, the VeriSoft [VeriSoft08] and the SPIN [Spin08]. Anderson et al [AnHaLoSu06a, AnHaLoSu05a, AnHaLoSu06b, and AnHaLoSu05b] verified the protocol described in [RaRaNa00] using the FDR model checker. Heintze et al [HeTyWiWo96] used the FDR model checker to verify the protocol presented in [CoTySi95]. Fanjul et al [FaTuCo98] used the SPIN model checker to verify the

protocol presented in [CoTySi95]. Nenadic [Nenadic05] used the SPIN to verify the protocols presented in [NeZhBa04a, NeZhBa04].

9.3.1. Spin Model Checker

The SPIN model checker was developed by Holzmann at Bell Labs in the 1980s [Spin08]. It has been awarded in 2001 the ACM's prestigious System Software Award [Spin08]. The SPIN is a free and open source model checker.

The fairness property has been verified in all the protocols presented in this thesis using the SPIN model checker. The reasons for choosing the SPIN is that, in addition to that it is model checker, it is a simulation tool as well. So, it helped simulating the scenarios of the proposed protocols. The second reason is that it has been successfully used to verify fair exchange protocols [Nenadic05 and FaTuCo98]. The third reason is that the language accepted by SPIN to specify the models (protocols) is Promela which is very similar to the language C. Hence, it is simple and easy to learn. Fourthly, the SPIN is free and open source tool.

To verify a system's property (fairness for example) using SPIN, the model (system) needs to be specified using the verification language Promela (Process Meta Language) [Holzmann04] and the properties to be verified needs to be specified using LTL (Linear Temporal Logic) [Holzmann04]. After specifying the model and the properties to be verified, these are fed into the SPIN's GUI (Graphical User Interface) which is called XSpin. The XSpin will identify if there are any syntax errors. If there are no errors then SPIN will verify whether or not the specified property is verified against the specified model. If it is verified then the verification result that confirms the correctness of the verification will be shown on the GUI. If, however, the specified property is not verified then a counterexample will be provided to identify the source of the error in the model.

9.3.1.1. Promela

The Promela (Process Meta Language) is the verification language accepted by the SPIN. There are three basic types that any Promela model is constructed from, processes, data objects, and message channels [Holzmann04].

Each party (entity) of a system is considered and modelled as a process. A process specifies all activities of the party. A process is defined using the keyword *proctype* and then followed by the process's name and then a set of parameters. This can be illustrated in the following example:

```
proctype Customer( parameters )
{
    statements
}
```

After specifying the process, it can be instantiated using the keyword *run*.

For the data objects, there are two types of variables: local variables, which are defined within the scope of a process, and global variables, which are defined outside the scope of processes. Promela defines basic types such as bit, byte, bool, int.

For the message channels, the exchange of data between processes is modelled using the message channels. They are declared using the keyword *chan*. For example,

```
chan ch = [0] of {bit};
```

defines a synchronous message channel and each message consists of one field which is of bit type.

The following statements send a message with the value of *c* which is 0 to the channel *ch*:

```
bit c =0;
ch!c;
```

The following statement receives a message from channel *ch* and stores the value in variable *m*:

```
ch?m;
```

A statement in Promela is either executable or blocked. If the statement is executable then it will be executed whereas if it is blocked then it will not be executed. Print statements and assignment statements are always executable. An expression statement is executable if it equals to true (not equals to zero).

9.3.1.2. LTL

The LTL (Linear Temporal Logic) [Holzmann04] is accepted by SPIN for specifying the system's properties to be verified. The operators of the LTL describe the order of events on a single computation path [ClGrPe99]. That is, the operators describe events on a sequence of states of a system.

In addition to the used logic operators such as "and" ($\wedge$), "or" ($\vee$), "not" ($\neg$), and "implies" ($\Rightarrow$); LTL provides "always" ($\Box$), "eventually" ($\Diamond$), and "next time" ($\bigcirc$).

9.4. Modelling the Protocols

The fairness properties for the ECH, EMH, and ECMH protocols have been verified using the SPIN model checker. The modelling of the three protocols is similar to each other and the modelling of the fairness property of the three protocols is exactly the same. Therefore, to avoid repetition only the modelling of the ECH and the

fairness property will be presented here. The full specifications of all the three protocols are given in Appendix A.

9.4.1. Modelling the ECH Protocol

In order to verify the fairness property in the ECH protocol using the SPIN model checker, the ECH protocol needs to be modelled (specified) using the Promela language and the fairness property needs to be specified using LTL.

Each party involved in the ECH protocol will be represented by a process in the Promela. All the behaviours of the party will be specified in the process. The parties involved in the ECH protocol are Customer, Merchant, and TTP.

9.4.1.1. Modelling Customer process

The Customer process in the Promela model represents the Customer entity in the ECH fair exchange protocol. The process will start by waiting for the message E-M1 from the Merchant process. Once the message is received then the process will list all possible behaviours of the Customer after receiving message E-M1 (these possible behaviours are discussed in section 5.3.3 of chapter 5). That is, E-M1 is correct and the Customer wants to complete the exchange, E-M1 is correct but the Customer does not want to complete the exchange, E-M1 is incorrect, or E-M1 is correct but the Customer contacts the TTP for resolution before they send E-M2 to the Merchant. So, all possible behaviours of the Customer are specified in the Customer process and then the model checker will verify all possible behaviours of the Customer with all possible behaviours of the Merchant. In the same way all possible behaviours of the Customer after sending E-M2, receiving E-M3, sending DR-M1, and receiving DR-M3 are specified.

The following presents a fragment that shows all possible behaviours of the Customer after receiving message E-M1 (the full specification of the Customer process is given in Appendix A).

```
proctype Customer(chan chCM, chCTTP)
{
    …
    …

     do
      :: (sendEM2 == FALSE) && (sendDRM1== FALSE)
          ->

        chCM?EM1;
        quitCustomer = FALSE;

       /* All possibilities after receiving EM1 */

          if
       :: TRUE ->
      /* EM1 is correct, C wants to exchange */
            chCM!EM2;
            sendEM2 = TRUE;

       :: TRUE ->
      /* EM1 is correct, C contacts TTP */
            chCTTP!DRM1;
            sendDRM1= TRUE;

       :: TRUE ->
      /* EM1 is correct, C quits early*/
            quitCustomer = TRUE;
            break;

       :: TRUE ->
            /* EM1 is incorrect, C quits */
            quitCustomer = TRUE;
            break;

          fi

          …
          …

od;
```

```
}
```

9.4.1.2. Modelling Merchant process

The Merchant process in Promela represents the Merchant entity in the ECH fair exchange protocol. The process will start by sending the message E-M1 to the Customer process. Once the message is sent, then the process will wait for the message E-M2 to be received from the Customer process. On receiving E-M2, the Merchant process will specify all possible scenarios and behaviours of the Merchant. The following shows a fragment of the Merchant process which shows sending message E-M1 to the Customer process (the full specification of the Merchant process is given in Appendix A).

```
proctype Merchant(chan chCM, chMTTP)
{
        …
        …

    do
        :: (sendEM1 == FALSE) ->
                chCM!EM1;
                 sendEM1 = TRUE;
                quitMerchant = FALSE;
                …
                …

    od;

}
```

9.4.1.3. Modelling TTP process

The TTP process in Promela represents the TTP entity in the ECH fair exchange protocol. The TTP will only be contacted by the Customer; therefore, in the model of the TTP it is listening for incoming messages from the Customer process. On receiving the message DR-M1 from the Customer process, the TTP process will send the resolution messages to both the Merchant process and the Customer process (the full specification of the TTP process can be found in the Appendix A).

9.4.1.4. Modelling Fairness property

Fairness is defined as either both the Customer and the Merchant receive each other's items or no one gets anything. That is, by the end of executing the ECH protocol, the Customer will get a correct digital product and the Merchant will get a correct payment or neither the Customer nor the Merchant will get anything. This fairness property is specified in LTL formula as follows:

$$\Box((\text{quit\_Merchant} \wedge \text{quit\_Customer}) \Rightarrow \Diamond((\text{receive\_CorrectPayment} \wedge \text{receive\_CorrectProduct}) \vee (\neg\text{receive\_CorrectPayment} \wedge \neg\text{receive\_CorrectProduct})))$$

This LTL formula specifies that it is always the case that when the Merchant and the Customer terminates the protocol execution then either the correct payment will be received by the Merchant and the correct digital product will be received by the Customer, or the correct payment will not be received by the Merchant and the correct digital product will not be received by the Customer.

9.4.1.5. The Verification results

XSpin has been used to write the protocol's specifications in Promela. Figure 43 shows the XSpin and the ECH protocol specification in Promela.

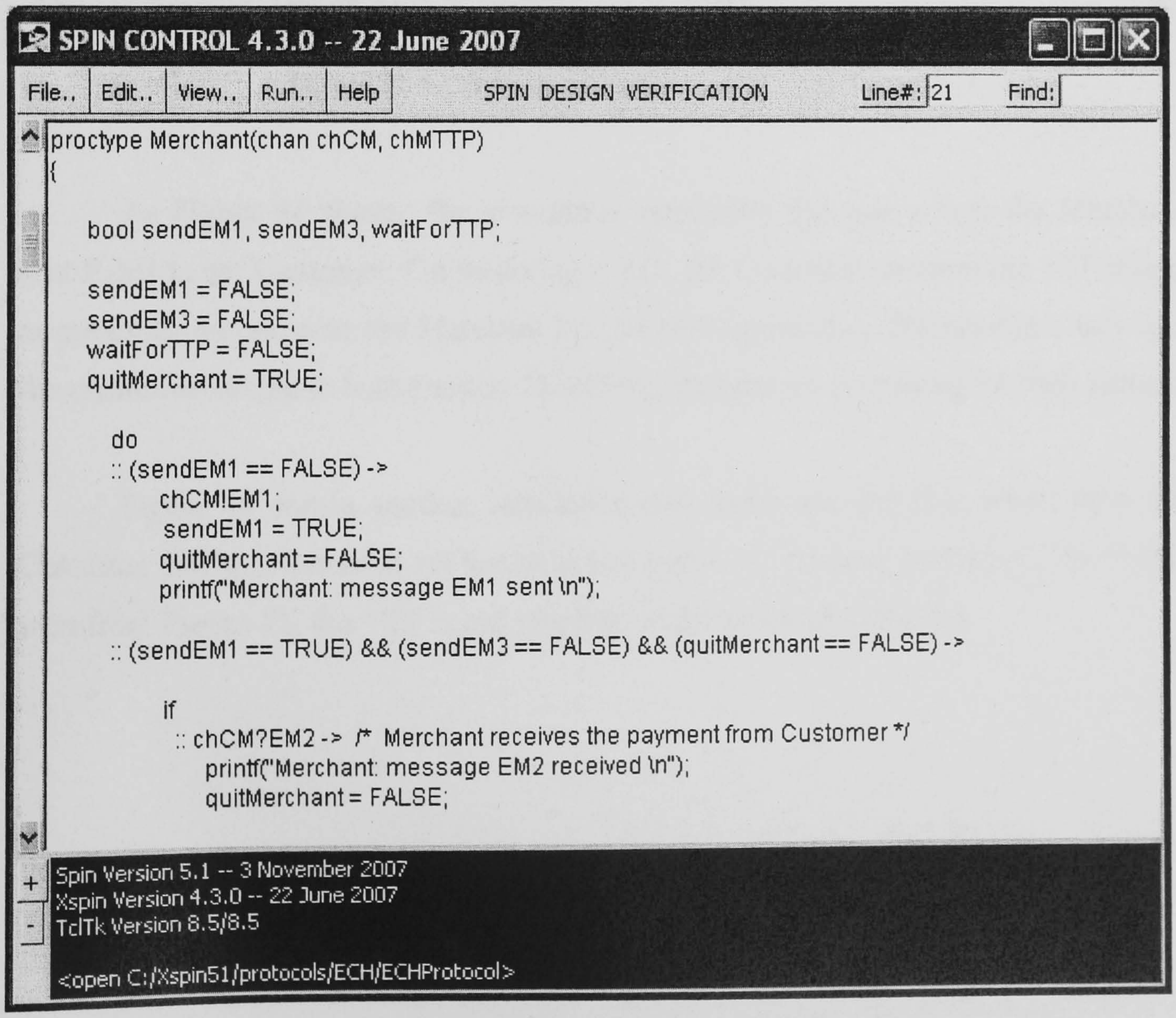

Figure 43: XSpin

XSpin provides a facility for simulating the protocol specified in Promela. Hence, a number of simulations for the ECH protocol are performed to check the behaviour of the parties. Figure 44 shows one random simulation of the ECH protocol.

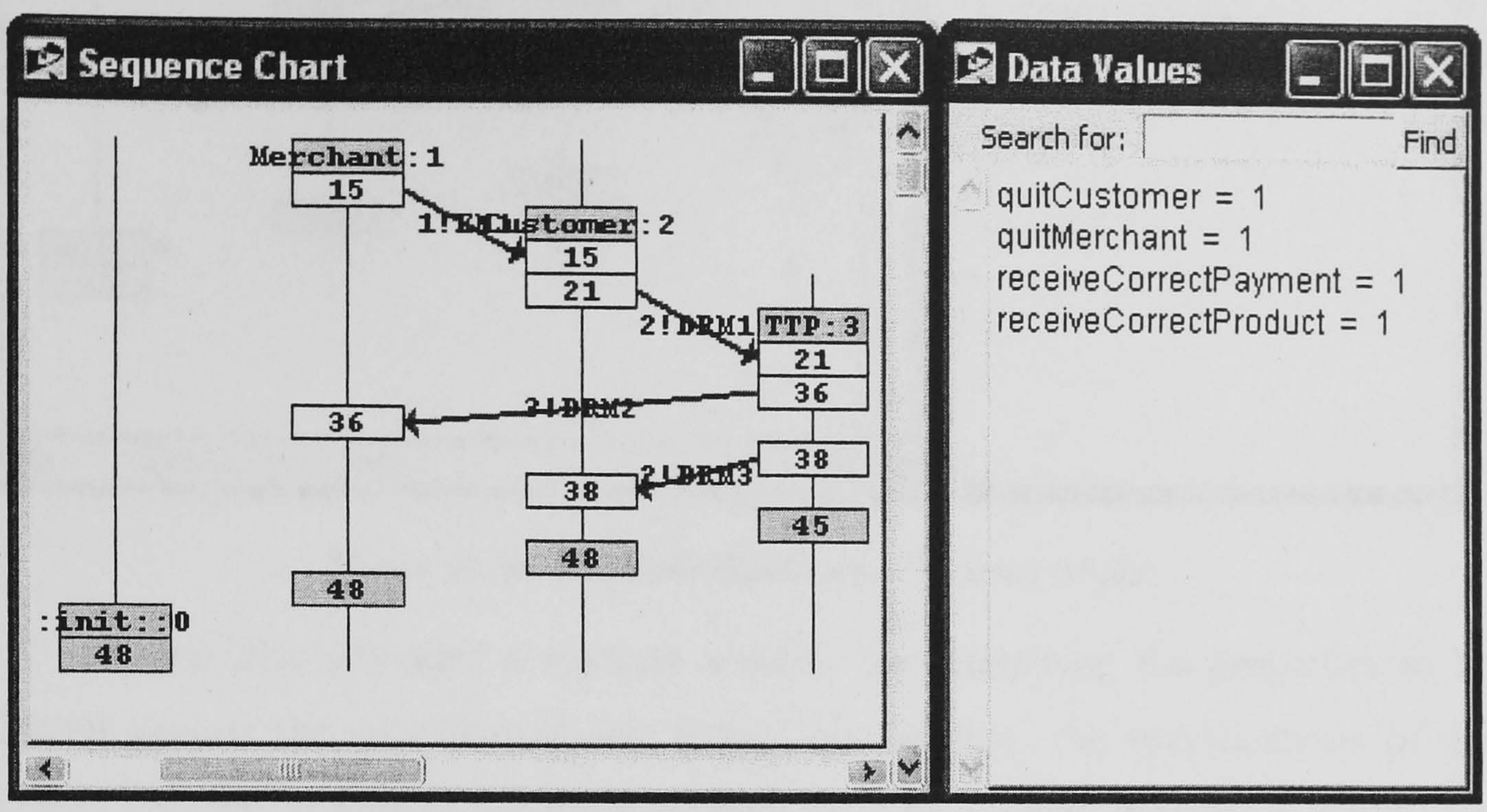

Figure 44: ECH Protocol Simulation (1) using XSpin

As Figure 44 shows, the simulation represents the case where the Merchant sent E-M1 to the Customer. On receiving E-M1, the Customer contacts the TTP trying to get an advantage over the Merchant but the ECH protocol is designed in a way that the resolution is sent to both parties. Therefore, the fairness is ensured for both parties.

Figure 45 shows another simulation that represents the case where both the Customer and the Merchant act honestly and hence the fairness in ensured. As can be seen from Figure 45, the TTP is not involved in the protocol execution.

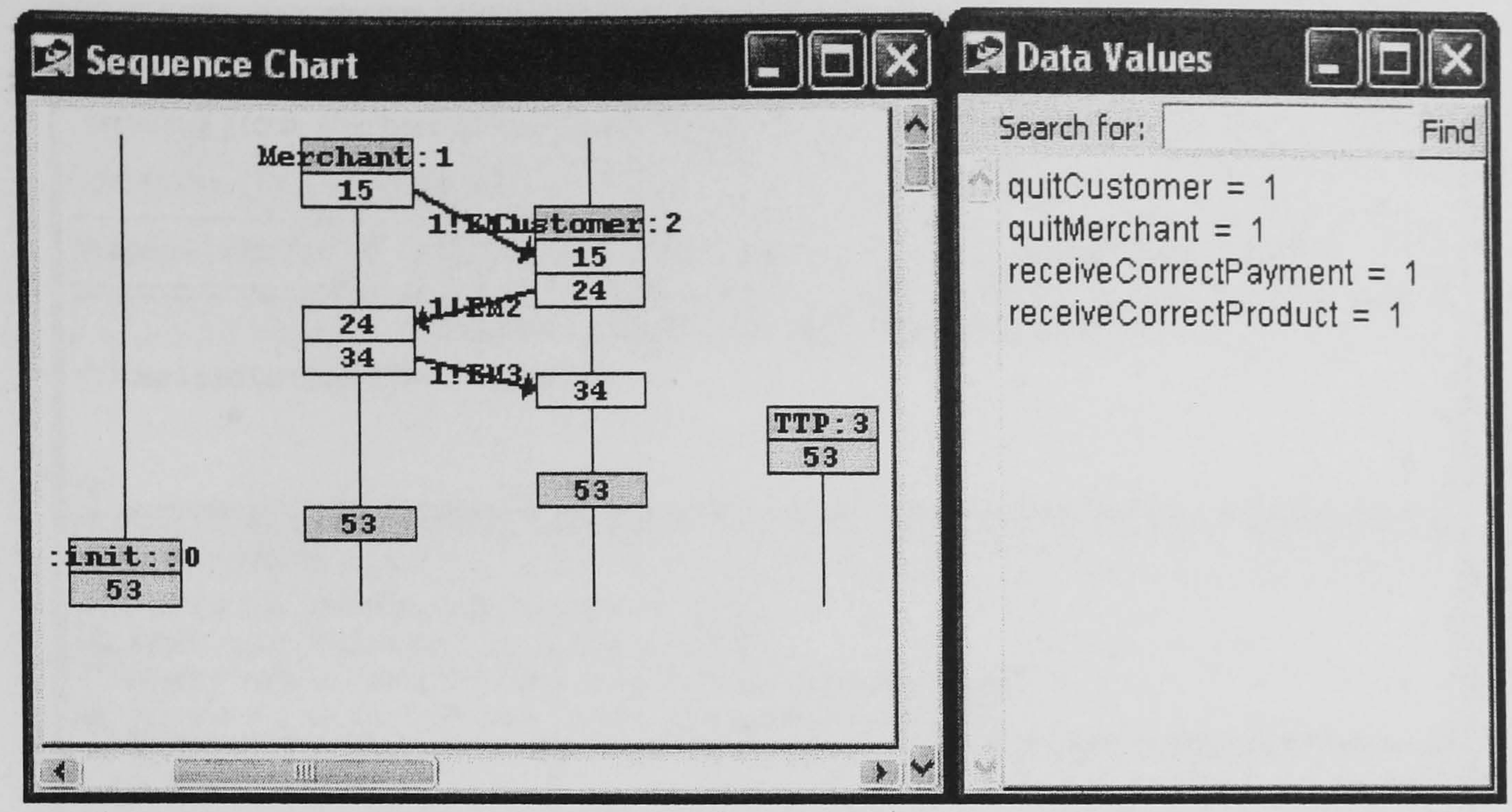

Figure 45: ECH Protocol Simulation (2) using XSpin

XSpin also provides a separate window for specifying the properties to be verified against the specified model. Using this window, the specifications of the fairness property is written i.e. the LTL formula. The window provides a facility for verifying whether or not the specified property is valid against the model specified in Promela. After writing the specification of the fairness property (this is through the top of Figure 46) and verifying it against the ECH protocol specification, SPIN shows that the fairness property is valid for the specified protocol. The verification result is shown in the bottom of Figure 46. Therefore, SPIN checked all possible behaviours of all parties and found that the fairness property holds.

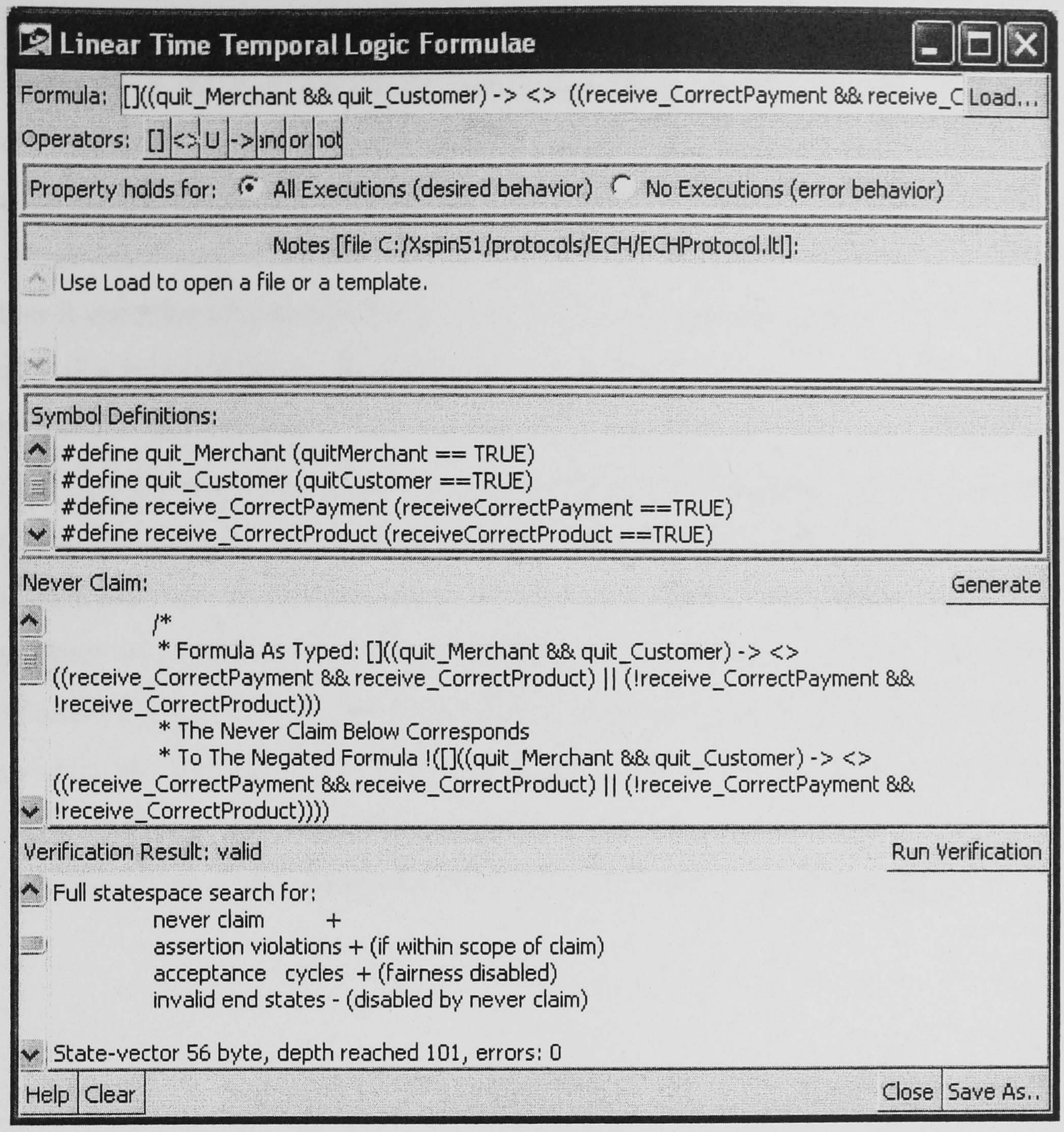

Figure 46: LTL Property Manager

9.5. Summary

This chapter presented the formal verification techniques used to verify systems. It has presented the modelling of the ECH, EMH, and ECMH protocols. The specification of the fairness property has also been presented. The SPIN model checker showed that the specified fairness property is valid against the specified models.

It is worth mentioning that the model (i.e. specification) of a protocol may be different from the protocol itself. This is due to either errors in modelling or the specification of the protocol cannot fully express the protocol. Reasons for the later can be as follows. Firstly, the model of the protocol, normally, specifies the protocol in an abstract way. Therefore, the model will not specify all items of the protocol rather it specifies all possible behaviours that will be checked by the model checker to verify if a property holds. Secondly, the formal methods may have some limitations which result in their lack to fully express the protocol [Nenadic05]. As a result, SPIN model checker helped in formally verifying the fairness property against the specified models. Additionally, it helped in simulating all scenarios of the protocols. However, implementing the protocols (which will be presented in the next chapter) is very important and different from the modelling of the protocols. This is because the implementation of the protocols will help dealing with real data (digital products and payment) and parties (C, M, TTP, CA, CB) in order to test whether or not all parties able to construct and send the protocols' messages, verify the correctness of the received messages, and requesting resolutions for disputes where applicable.

Chapter Ten

10. Protocols Implementation

10.1. Introduction

A prototype proof of concept implementation has been developed using the Java programming language. Java RMI (Remote Method Invocation) has been used for the remote interaction between the parties involved in the fair exchange protocols (e.g. Customer, Merchant, TTP, CA and CB).

This chapter presents the design and implementation of the prototype that implements the ECH, EMH and ECMH protocols.

10.2. High Level Design

The ECH, EMH and ECMH protocols are independently implemented. The design is different for each protocol because the parties are different in the protocols. The design for each protocol will be presented in the following sections.

10.2.1. High Level Design for ECH protocol

The system architecture for the ECH protocol consists of the following parties (see Figure 47):

- TTP Server
- CA Server

- Merchant Server
- Customer

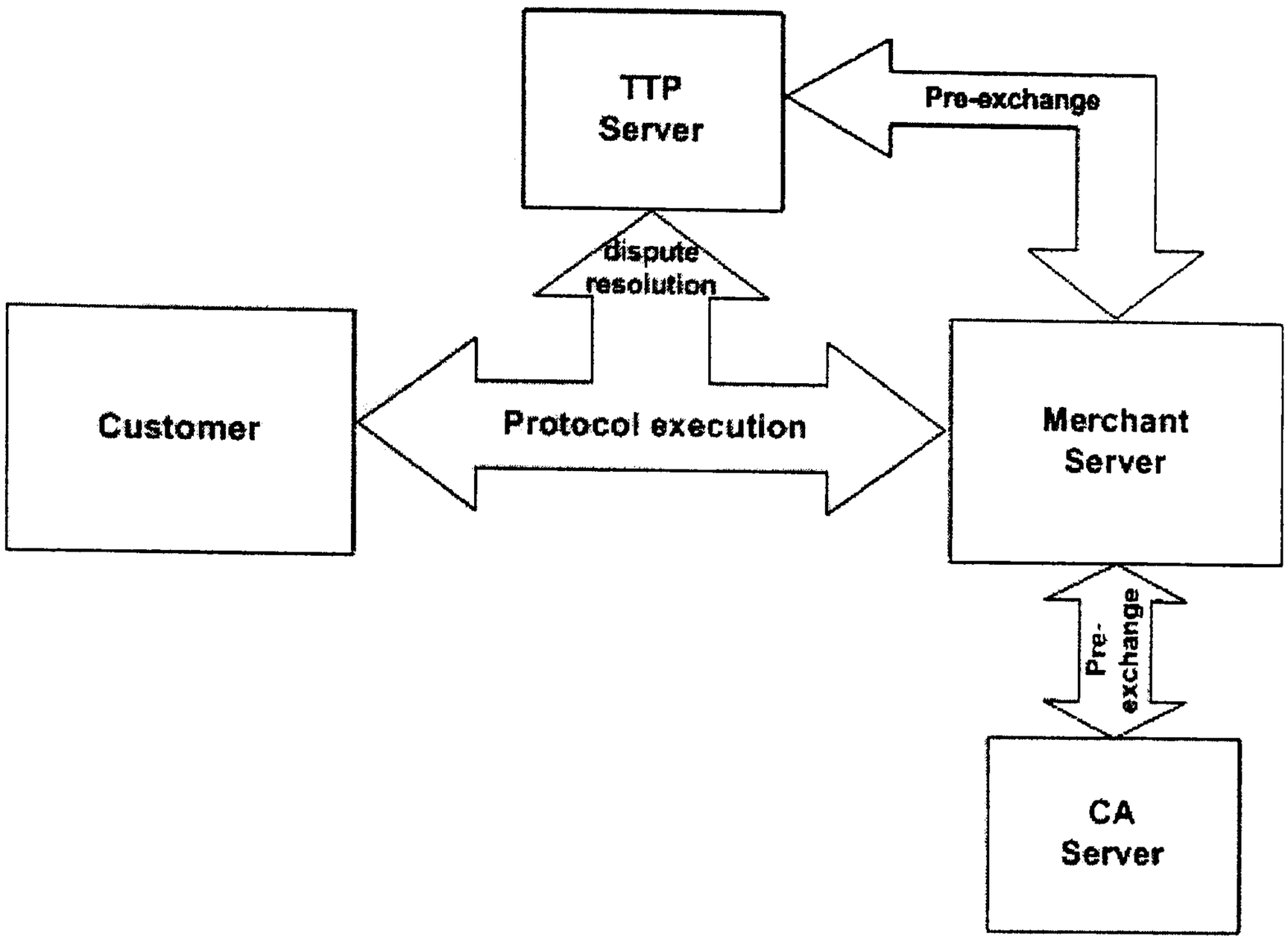

Figure 47: ECH-High level design

The Merchant Server communicates with the TTP Server and the CA Server in the pre-exchange phase. In the pre-exchange phase, the Merchant shares a key with the TTP Server and it also certifies the digital product to be exchanged using the CA Server.

The parties involved in the exchange phase are the Merchant Server and the Customer. In the exchange phase the ECH protocol messages are exchanged between the Merchant Server and the Customer.

The TTP Server will be contacted by the Customer if anything went wrong in the exchange phase. The TTP Server will validate the request and if valid will make the automatic resolution to both the Merchant Server and the Customer.

10.2.1.1. Activity diagrams for ECH protocol

All activities involved in the exchange phase between Customer and the Merchant server are shown in Figures 48 and 49 and if applicable the dispute resolution phase activities are shown in Figures 50, 51, and 52. These activity diagrams are based on the specifications of the ECH protocol in sections 5.2.3 and 5.2.4 of chapter 5.

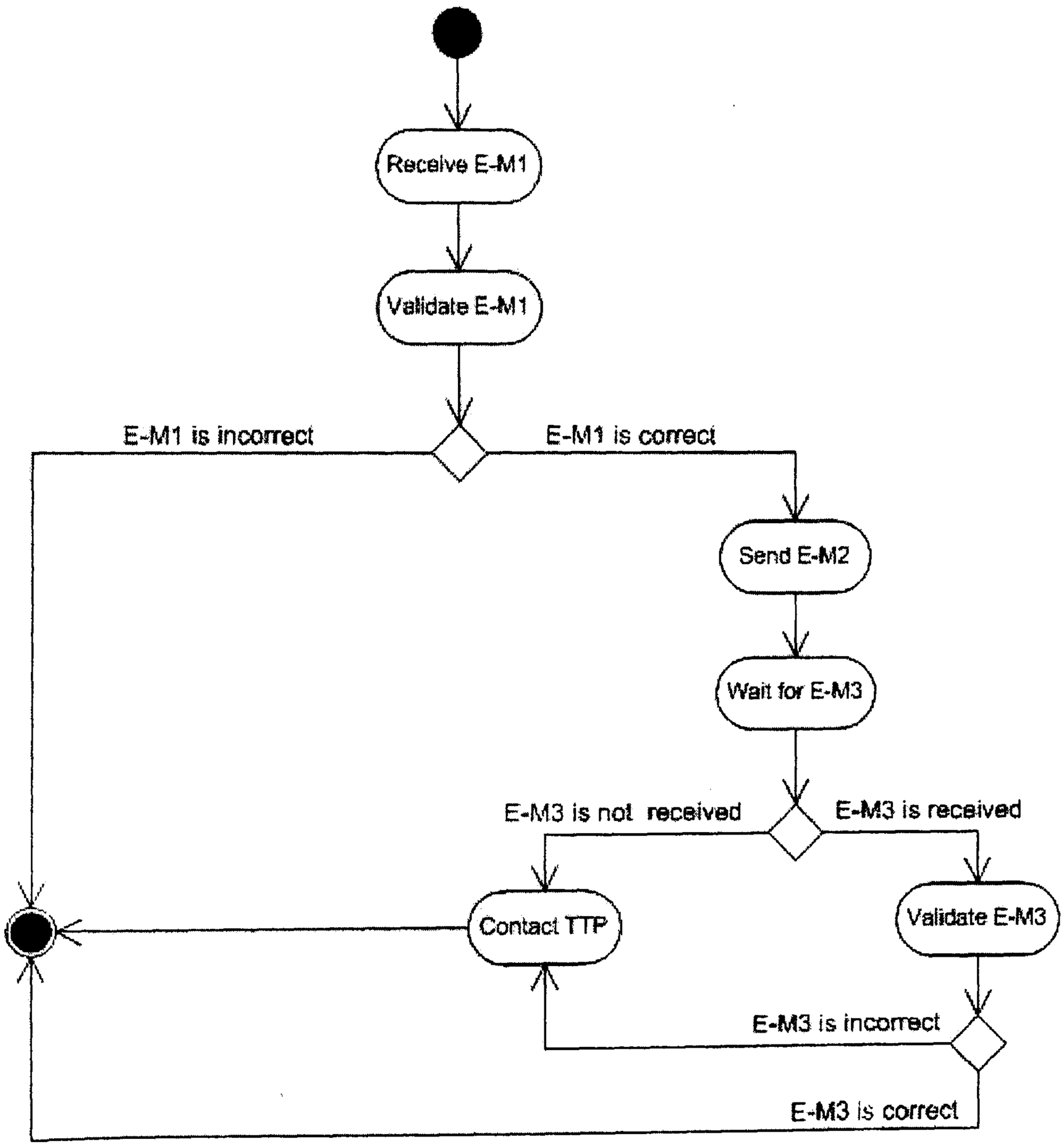

Figure 48: ECH Protocol-Activity Diagram-Exchange-Customer

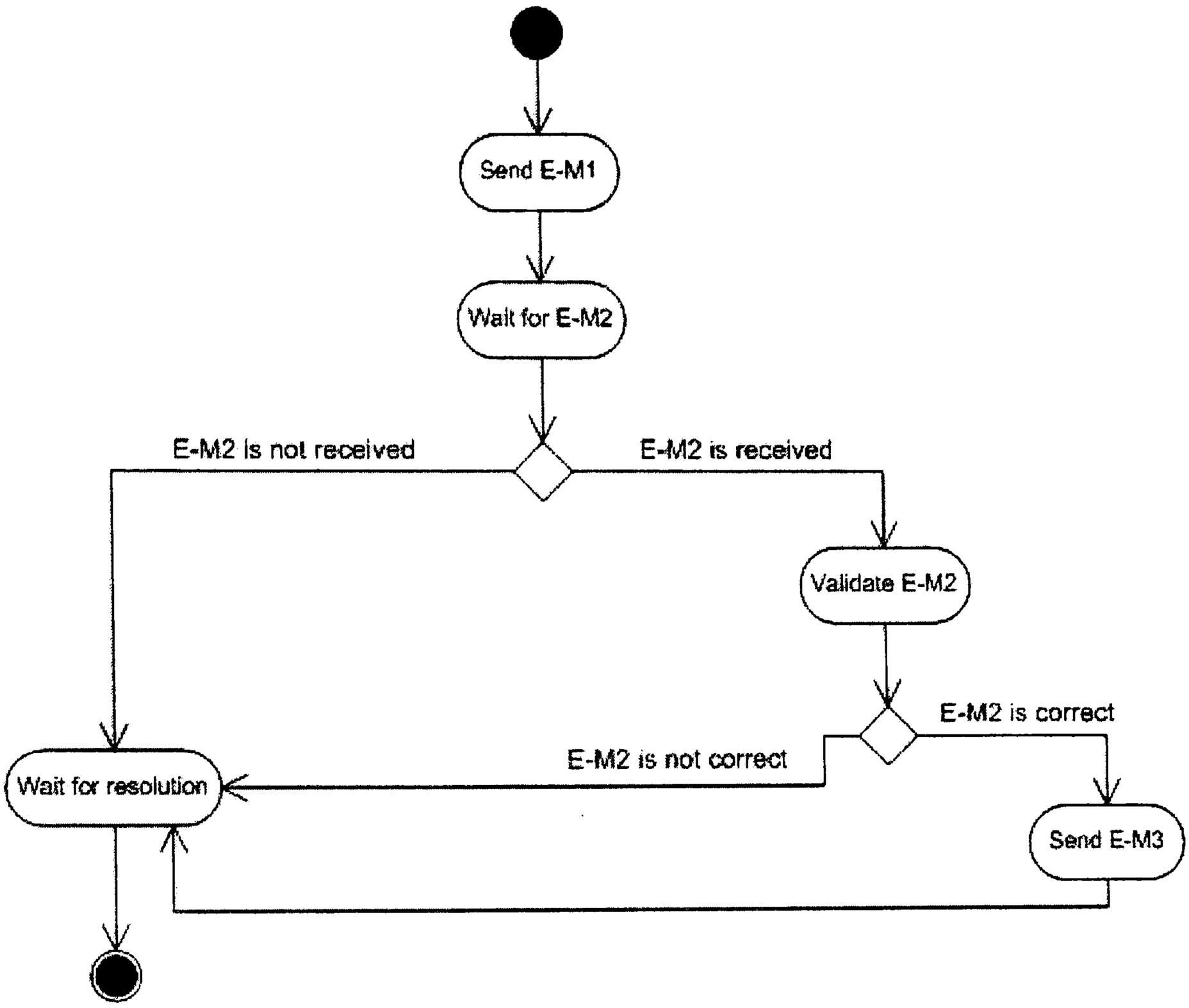

Figure 49: ECH Protocol-Activity Diagram-Exchange-Merchant Server

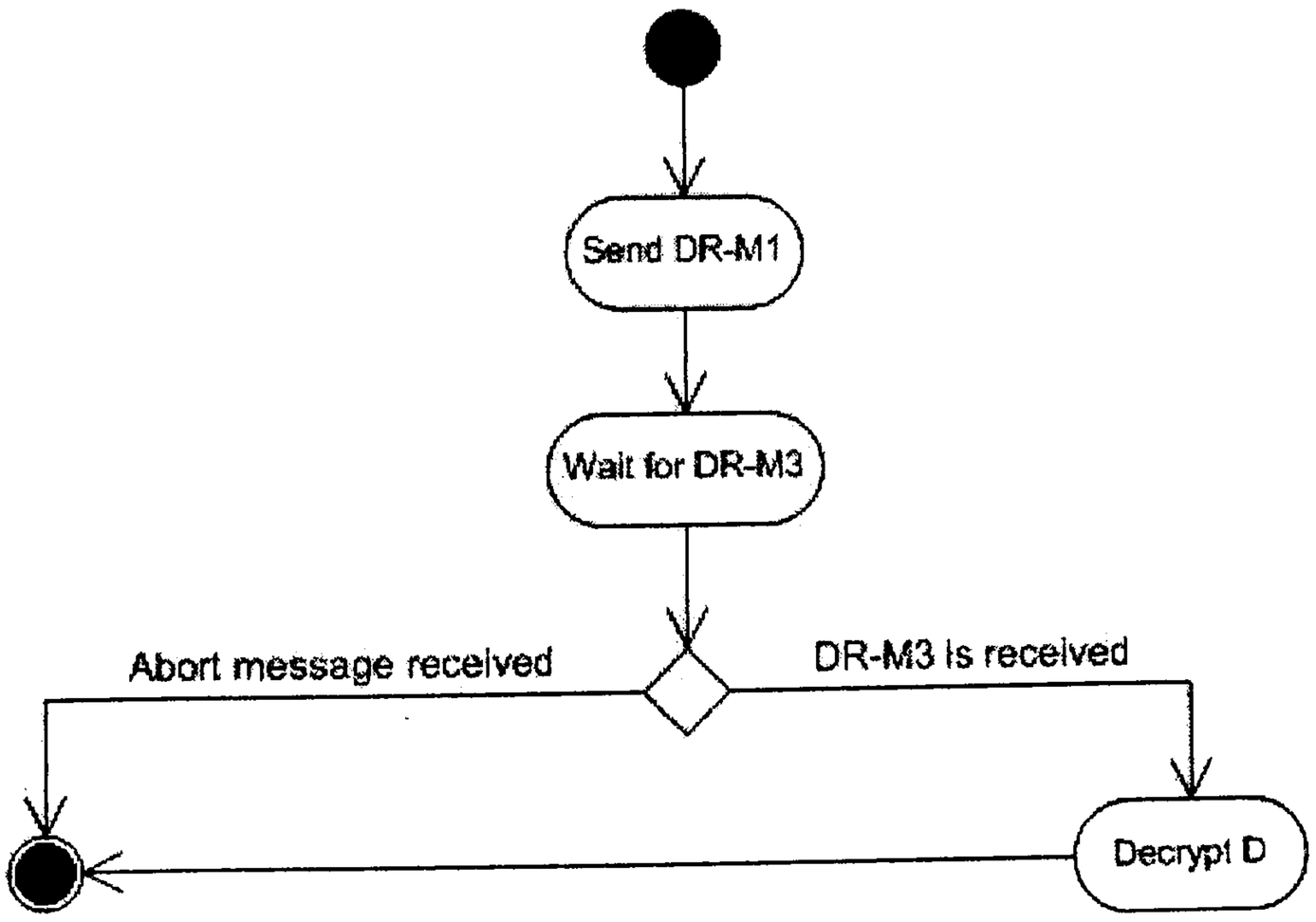

Figure 50: ECH-Activity Diagram-Dispute-Customer

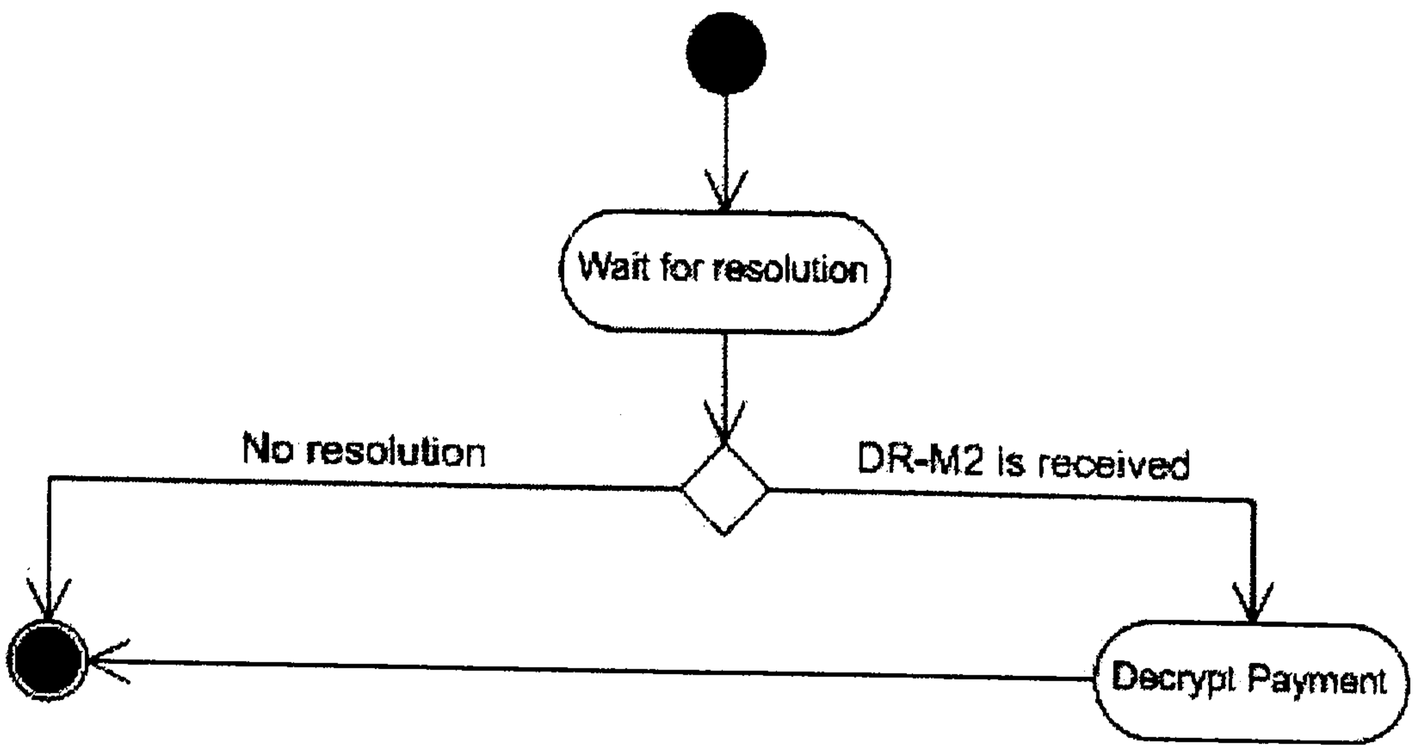

Figure 51: ECH-Activity Diagram-Dispute-Merchant Server

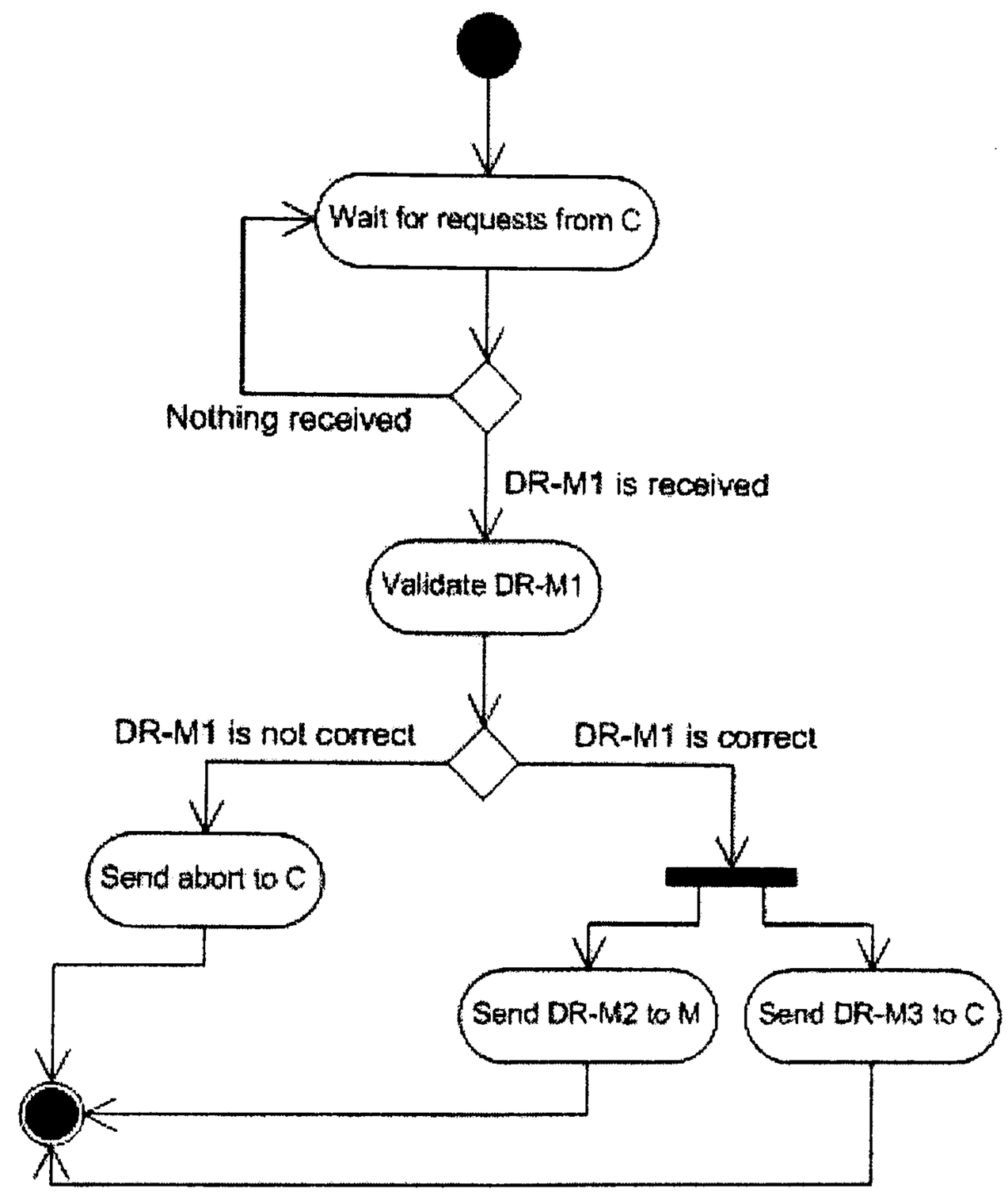

Figure 52: ECH-Activity Diagram-Dispute-TTP Server

10.2.1.2. TTP Server

The TTP Server provides three services to be called by the Merchant Server and Customer. Some of these services will be called in the pre-exchange phase and the other will be called in the dispute resolution phase if there is a dispute.

The Merchant Server calls services in the pre-exchange phase. The pre-exchange services are "Register" and "ShareKey" services. A Merchant Server registers with TTP Server using a username and password. Then, using the username and the password the Merchant Server calls the "ShareKey" service. The TTP Server will verify the correctness of the username and password. If they are correct then the TTP Server will share a key with the Merchant Server and issue the share public key certificate (C.mt) to the Merchant Server to be used in the exchange phase.

The Customer calls the TTP services in the dispute resolution phase if there is a dispute. The Customer needs to send to the TTP Server a correct DR-M1. On receiving the Customer's request the TTP Server will validate the request and if it is correct then an automatic resolution will be sent to both the Merchant Server and Customer.

10.2.1.3. CA Server

The CA Server provides one service to be called by the Merchant Server in the pre-exchange phase. This service is called "Certify". The Merchant Server calls this service by supplying the CA Server with the digital product identifier, symmetric key (km) to be used by the CA Server for computing the hash value of the encrypted digital product with km, the C.mt that the Merchant Server received from the TTP Server, and the price (the value of the digital product) that the Merchant wants. On receipt the CA Server will validate them and if they are correct will issue the digital product's certificate (D-Cert) to the Merchant Server.

10.2.1.4. Merchant Server

The Merchant Server provides three services to be used by Customer and the TTP Server in the exchange phase and the dispute resolution phase, respectively.

Two services to be used by the Customer to execute the exchange phase of the ECH protocol. The first service is "getEM1" that allows Customer to receive the first

message (E-M1) of the exchange phase. The second service is "sendEM2" that is used by Customer to send the second message (E-M2) to the Merchant Server and then used by the Merchant Server to send the third message (E-M3) to the Customer if and only if the Merchant Server found that the E-M2 is correct.

One service to be called by the TTP Server in case the Customer contacted the TTP for dispute resolution and the TTP decided that there is a resolution. That is, if there is no resolution then the TTP Server will not call this service. This service is called "sendDRM2".

10.2.1.5. Customer

The Customer is responsible for communicating with the Merchant Server to execute the exchange phase of the ECH protocol and the TTP Server to request a resolution for any disputes.

The Customer first calls the service "getEM1" which is located in the Merchant Server. On receiving E-M1 the Customer validates it and if correct then "sendEM2" service is called which is also located in the Merchant Server. Using this service the Customer sends E-M2 to the Merchant Server and waits for message E-M3 to be sent by the Merchant Server. On receiving E-M3 and decrypting the encrypted digital product the exchange phase of ECH protocol is complete.

If however the E-M3 is not received or it is incorrect then the Customer will automatically initiate a request to the TTP Server by sending DR-M1 via a "resolve" service. The Customer then waits for the response. The response might be a rejection message or a resolution for the dispute depending on the correctness of the message DR-M1. If the response is a resolution then the customer program will decrypt the encrypted digital product and notify the Customer of the completion of the dispute resolution phase.

10.2.2. High Level Design for EMH protocol

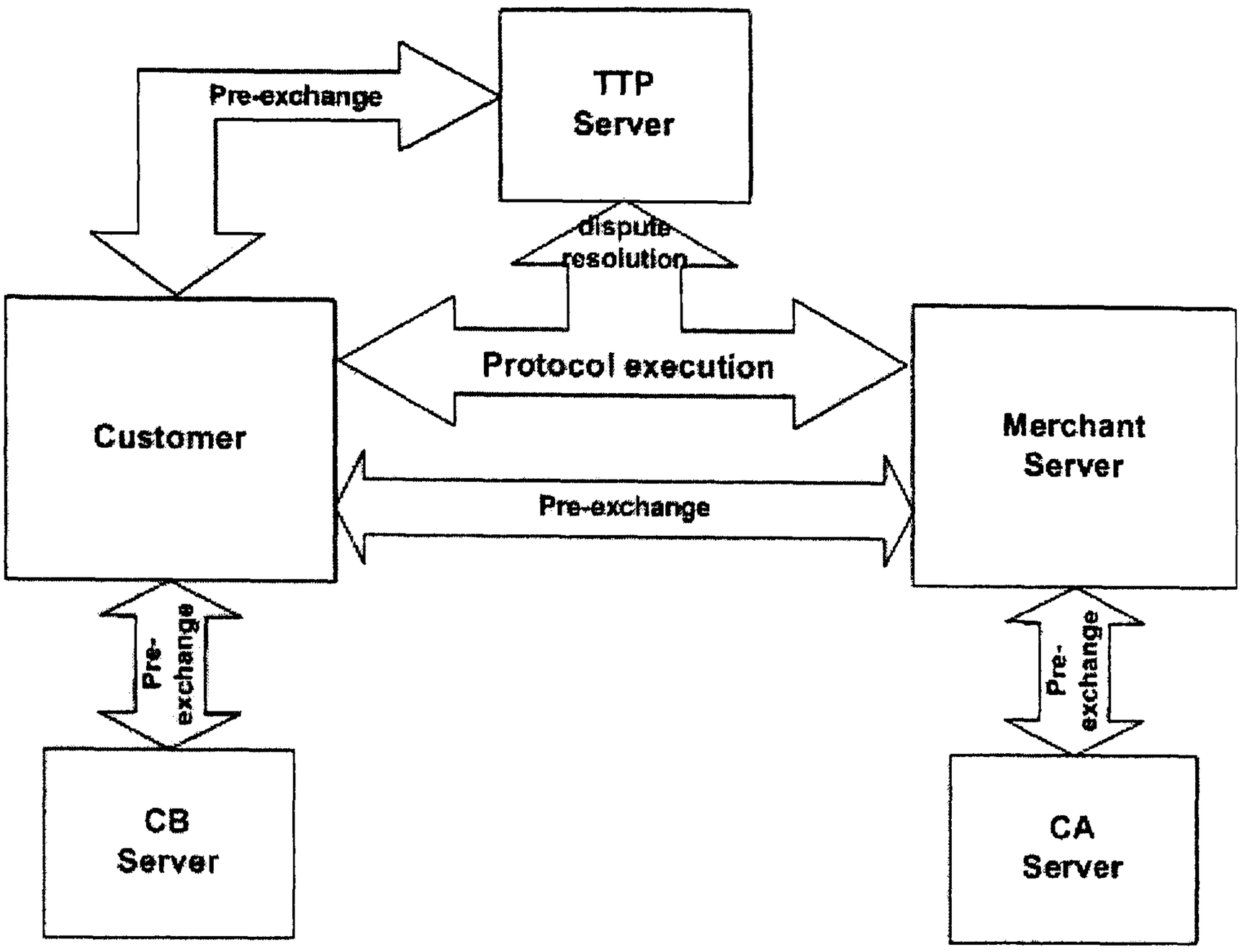

Figure 53: EMH-High level design

The system architecture for EMH protocol consists of the following parties (see Figure 53):

- TTP Server
- CA Server
- CB Server
- Merchant Server
- Customer

In the pre-exchange phase, the Merchant Server communicates with the CA Server to certify the digital product to be exchanged with Customer. In addition, the Customer communicates with the TTP Server to share a key and then communicates with the CB Server to certify the payment to be exchanged. The pre-exchange phase also includes the Customer registration with the Merchant Server and receiving the digital product certificate (DG-Cert) from the Merchant Server.

In the exchange phase Customer and Merchant Server exchange the EMH protocol messages.

The TTP Server will be contacted by the Merchant Server if anything went wrong in the exchange phase. The TTP Server will validate the request and if valid then automatic resolution will be sent to both Customer and Merchant Server.

10.2.2.1. Activity diagrams for EMH protocol

The activity diagrams for the EMH protocol are similar to but not exactly the same as ECH protocol activity diagrams. All activities involved in the exchange phase between Customer and Merchant server are shown in Figures 54 and 55 and if applicable the dispute resolution phase activities diagrams are shown in Figures 56, 57, and 58. These activity diagrams are based on the specifications of the EMH protocol in sections 6.2.3 and 6.2.4 of chapter 6.

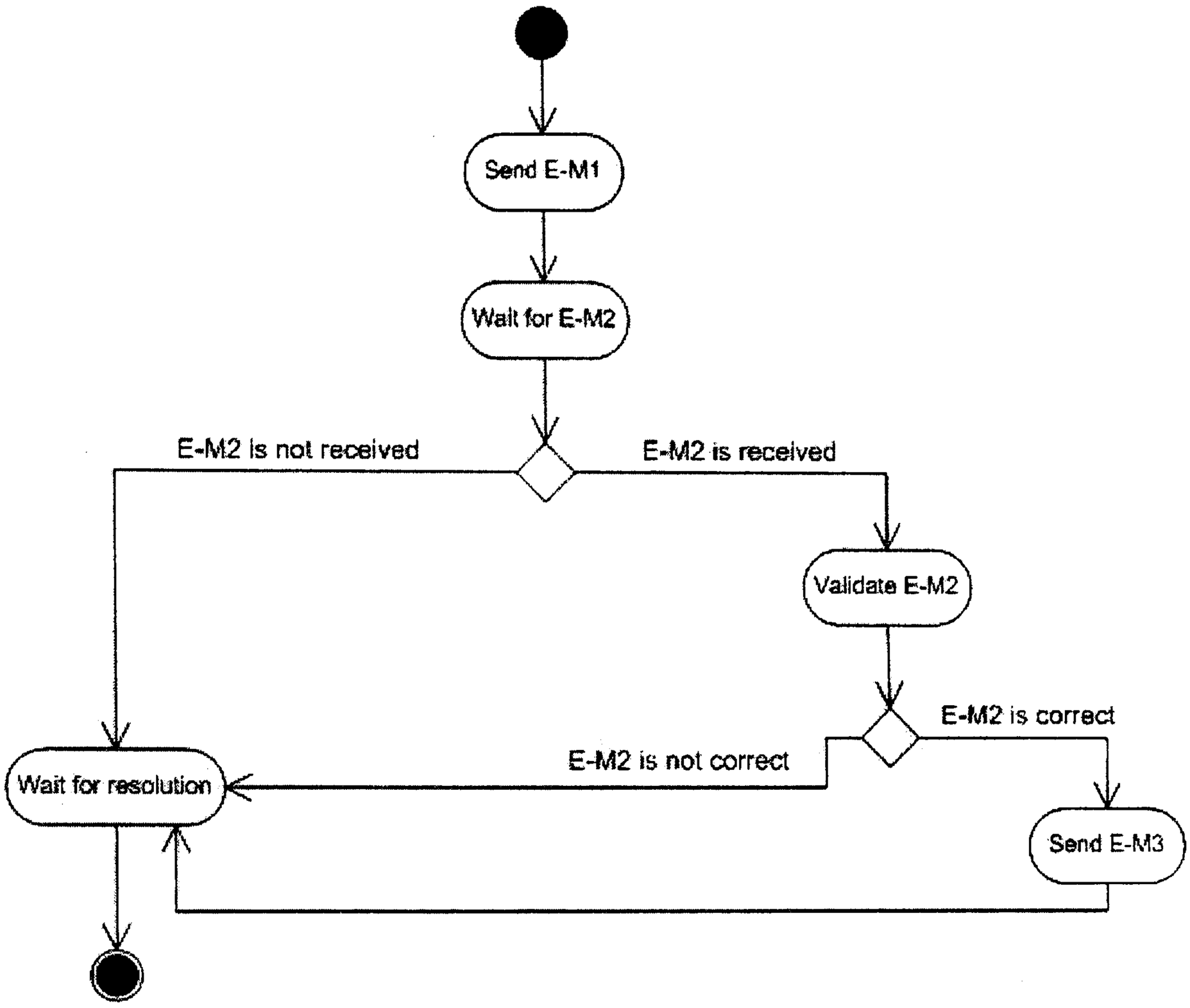

Figure 54: EMH Protocol-Activity Diagram-Exchange-Customer

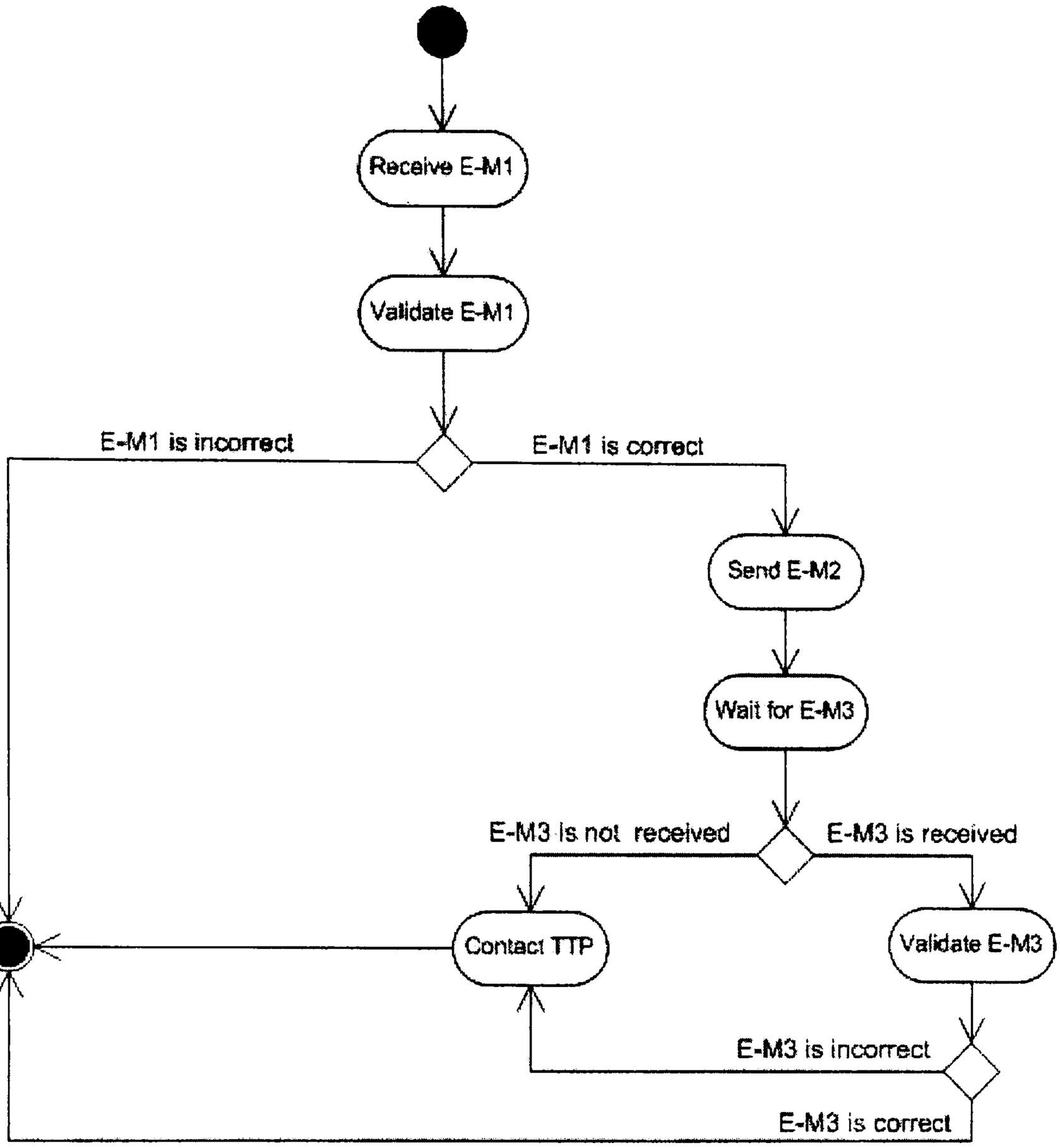

Figure 55: EMH Protocol-Activity Diagram-Exchange-Merchant Server

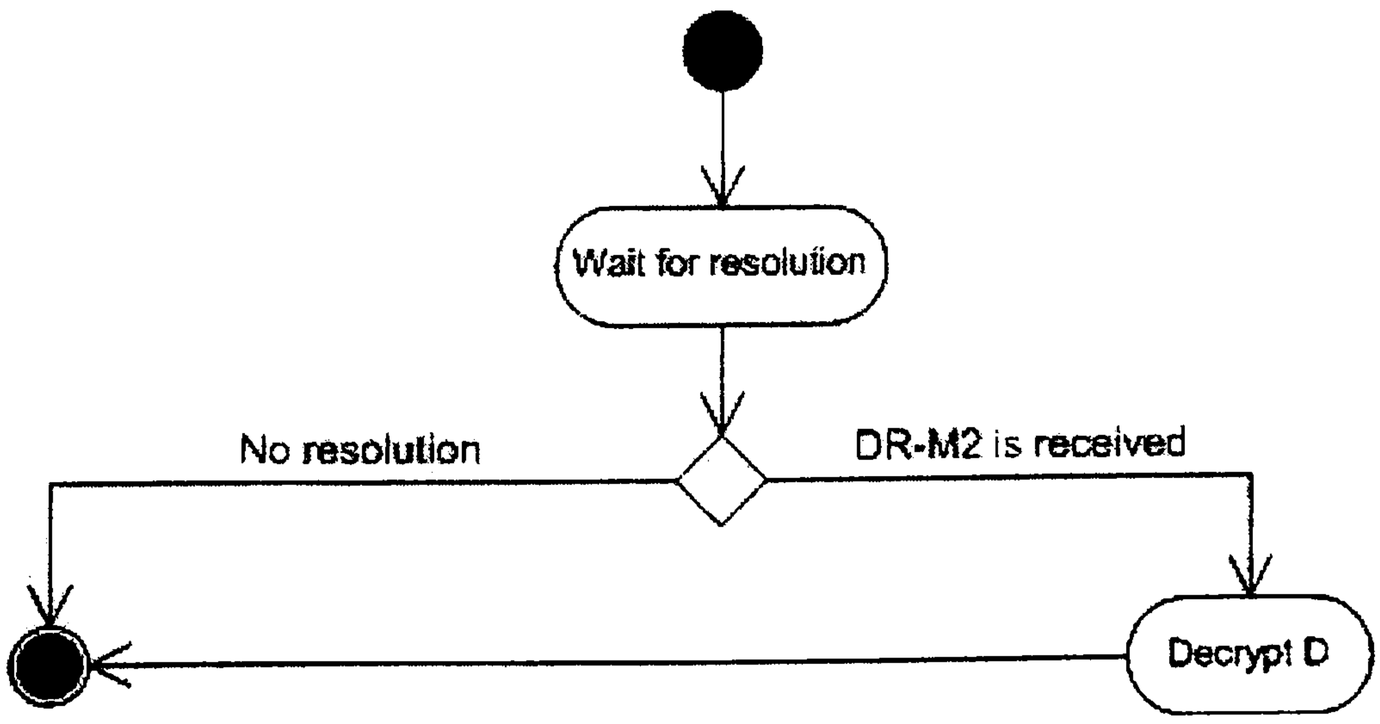

Figure 56: EMH-Activity Diagram-Dispute-Customer

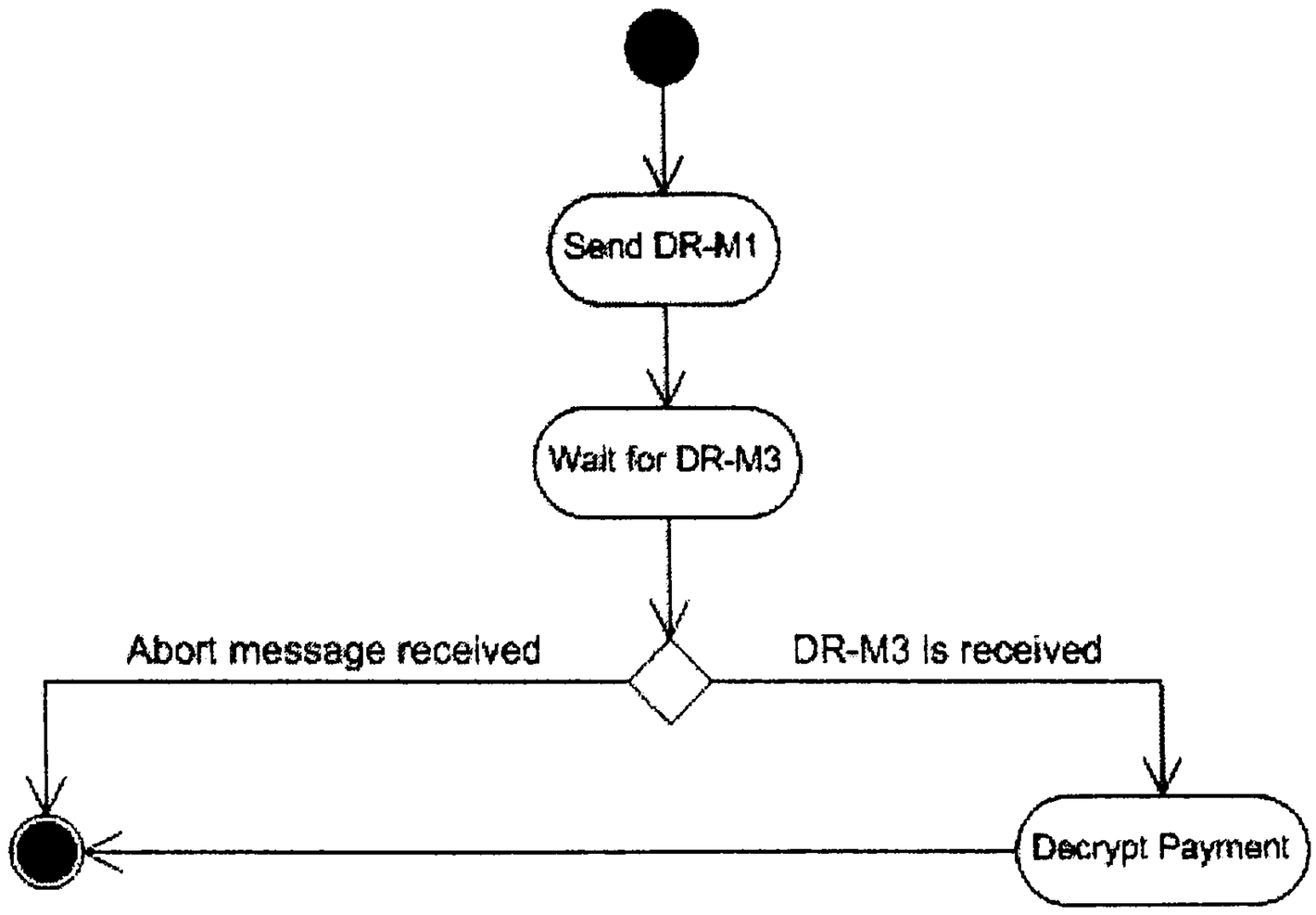

Figure 57: EMH-Activity Diagram-Dispute-Merchant Server

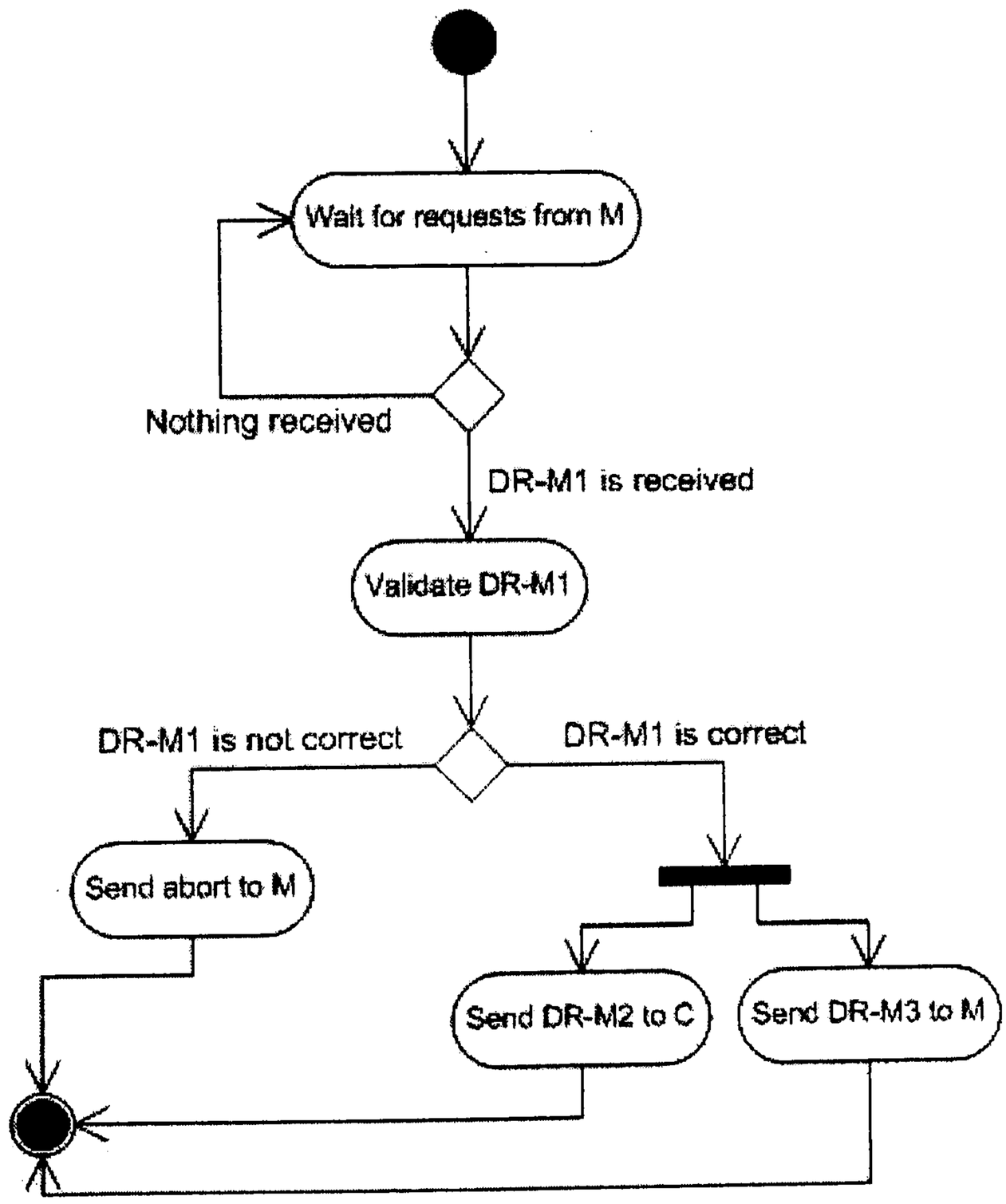

Figure 58: EMH-Activity Diagram-Dispute-TTP Server

10.2.2.2. TTP Server

The TTP Server in the EMH protocol provides the same services provided by the TTP Server in the ECH protocol with the only changes being in the party who calls these services. The Customer is the one who calls the "Register" and "ShareKey" services whereas the Merchant Server is the one who calls the "resolve" service. The certificate to be issued by the TTP Server by calling the "ShareKey" service is C.ct.

10.2.2.3. CA Server

The services provided by the CA Server is the same as the one discussed in the ECH protocol, the only change is in the digital certificate issued by the CA Server. In here the certificate to be issued is DG-Cert. Additionally, in order for the Merchant Server to get DG-Cert they need to supply the CA Server with the digital product identifier and the price of the digital product.

10.2.2.4. CB Server

The CB Server provides one service, "certify", to be called by the Customer in the pre-exchange phase of the protocol. The Customer needs to supply the CB Server with a symmetric key (kc) to be used by the CA Server for computing the hash value of the encrypted payment with kc, C.ct received from the TTP Server, payer, payee, and the amount of payment to be certified. On receiving these items the CB Server will validate them and if they are correct then it will issue the payment's certificate (P-Cert) to the Customer.

10.2.2.5. Merchant Server

The Merchant Server in the EMH protocol is different from the one in the ECH protocol and provides three services which are all called by the Customer. Two

of the services are called in the pre-exchange phase and one is called in the exchange phase.

The services to be called in the pre-exchange phase are "register" service which allows the Customer to register with the Merchant Server and the "getDGCert" which allows the Customer to receive the digital product certificate (DG-Cert) that will be used in constructing message E-M1 of the exchange phase.

The service to be called by the Customer in the exchange phase is "sendEM1". This service allows the Customer to send the message E-M1 to the Merchant Server. On receiving E-M1 from the Customer, the Merchant Server will validate it and if it is correct then it will call the service "sendEM2" that is defined in the Customer side to send E-M2. Then, the Merchant Server will wait for the message E-M3 from the Customer and on receiving it and decrypting the encrypted payment the exchange phase of EMH protocol is complete.

If however the E-M3 is not received or is incorrect then the Merchant Server will automatically initiate a request to the TTP Server by sending DR-M1 via the "resolve" service. The response might be a rejection message or a resolution for the dispute depending on the correctness of the message DR-M1. If the response is a resolution then the Merchant Server will decrypt the encrypted payment and then the dispute resolution phase is complete.

10.2.2.6. Customer

The Customer is different from the one in the ECH protocol in that it provides services for the Merchant Server and the TTP Server. The service for the Merchant Server is to be called in the exchange phase of the EMH protocol whereas the service for the TTP Server is to be called in the dispute resolution phase if there is a dispute. The service to be called by the Merchant Server is "sendEM2" while the service to be called by the TTP Server is "sendDRM2".

The Customer program starts the exchange phase by sending message E-M1 via the service "sendEM1" which is located in the Merchant Server. If the Merchant Server found E-M1 is correct then the Customer's service "sendEM2" will be called by the Merchant Server to send E-M2. On receiving E-M2, the Customer will verify it and if it is correct then the digital product will be decrypted and the message E-M3 will be sent to the Merchant Server.

The service "sendDRM2" will be called by the TTP Server if and only if the Merchant Server contacted the TTP Server for resolution and the TTP Server decided that there is a resolution. That is, if there is no resolution then the TTP Server will not call this service.

10.2.3. High Level Design for ECMH protocol

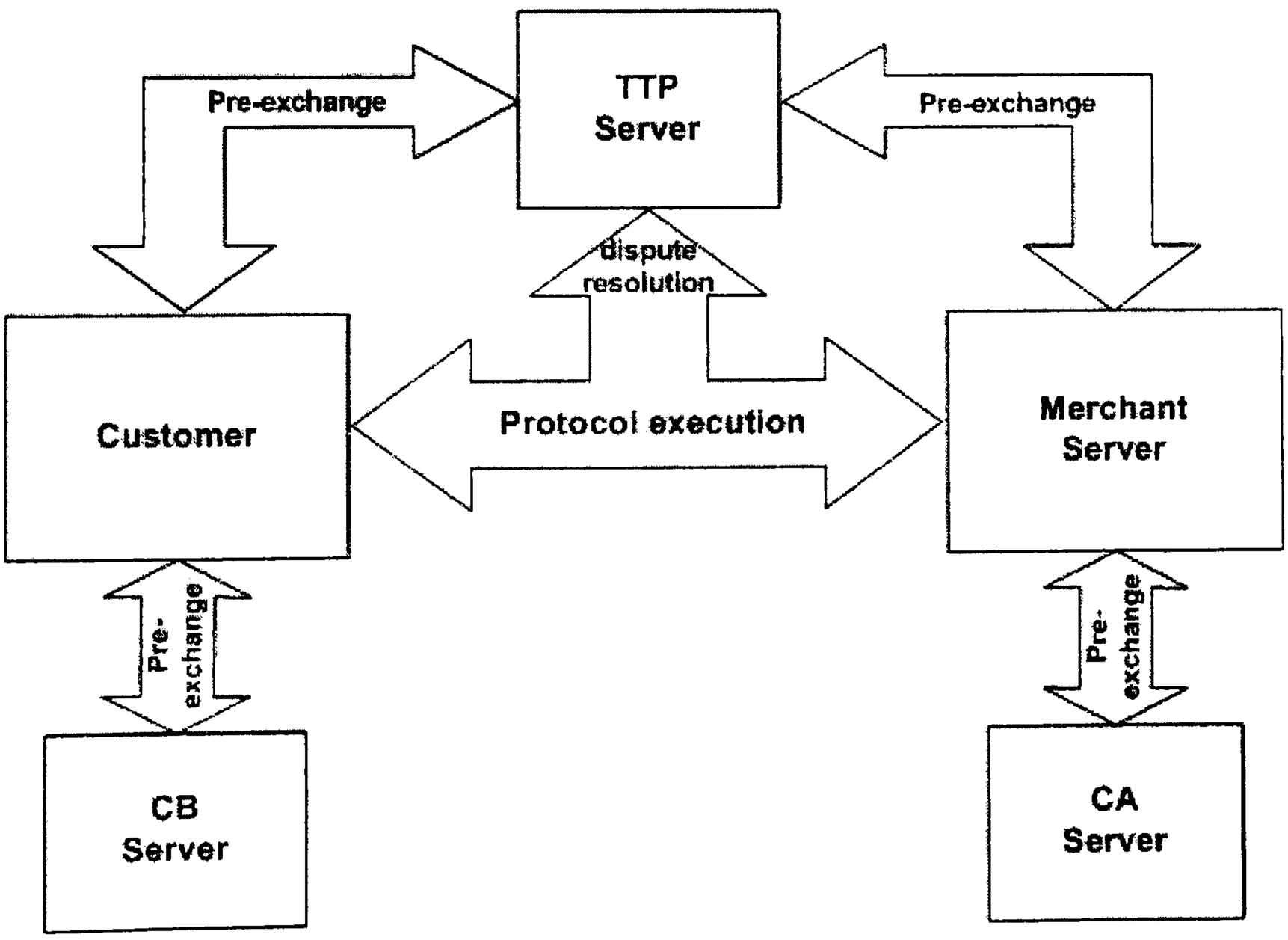

Figure 59: ECMH-High level design

The parties involved in the system architecture for ECMH protocol are the same as the ones appear in the EMH protocol. There are three differences between the two designs. The first one is that in the ECMH the Merchant Server needs to communicate with TTP Server in the pre-exchange phase to share a key. The second one is that in the ECMH Customer is the party who contacts TTP Server in case of disputes. Therefore, the TTP Server will validate Customer's request and if valid then the automatic resolution will be sent to both the Merchant Server and Customer. The third difference is that in the ECMH the Customer does not need to contact the Merchant Server in the pre-exchange phase. The high level design for the ECMH protocol appears in Figure 59.

10.2.3.1. Activity diagrams for ECMH protocol

The activity diagrams for ECMH protocol are similar to but not exactly the same as the activity diagrams for the ECH and EMH protocols. All activities involved in the exchange phase between Customer and Merchant Server are shown in Figures 60 and 61 and if applicable the dispute resolution phase activity diagrams are shown in Figures 62, 63, and 64. These activity diagrams are based on the specifications of the ECMH protocol in sections 7.2.3 and 7.2.4 of chapter 7.

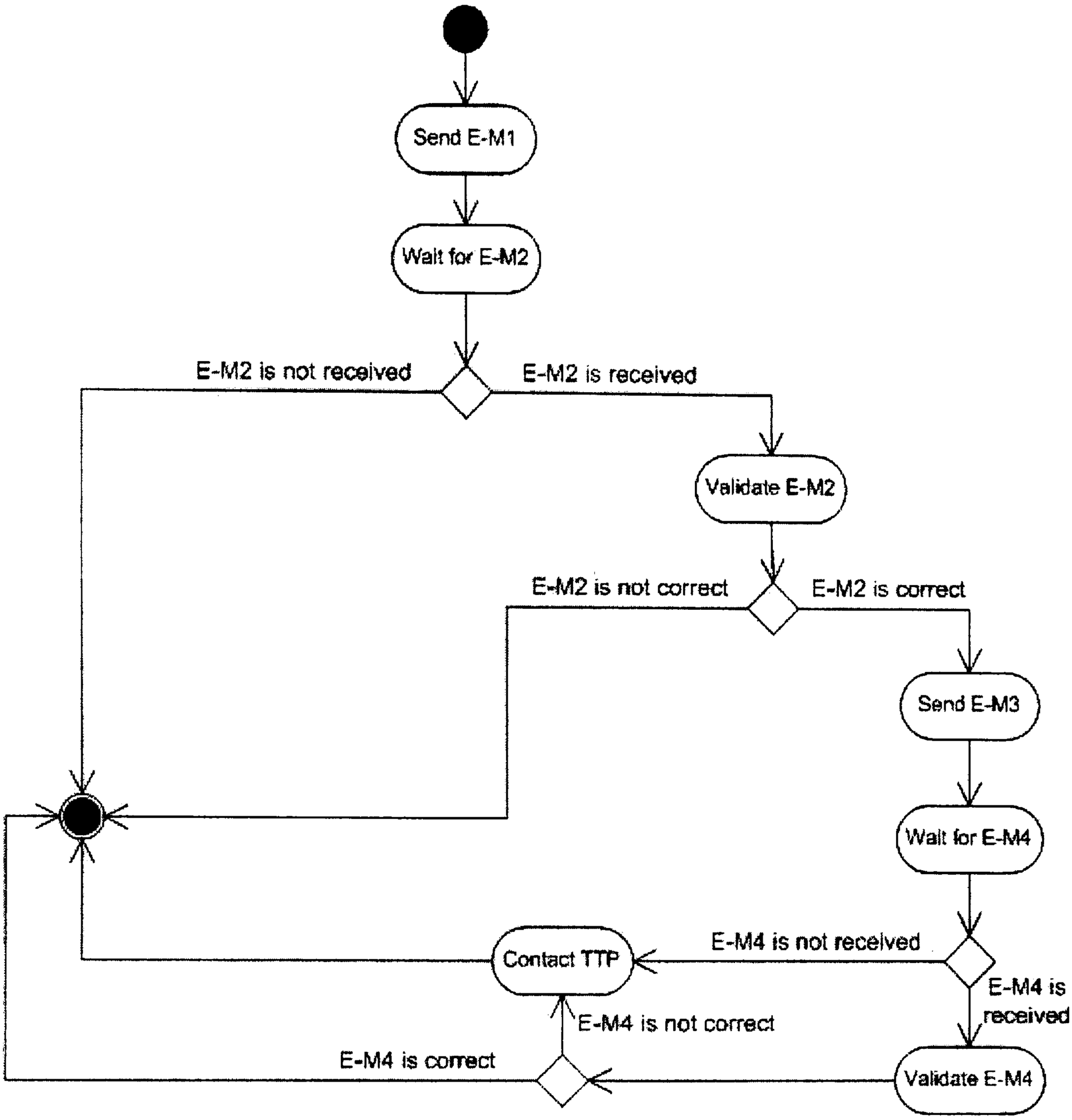

Figure 60: ECMH Protocol-Activity Diagram-Exchange-Customer

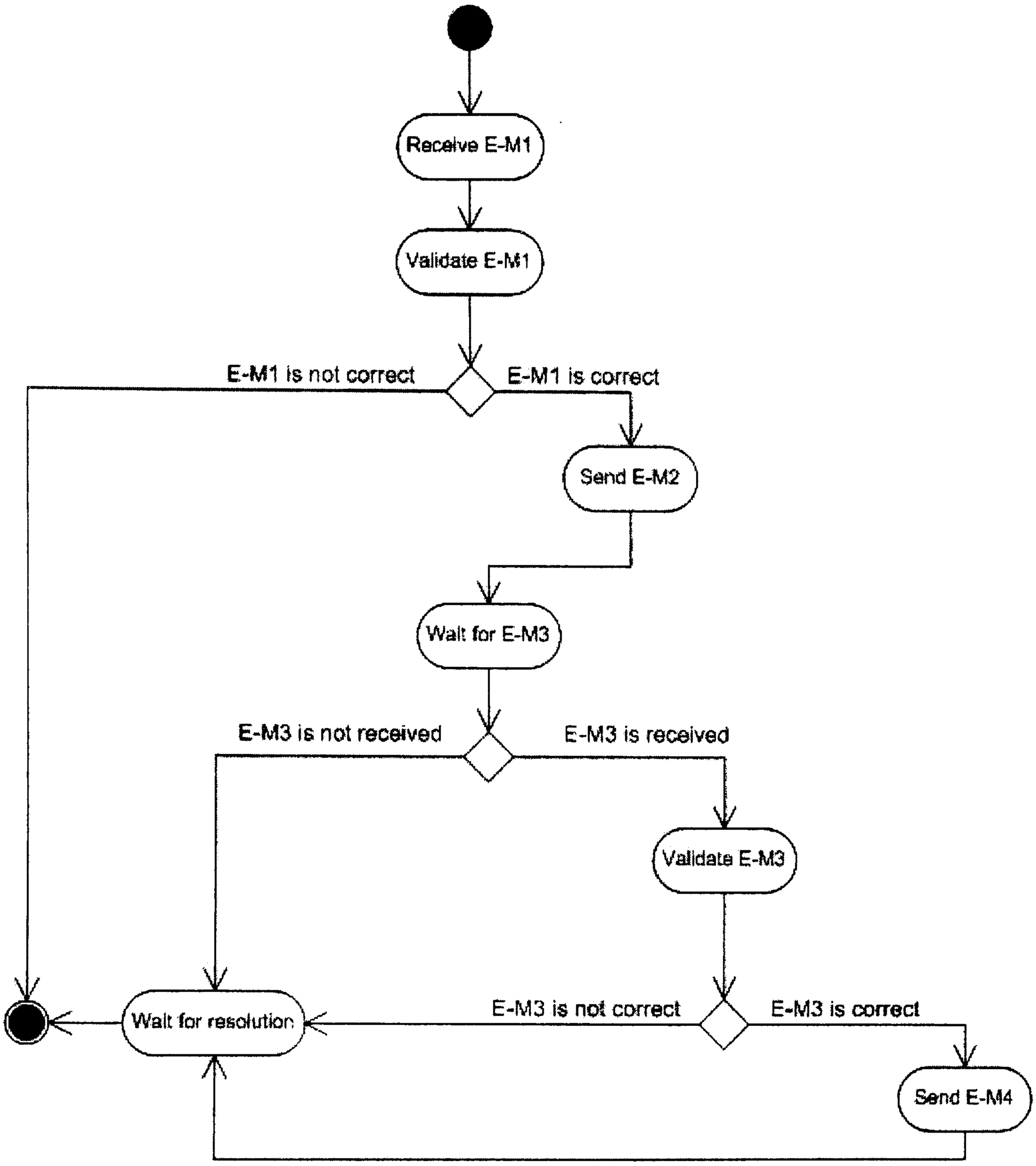

Figure 61: ECMH Protocol-Activity Diagram-Exchange-Merchant Server

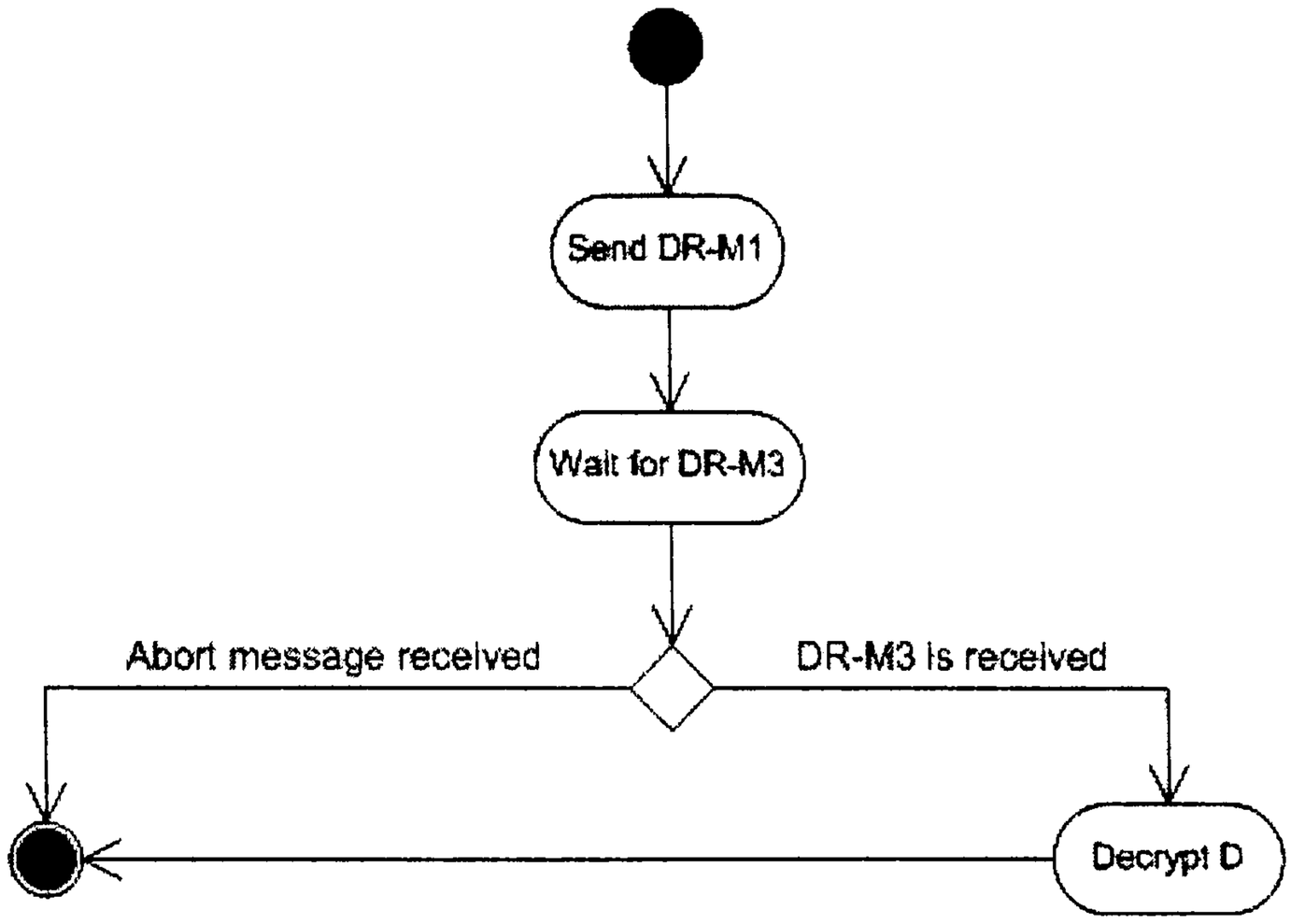

Figure 62: ECMH-Activity Diagram-Dispute-Customer

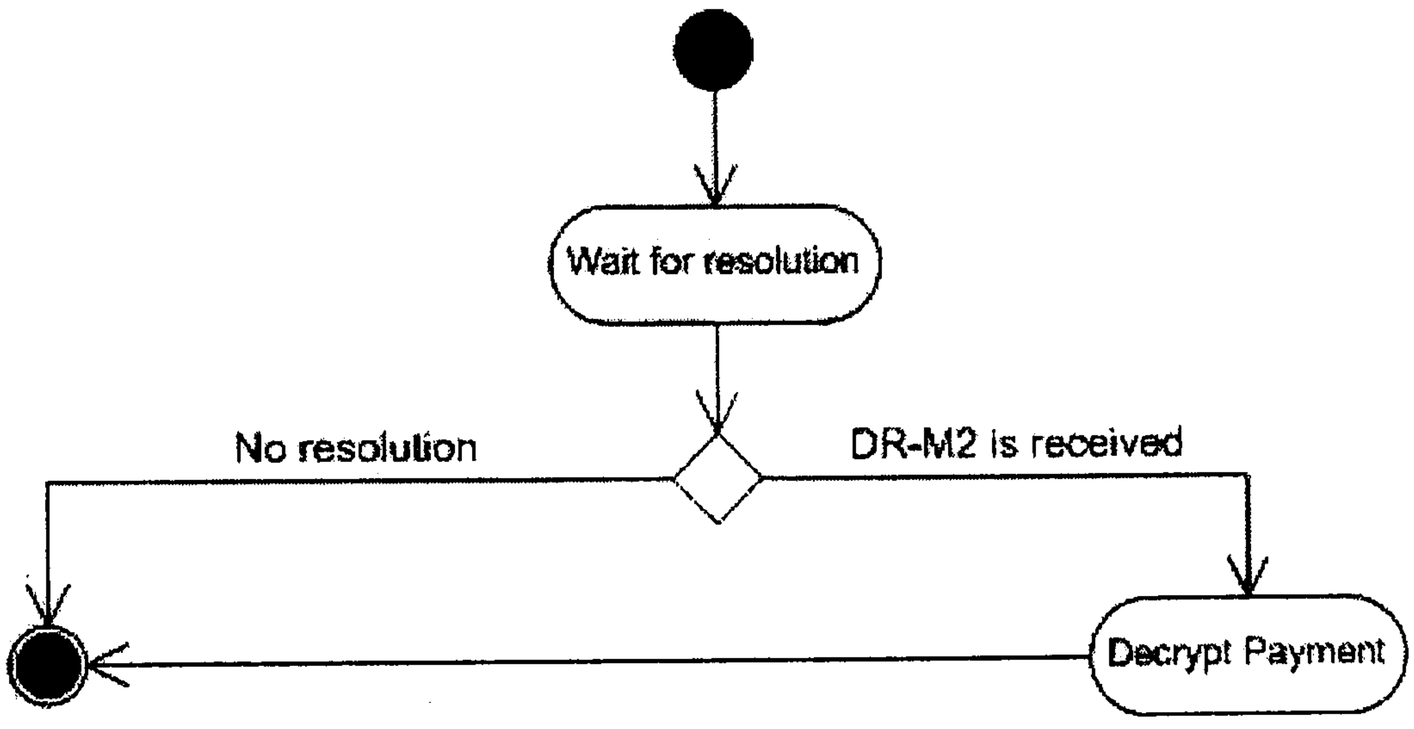

Figure 63: ECMH-Activity Diagram-Dispute-Merchant Server

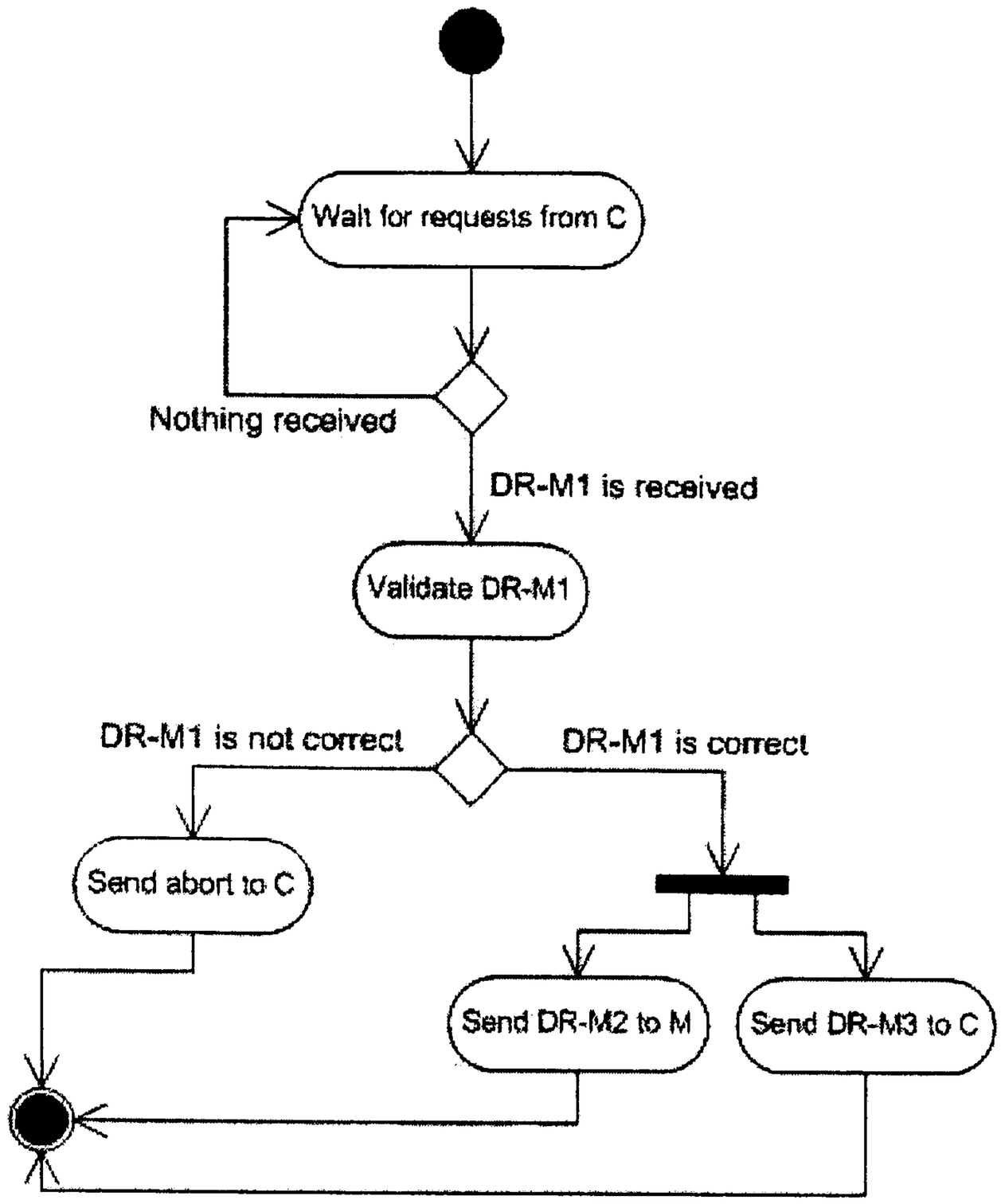

Figure 64: ECMH-Activity Diagram-Dispute-TTP Server

10.2.3.2. TTP Server

The services provided by the TTP Server in the ECMH protocol are the same as the ones provided in the EMH protocol. These services are "Register" service, "ShareKey" service, and "resolve" service. The "Register" service and the "ShareKey" service will be called by both the Merchant Server and the Customer in the pre-exchange phase. The "resolve" service however will only be called by the Customer in the dispute resolution phase if something went wrong in the exchange phase.

10.2.3.3. CA Server

The service provided by the CA Server is exactly the same as the one provided by the CA Server in the ECH protocol.

10.2.3.4. CB Server

The service provided by the CB Server is exactly the same as the one provided by the CB Server in the EMH protocol.

10.2.3.5. Merchant Server

The Merchant Server provides three services which are "sendEM1", "sendEM3", and "sendDRM2". The services "sendEM1" and "sendEM3" will be called by the Customer in the exchange phase whereas the service "sendDRM2" will be called by the TTP Server in the dispute resolution phase in case the Customer contacts the TTP Server for dispute resolution and there is a resolution. That is, if the Customer contacts the TTP Server for a dispute resolution but the TTP Server decided that there is no resolution then the TTP Server will not call the service "sendDRM2".

10.2.3.6. Customer

The Customer is the one who initiates the exchange phase of the ECMH protocol. It does so by first sending the message E-M1 via the service "sendEM1" that is located in the Merchant Server. Then, the Customer will wait for the message E-M2 to be sent by the Merchant Server. On receiving the message E-M2 the Customer will validate it and if correct then the Customer will send the message E-M3 via the service "sendEM3" that is located in the Merchant Server. Then, the Customer will wait for the message E-M4 to be sent by the Merchant Server. On receiving E-M4 and decrypting the encrypted digital product the exchange phase of the ECMH protocol is complete.

If however the E-M4 is not received or is incorrect then the Customer will automatically initiate a request to the TTP Server by sending DR-M1 via the "resolve" service. The response from the TTP Server might be a rejection message or a resolution for the dispute depending on the correctness of the message DR-M1. If the

response is a resolution then the Customer will decrypt the encrypted digital product and then the dispute resolution phase is complete.

10.3. The Tools

This section will discuss the tools that have been implemented to perform the scenarios of the ECH, EMH, and ECMH protocols. The tools for the ECH protocol and the EMH protocol are very similar to each other, and thus only the tools for the ECH protocol and the ECMH protocol will be presented.

10.3.1. The ECH Protocol Tool

This tool allows its user to perform the execution of the ECH protocol under some conditions. These conditions include having one or both parties honest, having one or both parties dishonest, and the number of times the ECH protocol to be executed. If the user chooses a party to be dishonest then they will be allowed to choose the type of dishonesty for that party. There are three types of dishonesty for M and two types of dishonesty for C that affect the fairness (these types were discussed in section 5.3.1 of chapter 5). The types of dishonesty for M are:

1. after receiving a correct E-M2 from C, M will not send E-M3,
2. after receiving a correct E-M2 from C, M will send incorrect E-M3, and
3. M will send to C incorrect E-M1

The types of dishonesty for C are:

1. after receiving a correct E-M1 from M, C will send to M incorrect payment in E-M2, and
2. after receiving a correct E-M1 from M, C will encrypt the payment with incorrect key in E-M2.

After executing the ECH protocol under the entered conditions, the user will be shown the messages that have been exchanged between C and M. In addition, if the TTP is contacted by C for resolution then the messages will also be presented.

The user will also be presented with a summary of the transactions that have been executed. The summary includes the number of times the TTP was contacted, the number of times the TTP rejected the dispute, the number of times the TTP resolved the disputes, the average time for E-M1 construction by M, the average time for E-M1 verification by C, the average time for E-M2 construction by C, the average time for E-M3 construction by M, the average time for decrypting the digital product, the average time for decrypting the payment, the average time for DR-M1 verification by TTP (if applicable as the TTP will not always be contacted).

Finally the tool presents a table that shows the scenarios that have been executed. Figure 65 shows the tool for the ECH protocol.

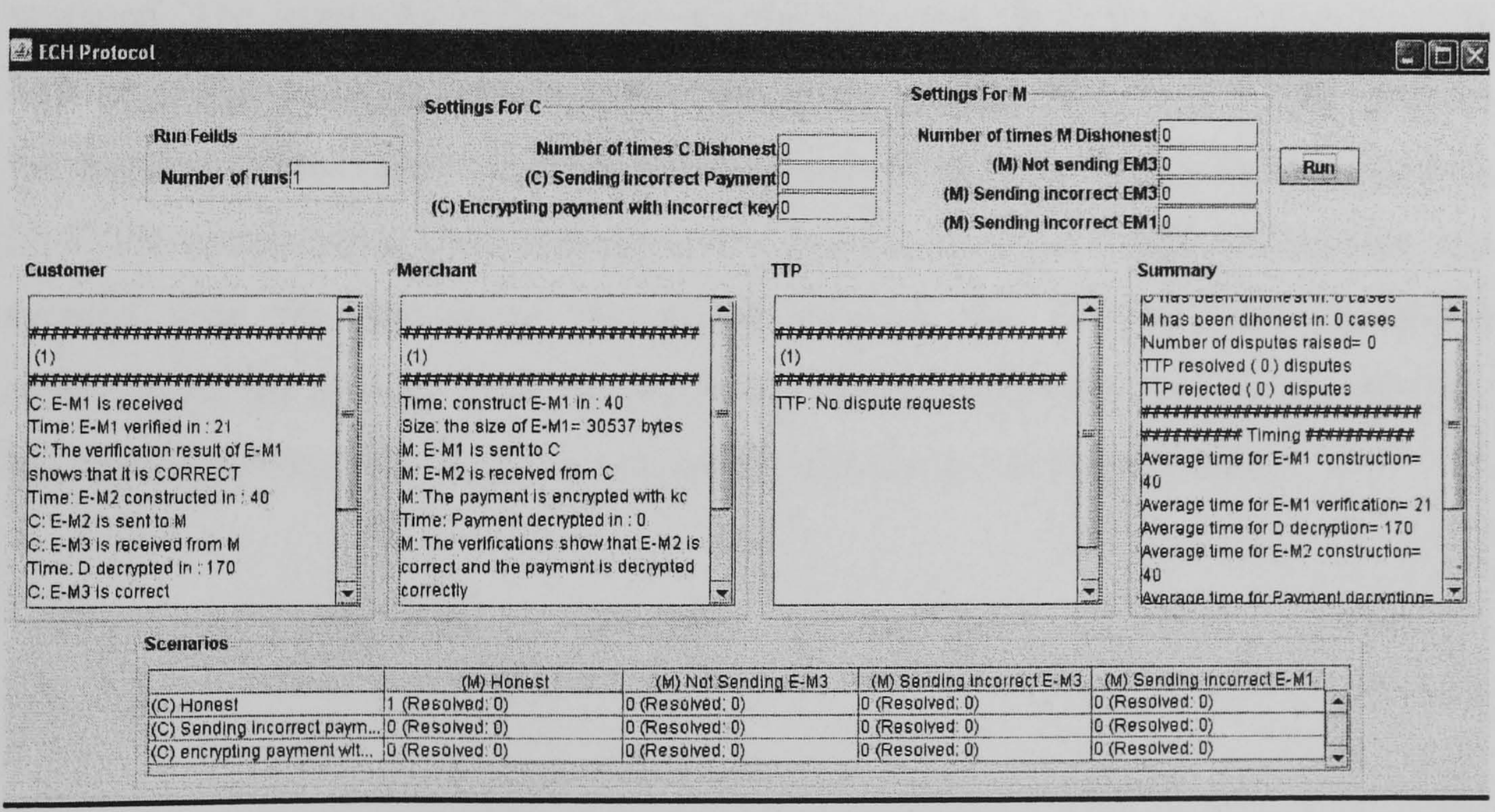

Figure 65: The Tool for ECH Protocol

10.3.2. The ECMH Protocol Tool

This tool is similar to the one presented in the previous section, the difference however is in the type of dishonesty for C and M. There are three types of dishonesty for M and two types of dishonesty for C (these types were discussed in section 7.3.1 of chapter 7).

The types of dishonesty for M are:

1. after receiving a correct E-M3 from C, M will not send E-M4,
2. after receiving a correct E-M3 from C, M will send incorrect E-M4, and
3. after receiving a correct E-M1 from C, M will send to C incorrect E-M2.

The types of dishonesty for C are:

1. C will send to M incorrect E-M1, and
2. after receiving a correct E-M2 from M, C will send to M incorrect E-M3

Output from the tool showing a summary of the transactions those have been executed. The summary includes the number of times the TTP was contacted, the number of times the TTP rejected the dispute, the number of times the TTP resolved the disputes, the average time for E-M1 construction and verification, the average time for E-M2 construction and verification, the average time for E-M3 construction, the average time for decrypting the digital product, the average time for E-M4 construction, the average time for decrypting the payment, the average time for DR-M1 verification by TTP. Figure 66 shows the tool for the ECMH protocol.

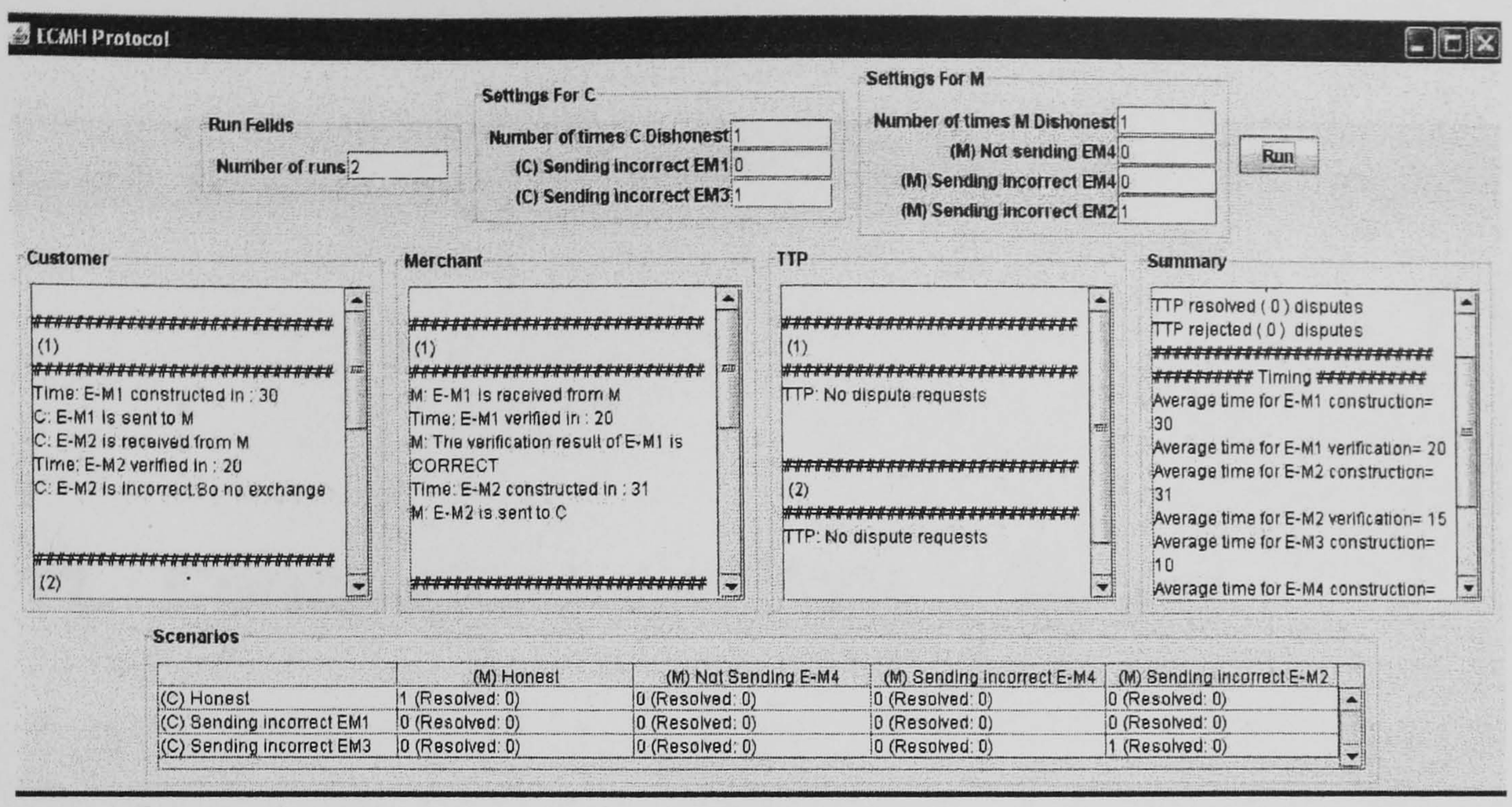

Figure 66: The Simulation Tool for ECMH Protocol

10.4. Summary

This chapter presented the design and implementation of the prototype proof of concept of the ECH, EMH and ECMH protocols. It also presented the tools that have been implemented to easily execute all the scenarios of the protocols.

Chapter Eleven

11. Conclusions and Future Work

11.1. Introduction

As the Internet grows its use in selling and buying products grow as well. More and more merchants offer their products and services on the Internet every day. These products and services on the Internet have attracted customers.

Buying products and services from the Internet requires vigilance on the part of customers when buying from unknown (and possibly dishonest) merchants. This is because after the customer pays the money to the merchant, the merchant might not send the product to the customer or might send incorrect product. Therefore, there is a need for fair exchange protocols to ensure that the customer receives the correct product and also the merchant receives the correct payment.

Two types of fair exchange protocols exist. The first type is one that does not involve a Trusted Third Party (TTP). In this type of protocol, the two parties exchange their items part by part. That is, the parties divide their items into parts and then exchange these parts until the whole items are exchanged. Therefore, many rounds are required to complete the exchange of the items between the two parties. Additionally, fairness is not ensured because one party needed to send their last part first to the other party. As a result, the party who receives the last part of the other party's item may disappear before sending their last part.

The second type is the one that involves a TTP. This type is divided into three kinds according to how the TTP is involved. The first kind is the one that involves an inline TTP where the parties send their items to the TTP who verifies them and if they are correct then forwards the relevant item to the other party. In the inline TTP based fair exchange protocols, the TTP is actively involved. This results in extra costs and the TTP may become a source of communication bottleneck. Furthermore, the TTP can be the main target of attacks. The second kind is the one that involves an online TTP where the parties exchange their items directly but the TTP is used during the exchange for validation purposes and for generating and/or storing evidence of a transaction. This kind of protocol reduces the use of the TTP but it still requires that the TTP to be available during the exchange of items. This requirement is seen as a disadvantage as it will result in extra messages to be exchanged as well as the TTP is the main target of attacks. The third kind of protocol is the one that involves an offline TTP (an optimistic fair exchange protocol) where the parties exchange their items directly. The TTP will only be involved if something goes wrong between the parties. Therefore, the disadvantages of the inline and online TTP based protocols are minimised because the use of the TTP is kept to minimum.

This thesis has focused on optimistic fair exchange protocols that are for the exchange of payments and digital products between customers and merchants.

The research has developed new concept in fair exchange protocols, that of enforcing one of the parties to be honest. Two different but similar optimistic fair exchange protocols have been specified, one enforces the customer to be honest (ECH protocol) and the other enforces the merchant to be honest (EMH protocol). Both protocols have been shown to be efficient as compared with existing protocols in that they require less messages and reduce the number of modular exponentiations. These protocols also simplify the process of automatic dispute resolution by reducing the number of possible incidents of disputes.

Enforcing both parties to be honest within one optimistic fair exchange protocol whilst being a logical consequence of the ECH and EMH protocols was shown not to be possible. However, a weaker condition of encouraging both parties to be honest was shown to be possible and incorporated into a third new protocol

(ECMH). This protocol while not being as efficient as the ECH and EMH protocols is as efficient in the number of messages required when compared with existing protocols in the literature. It also turns out that the ECMH protocol also makes automatic dispute resolution simpler by reducing the number of possible incidents where a dispute can be raised.

11.2. Criteria for Success

The result reported in this thesis was started with a set of aims labelled as criteria for success. These were enumerated in chapter 1. This section addresses each of the criteria to ascertain to what degree the research has succeeded.

1) Development of efficient optimistic fair exchange protocols

Three optimistic fair exchange protocols for exchanging payments and digital products between customers and merchants have been proposed. These protocols are the ECH protocol, the EMH protocol, and the ECMH protocol and are specified in chapters 5, 6, and 7, respectively. The ECH protocol and the EMH protocol applied the idea of encouraging a party to be honest and then enforcing the other party to be honest. This is an original contribution to fair exchange protocols. The ECH protocol encourages the merchant to be honest and then enforces the customer to be honest whereas the EMH protocol encourages the customer to be honest and then enforces the merchant to be honest. The same ideas were applied to try to enforce the two parties to be honest. However, due to the fact that only one party has to send their item first then this party is the one to be enforced to be honest. Therefore, enforcing two parties is not possible in the optimistic approach. As a result, instead of enforcing the two parties to be honest the ECMH protocol has been proposed which encourages the customer and the merchant to be honest.

The idea of enforcing a party to be honest which is applied in the ECH and the EMH protocols has resulted in reducing the number of messages needed in the exchange phase of the protocols. The number of messages needed to be exchanged between a customer and a merchant in the exchange phase of both the ECH and the

EMH protocols is only three messages. This number is the lowest number of messages when compared against the other relevant fair exchange protocols in the literature. On the other hand, the number of messages needed for the exchange phase of the ECMH protocol is four messages. This number of messages is the same as some protocols that appear in the literature. This shows how the idea of enforcing a party to be honest reduces the number of messages by one i.e. three messages only are required in the ECH and the EMH protocols.

With regard to number of messages in the dispute resolution phase, the lowest possible number of messages to be executed in the dispute resolution phase is three messages (this is in the case where there are two parties involved in the exchange and one TTP). This is because in the dispute resolution phase, a disputant needs to contact a TTP requesting a resolution which results in one message. In case the dispute has a resolution then the TTP will send the resolution to both parties which result in two messages. Therefore, the total number of messages is three. This is the number of messages needed in the dispute resolution phase of the ECH, EMH, and ECMH protocols.

Therefore, the number of messages exchanged during the proposed protocols (in both the exchange and dispute phases) is kept to minimum.

The ECH, EMH, and ECMH protocols were compared against each other. Then, the three protocols were compared with the relevant protocols from the literature. The comparisons were based on different criteria such as the number of messages in both the exchange and the dispute phases, number of modular exponentiations in both the exchange and the dispute phases. The results of the comparisons show that the ECH and EMH protocols have the lowest number of messages and also the lowest number of modular exponentiations in the exchange phase. The ECMH protocol has the lowest number of modular exponentiations compared with the related protocols from the literature. Details of these comparisons can be found in chapter 8.

This criterion has been met. A new result for the research was the notion of enforcing honesty on one of the parties in a transaction.

2) *Specification of the efficient optimistic fair exchange protocols*

The specifications of the new efficient optimistic fair exchange protocols (ECH, EMH, and ECMH) have been presented. These protocols have been specified using diagrams, informal descriptions and formal notations. The full specifications of the ECH protocol are discussed in section 5.2 of chapter 5, and section 10.2.1.1 of chapter 10. The full specifications of the EMH protocol are discussed in section 6.2 of chapter 6, and section 10.2.2.1 of chapter 10. The full specifications of the ECMH protocol are discussed in section 7.2 of chapter 7, and section 10.2.3.1 of chapter 10. The formal notation of all the three protocols is presented in section 4.3 of chapter 4.

3) *Built in automatic dispute resolution*

The ECH, EMH, and ECMH protocols are able to resolve any disputes automatically online with the help of the TTP. The dispute resolution phase of the ECH protocol that appears in section 5.2.4 (of chapter 5) shows the ability of the ECH protocol to resolve the disputes automatically online. Also, the dispute resolution phase of the EMH protocol that appears in section 6.2.4 (of chapter 6) shows the ability of the EMH protocol to resolve the disputes automatically online. Finally, the dispute resolution phase of the ECMH protocol that appears in section 7.2.4 (of chapter 7) shows the ability of the ECMH protocol to resolve the disputes automatically online. In all the three protocols the number of messages needed in the dispute resolution phase is only three messages.

By enforcing honesty the number of opportunities where a dispute can arise is reduced thus simplifying the dispute resolution phase.

4) *The protocols should ensure strong fairness for all parties*

It has been shown that the ECH, EMH, and ECMH protocols ensure strong fairness. That is, in all the three protocols either the customer receives the digital product and the merchant receives the payment or no one gets anything.

In the ECH protocol, if the customer sends the correct payment to the merchant (in message E-M2) but the merchant either sends (in message E-M3) incorrect decryption key or does not send the decryption key at all (i.e. it is not fair for the customer) then the customer can contact the TTP (in message DR-M1) to re-ensure the fairness. Hence, the strong fairness in the ECH protocol is ensured. The full details of the dispute resolution phase of the ECH protocol is described in section 5.2.4 of chapter 5.

In the EMH protocol, if the merchant sends the correct digital product to the customer (in message E-M2) but the customer either sends (in message E-M3) incorrect decryption key or does not send the decryption key at all (i.e. it is not fair for the merchant) then the merchant can contact the TTP (in message DR-M1) to re-ensure the fairness. Hence, the strong fairness in the EMH protocol is ensured. The full details of the dispute resolution phase of the EMH protocol is described in section 6.2.4 of chapter 6.

In the ECMH protocol, if the customer sends the correct decryption key to the merchant (in message E-M3) but the merchant either sends (in message E-M4) incorrect decryption key or does not send the decryption key at all (i.e. it is not fair for the customer) then the customer can contact the TTP (in message DR-M1) to re-ensure the fairness. Hence, the strong fairness in the ECMH protocol is ensured. The full details of the dispute resolution phase of the ECMH protocol is described in section 7.2.4 of chapter 7.

5) Analysis of the new protocols for completeness

All possible scenarios of executing the proposed protocols were studied. That is, all the possible scenarios of executing the ECH, the EMH, and the ECMH protocols were studied; see sections 5.3.3, 6.3.3 and 7.3.3. It has been shown that the strong fairness is ensured in all protocols in all scenarios under the assumptions made.

Detailed analyses were conducted on the proposed protocols to check whether or not each party is able to detect the dishonesty of the other party. The results of the

analyses show that each party is able to detect the dishonesty of the other party. The analysis for the ECH protocol can be found in section 5.3.1, and for the EMH protocol in section 6.3.1, and for the ECMH protocol in section 7.3.1.

All dispute possibilities and claims were studied and identified for the proposed protocols. The analyses show that the proposed protocols reduce the number of disputes by preventing them. It has been shown that there are resolutions in all cases were the claims are correct i.e. DR-M1 is correct. The complete analyses of all dispute claims of the ECH, EMH and ECMH protocols can be found in section 5.3.2, 6.3.2, and 7.3.2 respectively.

The SPIN model checker was used to formally verify the fairness and model the proposed protocols. It has been formally proven that the fairness is ensured for the three protocols (i.e. ECH, EMH, and ECMH protocols). Full details of the formal verification of the protocols can be found in chapter 9 and in the Appendix A.

6) A proof of concept implementation

All the three protocols were implemented in Java using Java-RMI. The implementation shows that in all the protocols each party was able to send messages and verify the correctness of the incoming messages. In addition, the implementation shows that the TTP was able to verify dispute requests and if they were valid then resolves them automatically online. All scenarios were tested and all of them worked as expected. The implementation details can be found in chapter 10.

11.3. Future Work

The research in this thesis was based on a number of assumptions enumerated in chapter 4. Different directions of future work have been identified, some to reduce or eliminate some of these assumptions. Some areas of future work are summarised as follows:

- Existing Fault-tolerant techniques from the literature should be combined with the proposed protocols in order to have fault-tolerant fair exchange protocols. These will ensure the fairness for all parties even in the event of a system crash and/or a communication failure
- Investigating the application of the idea of enforcing a party to be honest which is proposed in this thesis to the existing fair exchange protocols. This may reduce the number of messages in these protocols. Hence, more efficient fair exchange protocols may emerge
- The protocols presented in this thesis are for the exchange of a payment and a digital product. Extending this by applying these protocols to be used for certified e-mail, certified digital product delivery, contract signing
- Some customers prefer to be anonymous to merchants. The protocols presented in this thesis do not offer the anonymity for customer. This is because the merchant can identify the customer from the payment. A future work can study how to provide anonymity of customers in the proposed protocols
- Development of an adaptive system that includes all the three protocols (ECH, EMH, and ECMH) and switches between them to decide which one is more suitable to be used at any particular instance of a transaction. For example, if the two parties (the customer and the merchant) do not know each other then the adaptive system will chose the ECMH protocol. If one party tried to be dishonest then the adaptive system will make a note of this. Then, when the same parties want to exchange new items then the adaptive system will adapt itself to use the protocol that enforces the dishonest party to be honest
- Investigating the use of more than one TTP in the protocols
- The parties involved in the exchange phase of the proposed protocols are C and M. That is, they are two parties. Extending the ideas of the proposed protocols by having multi-party fair exchange protocols
- Investigating the use of different public-key cryptography algorithm. That is, using algorithm other than RSA
- Applying the ECH, EMH, and ECMH to be used for the exchange of a payment and a physical product

- Reducing the certificates included in the proposed protocols. That is, using the same D-Cert for selling the same product multiple times
- Including the time issues in the protocols and also in the dispute resolution

11.4. Summary

Three optimistic fair exchange protocols have been proposed in this thesis. The protocols are for the exchange of payment and digital product between a customer and a merchant. Two of the three protocols (namely, ECH and EMH protocols) use the idea of enforcing a party to be honest an idea that is originated in this thesis. The ECH protocol enforces the customer to be honest whereas the EMH enforces the merchant to be honest. The ECMH protocol, however, encourages the customer and the merchant to be honest. It has been shown that enforcing a party to be honest increases the efficiency of the fair exchange protocol leading to the fact that only three messages are required to exchange items between two parties. Additionally, the number of modular exponentiations that have to be calculated is reduced. Therefore, this answers the research question of this thesis which is: *"Is there a way of reducing the number of messages in the optimistic fair exchange protocols in order to have more efficient protocols?"*

The new protocols have been analysed and evaluated against each other and against the related protocols from the literature. It has been shown that all the three protocols ensure strong fairness under the assumptions made.

12. References

[AbGlHoPi02]: M. Abadi, N. Glew, B. Horne, and B. Pinkas: Certified email with a light online trusted third party: Design and implementation. In proceedings of the 11th International World Wide Web Conference (WWW'02), USA, ACM Press, pp. 387-395, 2002

[AlarajMunro07a]: A. Alaraj and M. Munro: An e-commerce Fair Exchange Protocol for exchanging Digital Products and Payments. In proceedings of the IEEE/ACM ICDIM'2007, France, pp. 248-253, 2007

[AlarajMunro07b]: A. Alaraj and M. Munro: An Efficient Fair Exchange Protocol that Enforces the Merchant to be Honest. In proceedings of the IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing 2007, CollaborateCom 2007, USA, pp. 196-202, 2007

[AlarajMunro08a]: A. Alaraj and M. Munro: An e-Commerce Fair Exchange Protocol that Enforces the Customer to be Honest. International Journal of Product Lifecycle Management, IJPLM, ISSN: 1743-5110, (To appear), 2008

[AlarajMunro08b]: A. Alaraj and M. Munro: An efficient e-Commerce Fair Exchange Protocol that encourages Customer and Merchant to be Honest. In proceedings of the 27th International Conference on Computer Safety, Reliability and Security, (SafeComp 2008), UK, Lecture Notes In Computer Science, LNCS, Vol. 5219, pp. 193-206, 2008

[AlfuraihSnow05]: S. Alfuraih and R. Snow: ODR and the E-commerce. In proceedings of Web Technologies, Applications, and Services conference 2005, Canada. pp.182-186, 2005

[AlHeMaQaRaTa98]: R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, and S. Tapcsiran: MOCHA: Modularity in Model Checking. In proceedings of the International Conference on Computer-Aided Verification, LNCS, Springer-Verlag, Vol. 1427, pp. 521-525, 1998

[AnHaLoSu05a]: B. Anderson, J. Hansen, P. Lowry, and S. Summers: Model Checking for E-Business Control and Assurance. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 35(3), pp. 445-450, 2005

[AnHaLoSu05b]: B. Anderson, J. Hansen, P. Lowry, and S. Summers: Model checking for design and assurance of e-business processes. Decision Support Systems (DSS), Vol. 39(3), pp. 333-344, 2005

[AnHaLoSu06a]: B. Anderson, J. Hansen, P. Lowry, and S. Summers: Standards and verification for fair-exchange and atomicity in e-commerce transactions. Information Sciences, Vol. 176(8), pp. 1045-1066, 2006

[AnHaLoSu06b]: B. Anderson, J. Hansen, P. Lowry, and S. Summers: The application of model checking for securing e-commerce transactions. Communications of the ACM, Vol 49(6), pp. 97-101, 2006

[AsShWa98]: N. Asokan, V. Shoup, M. Waidner: Asynchronous Protocols for Optimistic Fair Exchange. In proceedings of the IEEE Symposium on Research in Security and Privacy, pp. 86 - 99, 1998

[AsScWa97]: N. Asokan, M. Schunter, and M. Waidner: Optimistic Protocols for Fair Exchange. In proceedings of the 4th ACM conference on Computer and Communication Security, Switzerland, pp. 8-17, 1997

[Asokan98]: N. Asokan: Fairness in Electronic Commerce. PhD Thesis, University of Waterloo, Canada, 1998

[AsJaStWa97]: N. Asokan, P. Janson, M. Steiner, M. Waidner: State of the Art in Electronic Payment Systems. IEEE Computer Society Press, Vol. 30 (9), pp. 28- 35, 1997

[AtMeGo00]: G. Ateniese, B. de Medeiros, and M. Goodrich: TRICERT: Distributed certified email schemes. In proceedings of the ISOC 2000 Network and Distributed System Security Symposium, USA, 2000

[AhRyHa04]: N. Ahn, S. Ryu and I. Han: The impact of the online and offline features on the user acceptance of Internet shopping malls. Electronic Commerce Research and Applications, Vol. 3(4), pp. 405-420, 2004

[BaDeMa98]: F. Bao, R. Deng, W. Mao: Efficient and Practical Fair Exchange Protocols with Off-Line TTP. In proceedings of the IEEE Symposium on Research in Security and Privacy, IEEE Computer Society, USA, pp. 77-85, 1998

[BBC08]: http://news.bbc.co.uk/1/hi/business/6508983.stm, accessed on 20/05/08

[BeGoMiRi90]: M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest: A Fair Protocol for Signing Contracts. IEEE Transactions on Information Theory, Vol. 36(1), pp. 40-46, 1990.

[Blum83]: M. Blum: How to Exchange (Secret) Keys. ACM Transactions on Computer Systems, Vol. 1(2), pp. 175-193, 1983

[BuPf90]: H. Burk and A. Pfitzmann: Value Exchange Systems Enabling Security and Unobservability. Computers & Security, Vol. 9(9), pp. 715-721, 1990

[CaSiTy95]: L. Camp, M. Sirbu, and J. Tygar: Token and Notational Money in Electronic Commerce. In proceedings of the 1st conference on USENIX Workshop on Electronic Commerce, pp. 1-12, 1995

[ClGrPe99]: E. Clarke, O. Grumberg, and D. Peled: Model checking. MIT Press, 1999

[CoTySi95]: B. Cox, J. Tygar, M. Sirbu: NetBill security and transaction protocol. In proceedings of the 1st USENIX Workshop on Electronic Commerce, USENIX Association, pp. 77-88, 1995

[CoThYi03]: B. Corbitt, T. Thanasankit and H. Yi: Trust and e-commerce: a study of consumer perceptions. Electronic Commerce Research and Applications, Vol. 2(3), pp. 203-215, 2003

[CookLuo03]: D. Cook and W. Luo: The Role of Third-Party Seals in Building Trust Online. e-Service Journal, Vol. 2(3), pp. 71-84, 2003

[DeGoLaWa96]: R. Deng, L. Gong, A. Lazar, and W. Wang: Practical protocols for certified electronic mail. Journal of Network and Systems Management, Practical protocols for certified electronic mail Vol. 4(3), Springer, pp. 279-297, 1996

[DeChPh07]: S. Devane, M. Chatterjee, and D. Phatak: Secure E-Commerce Protocol for Purchase of e-Goods - Using Smart Card. In proceedings of the 3d IEEE International Symposium on Information Assurance and Security, IAS 2007. UK, pp. 9-14, 2007

[EBay]: eBay: http://www.ebay.com/ accessed on 12/06/08

[E-Bay06a]: http://pages.ebay.com/help/tp/problems-dispute-resolution.html accessed on 19/10/06

[E-Bay06b] http://pages.ebay.com/help/tp/problems-ov.html accessed on 20/10/06

[EvGoLe85]: S. Even, O. Goldreich, and A. Lempel: A Randomized Protocol for Signing Contracts. Communications of the ACM, Vol. 28(6), pp. 637-647, 1985

[EzhShr05]: P. Ezhilchelvan and S. Shrivastava: A Family of Trusted Third Party Based Fair-Exchange Protocols. IEEE Transactions on Dependable and Secure Computing, Vol. 2(4), pp. 273-286, 2005

[FaTuCo98]: J. Fanjul, J. Tuya, J. Corrales: Formal Verification and Simulation of the NetBill protocol Using SPIN. In proceedings of the 4th International SPIN Workshop, France, 1998

[FDR99]: Formal Systems (Europe) Ltd. Oxford, UK: Failures Divergence Refinement – FDR2 User Manual, version 2.64, 1999

[FordBaum97]: W. Ford and M. Baum: Secure Electronic Commerce. Upper Saddle River, New Jersey: Prentice Hall. ISBN 0134763424, 1997

[GalKov06]: B. Galitsky and B. Kovalerchuk: Analyzing Attitude in Customer Emails: A Tool for Complaint Assessment. In proceedings of the SIGIR 2006 Workshop on Directions in Computational Analysis of Stylistics in Text Retrieval, USA, pp. 17-36, 2006

[Ghosh98]: A. Ghosh: E-COMMERCE SECURITY. Canada: John Wiley. ISBN 0471192236, 1998

[Guadamuz03]: Andrés Guadamuz: PayPal and eBay- The legal implications of the C2C electronic commerce model. In proceedings of the 18th BILETA Conference: Controlling Information in the Online Environment, UK, 2003

[HeTyWiWo96]: N. Heintze, J. Tygar, J. Wing, H. Wong: Model checking Electronic Commerce Protocols. In proceedings of the 2nd USENIX workshop on Electronic Commerce, Vol. 2, 1996

[Hörnle02]: J. Hörnle: Online Dispute Resolution in Business to Consumer E-commerce Transactions. Journal of Information, Law and Technology, Vol. 2002(2), 2002

[Holzmann97]: G. Holzmann: The model checker SPIN. IEEE Transactions on Software Engineering, Vol. 23(5), pp. 279-295, 1997

[Holzmann04]: G. Holzmann: The spin model checker: primer and reference manual. Addison-Wesley, 2004

[Hsieh01]: C. Hsieh: E-commerce payment systems: critical issues and management strategies. Human Systems Management, Vol. 20(2), pp 131-138, 2001

[Jakobsson95]: M. Jakobsson: Ripping Coins for a Fair Exchange. Lecture Notes in Computer Science 921. pp. 220-230, 1995

[JaMrTsYu99]: M. Jakobsson, D. M'Raihi, Y. Tsiounis, and M. Yung: Electronic Payments: Where Do We Go from Here? Lecture Notes in Computer Science 1740, pp. 43-63, 1999

[Jones01]: R. Jones: The PayPal phenomenon: Lessons from the Leading Edge of Online Payments. CommerceNet Security and Internet Payments Research, pp. 6-11, 2001

[Katsh02]: E. Katsh: Online Dispute Resolution: The Next Phase. Lex Electronica, Vol. 7(2), 2002, available at http://www.lex-electronica.org/articles/v7-2/katsh.htm accessed on 19/10/06

[Ketchpel95]: S. Ketchpel: Transaction Protection for Information Buyers and Sellers. In proceedings of the Dartmouth Institute for Advanced Graduate Studies .95: Electronic Publishing and the Information Superhighway, USA, 1995

[KimKim05]: H. Kim and D. Kim: A Study of Online Transaction Self-Efficacy, Consumer Trust, and Uncertainty Reduction in Electronic Commerce Transaction. In proceedings of the 38th Hawaii International Conference on System Sciences, 2005, IEEE, 2005, available at http://csdl.computer.org/comp/proceedings/hicss/2005/2268/07/22680170c.pdf accessed on 16/11/2006

[KonarMazumdar06]: D. Konar and C. Mazumdar: An Improved E-Commerce Protocol for Fair Exchange. LNCS Vol. 4317, Springer-Verlag, pp. 305-313, 2006

[KrMaZh02]: S. Kremer, O. Markowitch, and J. Zhou: An Intensive Survey of Fair nonrepudiation Protocols. Computer Communications, Vol. 25(17), pp. 1606-1621, 2002

[KrMa01]: S. Kremer and O. Markowitch: Selective Receipt in Certified E-mail. In proceedings of the Progress in Cryptology, INDOCRYPT 01, Vol. 2247, pp. 136-149. Springer-Verlag, 2001

[LiNiJa00]: P. Liu, P. Ning, and S. Jajodia: Avoiding Loss of Fairness Owing to Process Crashes in Fair Data Exchange Protocols. In proceedings of the International Conference on Dependable Systems and Network, pp. 631-40, 2000

[MaGoKr03]: O. Markowitch, D. Gollmann, and S. Kremer: On Fairness in Exchange Protocols. LNCS Vol. 2587, Springer-Verlag, pp. 451-465, 2003

[Micali03]: S. Micali: Simple and fast optimistic protocols for fair electronic exchange. Annual ACM Symposium on Principles of Distributed Computing, PODC 2003, pp. 12-19, ACM Press, 2003

[Nenadic05]: A. Nenadic: A Security Solution for Fair Exchange and Non-repudiation in E-Commerce. PhD Thesis, University of Manchester, UK, 2005

[NeZhBa04]: A. Nenadic, N. Zhang, S. Barton: Fair Certified Email Delivery. 2004 ACM Symposium on Applied Computing, SAC '04, Cyprus, 2004

[NeZhBa04a]: A. Nenadic, N. Zhang and S. Barton: A Security Protocol for Certified E-Goods Delivery. In proceedings of the IEEE International Conference on Information Technology, Coding and Computing – ITCC '04, USA, IEEE Computer Society, pp. 22-28, 2004

[NeZhSh05]:A. Nenadic, N. Zhang and Q. Shi: RSA-based Verifiable and Recoverable Encryption of Signatures and Its Application in Certified E-mail Delivery. Journal of Computer Security, Vol 13(5), IOS Press, pp. 757-777, 2005

[NeZhChGo05]: A. Nenadic, N. Zhang, B. Cheetham and C. Goble: RSA-based Certified Delivery of E-Goods Using Verifiable and Recoverable Signature Encryption. Journal of Universal Computer Science, Vol. 11(1), Springer-Verlag, pp. 175-192, 2005

[NeZh03]: A. Nenadic and N. Zhang: Non-Repudiation and Fairness in Electronic Data Exchange. In proceedings of the 5th International Conference on Enterprise Information Systems ICEIS 2003, France, pp. 55-62, 2003

[PaVoGa03]: H. Pagnia, H. Vogt and F. G¨artner: Fair Exchange. The Computer Journal, Vol. 46(1), 2003

[PaChSi03]: J. Park, E. Chong, and H. Siegel: Constructing fair exchange protocols for ecommerce via distributed computation of RSA signatures. In proceedings of the 22nd annual symposium on Principles of distributed computing. USA, pp. 172-181, 2003

[PayPal08]: www.paypal.com , accessed on 13/05/08

[RayRay00a]: I. Ray and I. Ray: An Optimistic Fair Exchange E-Commerce Protocol with Automated Dispute Resolution. In proceedings of 1st Electronic Commerce and Web Technologies Conference EC-Web 2000, Lecture Notes in Computer Science, Springer-Verlag, 1875, pp. 84-93, 2000

[RayRay02]: I. Ray and I. Ray: Fair Exchange in E-commerce. ACM SIGecom Exchange, Vol. 3(2), pp. 9-17, 2002

[RayGeisterfer04]: I. Ray and M. Geisterfer: Towards a Privacy Preserving e-Commerce Protocol. LNCS Vol. 3182, Springer-Verlag, pp. 154-163, 2004

[RaRaNa00]: I. Ray, I. Ray, and N. Narasimhamurthy: A Fair-Exchange E-Commerce Protocol with Automated Dispute Resolution. In proceedings of the 14th Annual IFIP WG 11.3 Working Conference on Database Security, The Netherlands, pp. 27-38, 2000

[RaRaNa05]: I. Ray, I. Ray, and N. Narasimhamurthy: An Anonymous and Failure Resilient Fair-Exchange E-Commerce Protocol. Decision Support Systems, Vol. 39(3), pp. 267-292, 2005

[RayRay00b]: I. Ray and I. Ray: Failure analysis of an e-commerce protocol using model checking. In proceedings of the 2nd International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems,WECWIS 2000, pp.176-183, 2000

[RayRay01]: I. Ray and I. Ray: An Anonymous Fair-Exchange E-Commerce Protocol. In proceedings of the 1st International Workshop on Internet Computing and E-Commerce, pp. 1790-1797, 2001

[RayZhang08]: I. Ray and H. Zhang: Experiences in developing a fair-exchange e-commerce protocol using common off-the-shelf components. Journal of Electronic Commerce Research and Application, Vol. 7(2), pp. 247-259, 2008

[RiShAd78]: R. Rivest, A. Shamir, L. Adleman: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, Vol. 21(2), pp. 120-126, 1978

[RuUnHa03]: C. Ruppel, L. Underwood-Queen and S. Harrington: e-Commerce: The Roles of Trust, Security, and Type of e-commerce Involvement. e-Service Journal Vol. 2(2), pp. 25-45, 2003

[Saha05] G. Saha: Transient Software Fault Tolerance Using Single-Version Algorithm. ACM Ubiquity, Vol. 6(28), 2005, available at http://www.acm.org/ubiquity/ accessed on 27/06/08

[Schunter00]: M. Schunter: Optimistic fair exchange. PhD Thesis, University of Saarland, Germany, 2000

[Schneier96]: B. Schneier: Applied cryptography: protocols, algorithms and source code in C. Wiley, 1996

[SET]: Mastercard & VISA: SET Secure Electronic Transaction Specification: Protocol Definition, 1997

[ShmMit99]: V. Shmatikov and J. Mitchell: Analysis of a Fair Exchange Protocol. In proceedings of the 1999 FLoC Workshop on Formal Methods and Security Protocols, Italy, 1999

[ShZhMe03]: Q. Shi, N. Zhang, and M. Merabti: Signature-based approach to fair document exchange. IEE Proceedings - Communications, Vol. 150(1), pp. 21-27, 2003

[SMV08]: SMV System, available at http://www.cs.cmu.edu/~modelcheck/smv.html accessed on 31/05/08

[SoKoTa02]: W. Song, W. Kou, and C. Tan: An Investigation on Multiple e-Payment and Micro-Payment. In proceedings of the International Conference on Electronic Commerce, pp. 216-223, 2002

[Spin08]: SPIN official website, http://spinroot.com accessed on 30/05/08

[SSL]: A. Freier, P. Karlton, and P. Kocher: SSL: Secure Socket Layer 3.0 Specification. Netscape Communications Corporation, Transport Layer Security Working Group, 1996, available at http://wp.netscape.com/eng/ssl3/ accessed on 12/06/08

[Stat07]: National Statistics, UK, 2007, available at http://www.statistics.gov.uk/pdfdir/ecom1107.pdf, accessed on 27/05/08

[Stallings99]: W. Stallings: Cryptography and network security: principles and practice. Prentice Hall, 1999

[Srinivasan04]: S. Srinivasan: Role of trust in e-business success. Information Management & Computer Security, Vol. 12(1), pp. 66-72, 2004

[SquareT06]: http://www.squaretrade.com/cnt/jsp/index.jsp, accessed on 19/10/06

[TsiakisStheph05]: T. Tsiakis and G. Sthephanides: The concept of security and trust in electronic payments. Computers & Security, Vol. 24(1), pp. 10-15, 2005

[TuKiLeVi04]: E. Turban, D. King, J. Lee and D. Viehland: Electronic Commerce: A Managerial Perspective. Upper Saddle River, New Jersey: Prentice Hall, ISBN 0131230158, 2004

[Tygar96]: J. Tygar: Atomicity in Electronic Commerce. In proceedings of the 15th annual ACM symposium on Principles of distributed computing, pp. 8-26, 1996

[VeriSoft08]: VeriSoft, available at http://cm.bell-labs.com/who/god/verisoft/ accessed on 31/05/08

[Violino02]: B. Violino: Building B2B Trust. Computerworld, Vol. 36(25), pp. 32, 2002

[WaHiBaWh00]: W. Wang, Z. Hidvegi, A. Bailey, and A. Whinston: E-process design and assurance using model checking. Computer, Vol. 33(10), pp. 48-53, 2000

[Wang05]:G. Wang: An AbuseFree Fair Contract Signing Protocol Based on the RSA Signature. In proceedings of the 14th International conference on World Wide Web, WWW-2005, pp. 412-421, 2005

[Whiteley00]: D. Whiteley: e-Commerce: Strategy, Technologies and Applications. London: McGraw-Hill, ISBN 0077095529, 2000

[Wright02]: D. Wright: Comparative Evaluation of Electronic Payment Systems. INFOR Journal, Vol. 40(1), 2002

[ZhMaMa06]: Q. Zhang K. Markantonakis K. Mayes: A practical fair exchange e-payment protocol for anonymous purchase and physical delivery. In proceedings of the 4th ACS/IEEE International Conference on Computer Systems and Applications, AICCSA-06, UAE, pp. 851-858, 2006

[ZhShMe04]: N. Zhang, Q. Shi, and M. Merabti: A unified approach to a fair document exchange system. Journal of Systems and Software, Vol. 72(1), pp. 83–96, 2004

[ZhShNeMeAs05]: N. Zhang, Q. Shi, A. Nenadic , M. Merabti, and R. Askwith: Efficient fair digital-signature exchange based on misbehaviour penalisation. IEE Proceedings - Communications, Vol. 152(3), pp. 257-261, ISSN 1350-2425, 2005

[ZhShMeAs06]: N. Zhang, Q. Shi, M. Merabti, and R. Askwith: Practical and Efficient Fair Document Exchange over Networks. The Journal of Network and Computer Applications, Vol 29(1), Elsevier Science Publisher, pp. 46-61, ISSN 10848045, 2006

[ZhGo96a]: J. Zhou and D. Gollmann: Observations on Non-Repudiation. In proceedings of Advances in Cryptology - EUROCRYPT 96. LNCS 1163, Springer-Verlag, pp. 133-144, 1996

[ZhGo96b]: J. Zhou and D. Gollmann: A Fair Non-repudiation Protocol. In proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society, USA, pp. 55-61, 1996

[ZhSh96]: N. Zhang and Q. Shi: Achieving Non-Repudiation of Receipt. The Computer Journal, Vol. 39(10), pp. 844-853, 1996

Appendix A

This appendix includes the Promela code used in SPIN to formally verify the protocols. The codes for the ECH, EMH, and ECMH protocols are presented separately in the following sections.

Modelling the ECH protocol

```
#define TRUE 1
#define FALSE 0

/*
 ECH Protocol messages
*/
mtype = {EM1,EM2,EM3,DRM1,DRM2,DRM3};

bool quitMerchant, quitCustomer;
bool receiveCorrectPayment, receiveCorrectProduct;

/* Merchant     Sends messages:EM1, EM3
                Receives messages: EM2, DRM2
*/
proctype Merchant(chan chCM, chMTTP)
{
  bool sendEM1, sendEM3, waitForTTP;

  sendEM1 = FALSE;
  sendEM3 = FALSE;
  waitForTTP = FALSE;
  quitMerchant = TRUE;
```

```
do
:: (sendEM1 == FALSE) ->
        chCM!EM1;
        sendEM1 = TRUE;
        quitMerchant = FALSE;
        printf("Merchant: message EM1 sent \n");

:: (sendEM1 = = TRUE) && (sendEM3 = = FALSE) && (quitMerchant =  =
FALSE) ->

 if
  :: chCM?EM2 ->  /*  Merchant receives the payment from Customer */
      printf("Merchant: message EM2 received \n");
      quitMerchant = FALSE;

       /* Now all possibilities after receiving EM2 by M will be studied */
      if

        :: TRUE -> /* payment is correct and M is honest */
           receiveCorrectPayment = TRUE;
           chCM!EM3;
           sendEM3 = TRUE;
           quitMerchant = TRUE;
           waitForTTP = TRUE;
           printf("Merchant: message EM3 sent \n");

                :: TRUE -> /* payment is correct,M quits the protocol,M is dishonest */
                   receiveCorrectPayment = TRUE;
                   quitMerchant = TRUE;
                   waitForTTP = TRUE;
                   printf("Merchant: The payment is ok but M quits the protocol before
sending message EM3 to C ; M is dishonest \n");
```

```
                :: TRUE -> /* payment is incorrect; M quits the protocol */
                    quitMerchant = TRUE;
                    printf("Merchant: The payment is incorrect and hence M will not
send EM3 to C BUT M will wait for TTP in case C contacted TTP  \n");
                    waitForTTP = TRUE;

            fi

 /* Nothing is received after sending EM1; it seems that C does not
 want to exchange, BUT M will wait for any messages from TTP
in case C contacted them */

    :: timeout ->
        quitMerchant = TRUE;
        waitForTTP = TRUE;
        fi
    :: (waitForTTP = = TRUE) ->

        if
            :: chMTTP?DRM2 ->  /*  M receives the payment from TTP because C
contacts TTP before C sends EM2 or because EM3 either incorrect or has not been
sent to C by M */

                    receiveCorrectPayment = TRUE;
                    quitMerchant = TRUE;
                    printf("Merchant: M received payment from TTP; The protocol is fair
by the help of TTP \n");
                    break;

                :: timeout ->
                    quitMerchant = TRUE;
                    break;
```

```
        fi


    od;

}

/* Customer   Sends messages:EM2, DRM1
                  Receives messages: EM1, EM3, DRM3

*/
proctype Customer(chan chCM, chCTTP)
{

    bool sendEM2, sendDRM1;
    sendEM2 = FALSE;
    sendDRM1 = FALSE;
    quitCustomer = TRUE;

   do
    :: (sendEM2 = = FALSE) && (sendDRM1= = FALSE) ->
            chCM?EM1;
            printf("Customer: message EM1 received \n");
            quitCustomer = FALSE;

    /* Now all possibilities after receiving EM1 by C will be studied */

    if

     :: TRUE ->  /* EM1 is correct; and C want to complete the exchange */
          chCM!EM2;
          sendEM2 = TRUE;
```

```
                printf("Customer: message EM2 sent \n");

            :: TRUE -> /* EM1 is correct, but C contacts TTP to get the decryption
                          key before C sends the payment to M */

                chCTTP!DRM1;
                sendDRM1= TRUE;
                printf("Customer: message DRM1 sent \n");

            :: TRUE -> /* EM1 is correct, but C quits the protocol i.e. C does not
                          want to complete the exchange */

                quitCustomer = TRUE;
                printf("Customer: Verification of EM1 is ok but Customer quits the
protocol before sending message EM1 \n");
                break;

            :: TRUE -> /* EM1 is incorrect; C quits the protocol */
                quitCustomer = TRUE;
                printf("Customer: EM1 is incorrect and hence Customer quits the
                        protocol before sending message EM1 \n");
                break;

        fi

    :: (sendEM2 = = TRUE) && (sendDRM1= = FALSE) && (quitCustomer = =
    FALSE) ->

        if
            :: chCM?EM3 -> /*  EM3 is received  */
                    printf("Customer: message EM3 received \n");

            /* Now all possibilities after receiving EM3 by C will be studied */
```

```
if

:: TRUE -> /* The decryption key is correct and C finishes
              the protocol as it was fair */
              receiveCorrectProduct = TRUE;
              quitCustomer = TRUE;
              printf("Customer: C decrypts the digital product correctly; Fair
protocol \n");
              break;

           :: TRUE -> /* The decryption key is correct , but C contacts
                         TTP to get the decryption key again */

              receiveCorrectProduct = TRUE;
              chCTTP!DRM1;
              sendDRM1= TRUE;
              printf("Customer: C received correct decryption key from M
but C sends to TTP to get the decryption key again \n");

              :: TRUE -> /* The decryption key is incorrect; C contatc TTP
                            to get the decryption key */

                 chCTTP!DRM1;
                 sendDRM1= TRUE;
                 printf("Customer: decryption key is incorrect and hence
                         Customer sends DRM1 to TTP \n");

fi

:: timeout -> /* C has not received EM3; So, C will contact TTP as C
sent EM2 to M but has not received EM3 */
              chCTTP!DRM1;
```

```
                    sendDRM1= TRUE;
                    printf("Customer: C has not received EM3; So, C will contact
TTP as C sent EM2 to M but has not received EM3\n");

            fi
        :: (sendDRM1= = TRUE) && (quitCustomer = = FALSE) ->

                    chCTTP?DRM3;
                    receiveCorrectProduct = TRUE;
                    quitCustomer = TRUE;
                    printf("Customer: C received correct decryption key from TTP; The
protocol is fair by the help of TTP \n");
                    break;
        od;
}

/* TTP    Sends messages:DRM2, DRM3
          Receives messages: DRM1

*/
proctype TTP(chan chMTTP, chCTTP)
{

      do
        :: chCTTP?DRM1 -> /*  when TTP receives DRM1 from C then resolve  the
disputes fairly */
            chMTTP!DRM2;
            chCTTP!DRM3;
            break;
      od;
}
init
{
```

```
chan chCM     = [0] of {mtype};
chan chCTTP  = [0] of {mtype};
chan chMTTP  = [0] of {mtype};

run Merchant(chCM, chMTTP);
run Customer(chCM, chCTTP);
run TTP(chMTTP, chCTTP);

}
```

Modelling the EMH protocol:

```
#define TRUE 1
#define FALSE 0

/*
 EMH Protocol messages
*/

mtype = {EM1,EM2,EM3,DRM1,DRM2,DRM3};

bool quitMerchant, quitCustomer;
bool receiveCorrectPayment, receiveCorrectProduct;

/* Customer     Sends messages:EM1, EM3
                Receives messages: EM2, DRM2

*/
proctype Customer(chan chCM, chCTTP)
{
```

```
bool sendEM1, sendEM3, waitForTTP;

sendEM1 = FALSE;
sendEM3 = FALSE;
waitForTTP = FALSE;
quitCustomer = TRUE;

do
:: (sendEM1 = = FALSE) ->
        chCM!EM1;
        sendEM1 = TRUE;
        quitCustomer = FALSE;
        printf("Customer: message EM1 sent \n");

:: (sendEM1 = = TRUE) && (sendEM3 = = FALSE) && (quitCustomer = =
FALSE) ->
if
 :: chCM?EM2 ->  /*  Customer receives the payment from Merchant */
     printf("Customer: message EM2 received \n");
     quitCustomer = FALSE;

     /* Now all possibilities after receiving EM2 by C will be studied */
     if

       :: TRUE -> /* The digital product is correct; and C want to complete the
exchange fairly; C is honest */
                  receiveCorrectProduct = TRUE;
                  chCM!EM3;
                  sendEM3 = TRUE;
                  quitCustomer = TRUE;
                  waitForTTP =TRUE;
                  printf("Customer: message EM3 sent \n");
```

```
:: TRUE -> /* The digital product is correct, but C quits the
              protocol before sending the decryption key i.e. C is
              dishonest */
    receiveCorrectProduct = TRUE;
    quitCustomer = TRUE;
    waitForTTP =TRUE;
    printf("Customer: The digital product is ok but C quits the protocol
before sending message EM3 to M ; C is dishonest \n");

:: TRUE -> /* The digital product is incorrect; C quits the
              protocol BUT will wait for TTP in case M asked
              for resolution */

    quitCustomer = TRUE;
    printf("Customer: The digital product is incorrect and hence C will
not send EM3 to M, also C will wait for TTP in case M asked for resolution \n");
    waitForTTP =TRUE;

  fi


  /* Nothing is received after sending EM1; it seems that M does not want to
  exchange, BUT C will wait for TTP in case M has contacted them */
 :: timeout ->
    quitCustomer = TRUE;
    waitForTTP = TRUE;
 fi

:: (waitForTTP == TRUE) ->

   if
```

```
:: chCTTP?DRM2 -> /*C receives the digital product from TTP
                    because M contacts TTP before M sends EM2
                    or because EM3 either incorrect or has not been
                    sent to M by C */

        receiveCorrectProduct = TRUE;
        quitCustomer = TRUE;
                printf("Customer: C received digital product from TTP; The protocol is
fair by the help of TTP \n");
                break;

            :: timeout ->
                quitCustomer = TRUE;
                 break;

        fi
    od;
}

/* Merchant   Sends messages:EM2, DRM1
                Receives messages: EM1, EM3, DRM3

*/
proctype Merchant(chan chCM, chMTTP)
{
    bool sendEM2, sendDRM1;
    sendEM2 = FALSE;
    sendDRM1 = FALSE;
    quitMerchant = TRUE;

    do
     :: (sendEM2 = = FALSE) && (sendDRM1= = FALSE) ->
            chCM?EM1;
            printf("Merchant: message EM1 received \n");
```

```
                quitMerchant = FALSE;

        /* Now all possibilities after receiving EM1 by M will be studied */
                if

                :: TRUE -> /* EM1 is correct; M want to complete the exchange
                                */
                        chCM!EM2;
                        sendEM2 = TRUE;
                        printf("Merchant: message EM2 sent \n");

                :: TRUE -> /* EM1 is correct, but M contacts TTP to get
                        the decryption key before M sends digital product to C */

                        chMTTP!DRM1;
                        sendDRM1= TRUE;
                        printf("Merchant: message DRM1 sent \n");

                        :: TRUE -> /* EM1 is correct, but M quits the protocol i.e. M
                                        does not want to complete the exchange */

                                quitMerchant = TRUE;
                                printf("Merchant: Verification of EM1 is ok but Merchant quits
the protocol before sending message EM1 \n");
                                break;

                        :: TRUE -> /* EM1 is incorrect; M quits the protocol */

                                quitMerchant = TRUE;
                                printf("Merchant: EM1 is incorrect and hence Merchant quits
the protocol before sending message EM1 \n");
                                break;

                fi
```

```
:: (sendEM2 = = TRUE) && (sendDRM1= = FALSE) && (quitMerchant
    = =  FALSE) ->

     if
        :: chCM?EM3 -> /*  EM3 is received  */
                printf("Merchant: message EM3 received \n");
                /* Now all possibilities after receiving EM3 by M will be studied */
                if

                :: TRUE -> /* The decryption key is correct and M finishes the
protocol as it was fair */
                        receiveCorrectPayment = TRUE;
                        quitMerchant = TRUE;
                        printf("Merchant: M decrypts the payment correctly;
Fair protocol \n");
                        break;

                :: TRUE -> /*  The decryption key is correct , but M
                            contacts TTP to get the decryption key again */

                        receiveCorrectPayment = TRUE;
                        chMTTP!DRM1;
                        sendDRM1= TRUE;
                        printf("Merchant: M received correct decryption key
from C but M sends to TTP to get the decryption key again \n");

        :: TRUE -> /* The decryption key is incorrect; M contacts
                    TTP to get the decryption key */

                chMTTP!DRM1;
                sendDRM1= TRUE;
```

```
                printf("Merchant: decryption key is incorrect and hence
Merchant sends DRM1 to TTP \n");

                fi

            :: timeout -> /* M has not received EM3; So, M will contact TTP as M
sent EM2 to C but has not received EM3 */

                chMTTP!DRM1;
                sendDRM1= TRUE;
                printf("Merchant: M has not received EM3; So, M will contact
TTP as M sent EM2 to C but has not received EM3 \n");

            fi

        :: (sendDRM1= = TRUE) && (quitMerchant = = FALSE) ->

            chMTTP?DRM3;
            receiveCorrectPayment = TRUE;
            quitMerchant = TRUE;
            printf("Merchant: M received correct decryption key from TTP; The
protocol is fair by the help of TTP \n");
            break;

    od;

}

/* TTP    Sends messages:DRM2, DRM3
          Receives messages: DRM1

*/
proctype TTP(chan chMTTP, chCTTP)
{
```

```
    do
      :: chMTTP?DRM1 ->
            chCTTP!DRM2;
            chMTTP!DRM3;
            break;
    od;

}

init
{

        chan chCM    = [0] of {mtype};
        chan chCTTP  = [0] of {mtype};
        chan chMTTP  = [0] of {mtype};

        run Customer(chCM, chCTTP);
        run Merchant(chCM, chMTTP);
        run TTP(chMTTP, chCTTP);

}
```

Modelling the ECMH protocol:

```
#define TRUE 1
#define FALSE 0

/*
 ECMH Protocol messages
*/
```

```
mtype = {EM1,EM2,EM3,EM4,DRM1,DRM2,DRM3};

bool quitMerchant, quitCustomer;
bool receiveCorrectPayment, receiveCorrectProduct;

/* Customer     Sends messages:EM1, EM3,DRM1
                Receives messages: EM2, EM4, DRM3
*/
proctype Customer(chan chCM, chCTTP)
{

    bool sendEM1, sendEM3, sendDRM1;

    sendEM1 = FALSE;
    sendEM3 = FALSE;
    sendDRM1= FALSE;
    quitCustomer = TRUE;

    do
    :: (sendEM1 == FALSE) ->
        chCM!EM1;
        sendEM1 = TRUE;
        quitCustomer = FALSE;
        printf("Customer: message EM1 sent \n");


    :: (sendEM1 = = TRUE) && (sendEM3 = = FALSE) && (sendDRM1 = =
    FALSE) && (quitCustomer = = FALSE) ->

    if
    :: chCM?EM2 ->  /*  Customer receives the encrypted payment */
                    from Merchant */

        printf("Customer: message EM2 received \n");
```

```
quitCustomer = FALSE;

/* Now all possibilities after receiving EM2 by C will be studied */
if

:: TRUE -> /* EM2 is correct; and C want to complete the exchange by
sending EM3 to be able to receive EM4*/

chCM!EM3;
sendEM3 = TRUE;
quitCustomer = FALSE;
printf("Customer: message EM3 sent \n");

:: TRUE -> /* C contacts TTP before sending EM3 to M; this
may happen either EM2 is correct or incorrect */

chCTTP!DRM1;
sendDRM1 = TRUE;
quitCustomer = FALSE;
printf("Customer: message DRM1 sent to TTP \n");

:: TRUE -> /* The verification of EM2 is ok, but C quits the
protocol before sending EM3; So no party is hurt i.e.
no one loose anything */

quitCustomer = TRUE;
printf("Customer: The verification of EM2 is ok, but C quits the
protocol before sending EM3; So no party is hurt i.e. no one loose anything");
break;

:: TRUE -> /* EM2 in incorrect i.e. the verification of EM2 is
negative; C quits the protocol */
```

```
                    quitCustomer = TRUE;
                    printf("Customer: EM2 in incorrect i.e. the verification of EM2
is negative; and hence C quits the protocol before sending message EM3 to M \n");
                    break;

                fi
            :: timeout -> /* Nothing is received after sending EM1; it seems that M
                          does not want to exchange */

                quitCustomer = TRUE;
                printf("Customer: Nothing is received after sending EM1; it seems that
M does not want to exchange \n");

                break;

            fi
        :: (sendEM3 = = TRUE) && (sendDRM1 = = FALSE) && (quitCustomer
            = = FALSE) -> /* C sent EM3 and waiting for EM4 */

            if
            :: chCM?EM4 ->  /*  C receives the decryption key from M */
                printf("Customer: message EM4 received \n");
                quitCustomer = FALSE;

                /* Now all possibilities after receiving EM4 by C will be studied */
                if
                :: TRUE ->  /* EM4 is correct; so, C finishes the protocol*/
                        receiveCorrectProduct = TRUE;
                        quitCustomer = TRUE;
                        printf("Customer: message EM4 received; fair protocol \n");
                        break;
```

```
:: TRUE ->  /* EM4 is correct; but C contacts TTP */
        receiveCorrectProduct = TRUE;
        chCTTP!DRM1;
        sendDRM1 = TRUE;
        quitCustomer = FALSE;
        printf("Customer: EM4 is correct; but C contacts TTP \n");

:: TRUE -> /* EM4 is incorrect; So, C contacts TTP for resolution
                */

        chCTTP!DRM1;
        sendDRM1 = TRUE;
        quitCustomer = FALSE;
       printf("Customer: EM4 is incorrect, So, C contacts TTP for
            resolution \n");

    fi

:: timeout ->   /* Nothing is received after sending EM3; So C must
                contact TTP to resolve dispute as M seems dishonest */

        chCTTP!DRM1;
        sendDRM1 = TRUE;
        quitCustomer = FALSE;
        printf("Customer: Nothing is received after sending EM3; So C must
contact TTP to resolve dispute as M seems dishonest \n");

    fi
:: (sendDRM1 = = TRUE) && (quitCustomer = = FALSE) -> /* C sent
                DRM1 to TTP and waiting for DRM3 */
    if
```

```
                :: chCTTP?DRM3 ->  /*C receives the decryption key from TTP
                                        as a resolution for the fairness */

                        receiveCorrectProduct = TRUE;
                        quitCustomer = TRUE;
                        printf("Customer: C received decryption key from TTP; The
protocol is fair by the help of TTP \n");
                        break;

                fi
        od;

}

/* Merchant   Sends messages:EM2, EM4
                Receives messages: EM1, EM3, DRM2

*/
proctype Merchant(chan chCM, chMTTP)
{

    bool sendEM2, sendEM4, waitForTTP;

    sendEM2 = FALSE;
    sendEM4 = FALSE;
    waitForTTP = FALSE;
    quitMerchant = TRUE;

    do
     :: (sendEM2 == FALSE) ->
        chCM?EM1;
        printf("Merchant: message EM1 received \n");
        quitMerchant = FALSE;
```

```
        /* Now all possibilities after receiving EM1 by M will be studied */
      if

      :: TRUE -> /* EM1 is correct; and M want to send EM2*/
            chCM!EM2;
            sendEM2 = TRUE;
            printf("Merchant: message EM2 sent \n");

         :: TRUE -> /* EM1 is correct, but M does not want to exchange */
               quitMerchant = TRUE;
               printf("Merchant: Verification of EM1 is ok but Merchant quits the
protocol before sending message EM2 \n");
               break;

         :: TRUE -> /* EM1 is incorrect; M quits the protocol */
               quitMerchant = TRUE;
               printf("Merchant: EM1 is incorrect and hence Merchant quits the
                     protocol before sending message EM1 \n");
            break;

      fi

   :: (sendEM2 = = TRUE) && (sendEM4 = = FALSE) && (quitMerchant
      = = FALSE) ->

      if
      :: chCM?EM3 -> /*  EM3 is received  */
         printf("Merchant: message EM3 received \n");

         /* Now all possibilities after receiving EM3 by M will be studied*/
         if

         :: TRUE ->  /* The decryption key is correct and M sends EM4
```

```
                to finishes the protocol fairly */

            receiveCorrectPayment = TRUE;
            chCM!EM4;
            sendEM4 = TRUE;
            waitForTTP = TRUE;
            quitMerchant = TRUE;
                printf("Merchant: M decrypts the payment correctly; so, M
                    sends EM4 to finish the protocol fairly \n");

        :: TRUE -> /* The decryption key is correct; but M quits the
                protocol; M is dishonest */

            receiveCorrectPayment = TRUE;
            waitForTTP = TRUE
            quitMerchant = TRUE;
                printf("Merchant: The decryption key is correct; but M quits the
protocol; M is dishonest \n");

            :: TRUE -> /* The decryption key is incorrect; M quits the
                    protocol so no party looses anything */

                quitMerchant = TRUE;
                waitForTTP = TRUE;
                printf("Merchant: The decryption key is incorrect; M quits the
protocol; so no party looses anything \n");
    fi

:: timeout -> /* M has not received EM3; So, M will wait because C
                might contact TTP before sending EM3 */

            quitMerchant = TRUE;
```

```
                    waitForTTP = TRUE;

            fi

        :: ( waitForTTP == TRUE) ->

            if
                :: chMTTP?DRM2 -> /*  M received correct decryption key for
                                      the payment from TTP; The protocol is fair
                                      by the help of TTP */

                    receiveCorrectPayment = TRUE;
                    quitMerchant = TRUE;
                    printf("Merchant: M received correct decryption key from TTP;
The protocol is fair by the help of TTP \n");
                    break;

                :: timeout ->
                        quitMerchant = TRUE;
                        break;
            fi

        od;
}


/* TTP    Sends messages:DRM2, DRM3
          Receives messages: DRM1

*/
proctype TTP(chan chMTTP, chCTTP)
{

        do
```

```
        :: chCTTP?DRM1 ->
                chMTTP!DRM2;
                chCTTP!DRM3;
                break;
    od;

}

init
{

        chan chCM    = [0] of {mtype};
        chan chCTTP  = [0] of {mtype};
        chan chMTTP  = [0] of {mtype};

        run Customer(chCM, chCTTP);
        run Merchant(chCM, chMTTP);
        run TTP(chMTTP, chCTTP);

}
```