DESIGN AND DEVELOPMENT OF CRYPTOGRAPHIC FAIR EXCHANGE PROTOCOLS

by

ÇAĞIL CAN ÖNİZ

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
November 2004

DESIGN AND DEVELOPMENT OF
CRYPTOGRAPHIC FAIR EXCHANGE PROTOCOLS

APPROVED BY:

Asst. Prof. Erkay Savaş (Thesis Advisor)	………………………….
Asst. Prof. Albert Levi (Thesis Co-Advisor)	………………………….
Asst. Prof. Özgür Gürbüz	………………………….
Asst. Prof. Selim Balcısoy	………………………….
Asst. Prof. Cem Güneri	………………………….

DATE OF APPROVAL: ………………………….

ABSTRACT

In this thesis, the problem of fair exchange on specific cases is addressed. The main idea of fair exchange is as follows: Two entities that do not trust each other want to exchange some arbitrary data over a communication network. Since they do not trust each other, neither party wants to transmit their own data before receiving the other entity's data. Even though either party could prove an unjust situation after termination of the protocol, if they are in different countries, solving disputes may require time and money due to the bureaucracy of international laws.

In this thesis, a special application of fair exchange, a fair *e-commerce* protocol for large e-goods is designed and implemented. The proposed protocol provides a method for fair exchange of e-money to e-products, and a method for verifying the contents of the exchanged items. The presented protocol is efficient such that when none of the parties tries to cheat, only three messages are sufficient. In case of disputes, three more messages are needed. Furthermore, in most of the previously proposed protocols in the literature, e-goods are transferred multiple times among some entities. This situation is too costly when e-goods are large. In the presented protocol, e-goods are transferred only once. Another important property of the protocol is the anonymity of the customer; no information about the customers shopping habits can be gathered through the protocol. The implementation results show that the protocol is efficient and secure and that small number of cryptographic operations is sufficient.

In addition to the fair e-commerce protocol, another special application of fair exchange, a *fair multimedia exchange protocol* using a different method is designed and implemented. This protocol is designed due to different requirements of different applications. In the fair multimedia exchange protocol, two entities want to exchange some multimedia files such as video or audio files. This protocol requires lower security and has a different a lower degree of fairness as compared to the fair e-commerce protocol. Fair multimedia exchange protocol uses a *baby-step* approach in which the

probability of protocol completion is gradually increased over several cycles. In baby-step approach protocols, entities exchange pieces of the items, which they want to barter. At protocol completion, the complete items are formed by using the pieces exchanged.

ÖZET

Bu tez, adil takas problemini bazı özel durumlar için ele almaktadır. Genel olarak adil takas problemi, birbirlerine güvenmeyen iki tarafın rastgele seçtikleri verileri takas etme sorunu olarak tanımlanabilir. Bu iki taraf birbirlerine güvenmedikleri için, almayı bekledikleri verileri elde etmeden kendi verilerini yollamak istemezler. Bu iki tarafın farklı ülkelerde bulunması ve taraflardan birinin haksızlığa uğraması halinde, uluslar arası hukuk bürokrasisi yüzünden bu anlaşmazlığı çözmek para ve zaman gerektirebilir.

Bu tezde adil takas probleminin özel bir uygulaması olan, büyük boyutlardaki elektronik mallar için adil e-ticaret protokolü tasarlanmış ve uygulanmıştır. Önerilen bu protokol elektronik para karşılığında elektronik malları adil bir şekilde takas eder. Aynı zamanda takas edilen elektronik malların kalitesi ve içeriğinin kontrolünü de yapar. Sunulan bu protokol verimli bir şekilde çalışmaktadır. Öyleki, taraflardan hiçbiri hile yapmayı denemezse, sadece üç mesaj yeterlidir. Taraflardan biri hile yapmayı denerse, anlaşmazlığı çözmek için üç mesaja daha ihtiyaç olacaktır. Literatürde daha önce yapılan başka çalışmalarda önerilen protokollerde, elektronik mallar taraflar arasında birçok kez transfer edilmiştir. Bu durum elektronik malların büyük boyutlarda olması halinde yüksek maliyetlere sebep olmaktadır. Bu tezde önerilen protokolde elektronik mallar sadece bir kez transfer edilmektedir. Bu protokolun başka önemli bir özelliği ise müşterilerin kimliklerinin anonim bırakılmasıdır. Öyleki, protokol akışı sırasında müşterilerin alışveriş alışkanlıkları hakkında hiçbir bilgi toplanamamaktadır. Uygulama sonuçları, adil e-ticaret protokolünün verimli, güvenilir ve az sayıda kriptografik operasyona ihtiyaç olduğunu göstermektedir.

Bu tezde sunulan e-ticaret protokolü dışında yine adil takas probleminin özel bir uygulaması olan, ancak farklı bir yöntemle tasarlanmış ve uygulanmış bir adil çoğulortam takas protokolü sunulmaktadır. Bu protokolü tasarımının ardındaki amaç farklı tipdeki uygulamaların farklı yöntem gereksinimleridir. Adil çoğulortam takas protokolünde iki birey birbiri ile bazı çoğulortam dosyalarını (ör: görüntü veya ses dosyaları) takas etmek isterler. Bu protokol adil e-ticaret protokolüne göre daha az

güvenlik gerektirmekte ve daha düşük derecede adalet sağlamaktadır. Adil çoğulortam takas protokolünde *bebek-adımları* yöntemi kullanılmıştır. Bu yöntemde protokolünün başarılı bir biçimde tamamlanma olasılığı her adımda artmaktadır. Taraflar değişmek istedikleri elektronik malları parçalara ayırıp birbirlerine sırayla bu parçaları yollarlar. Protokol sona erdiğinde elektronik mallar elde edilen parçalar birleştirerek oluşturulur.

LIST OF SYMBOLS

$CERT_{TP^{i}}$: ith Certificate (in terms of price, description, and contents) of a product signed by the trusted third party

H : Hash function (i.e. SHA-1)

E_X (data) : Encryption of data with key X

E-good : An e-product or electronic item such as, database or multimedia file

Price : Price of the e-good

Description : A string describing contents of the product

KUX : Public key of identity X

KRX : Private key of identity X

SIG_X(Data) : Data signed by identity X; equivalent to $E_{KR^{X}}$ (H(Data))

TP : Trusted third party

M : Merchant

C : Customer or Client

|| : Concatenation Operation

PID : Product Identifier

Token : Electronic money; it contains credit card information such as credit card brand, credit card number, amount of money to be transferred, destination account number and PID . Confidential information of the token is only readable by the bank

A➔B:X : A sends data X to B

To my family…

ACKNOWLEDGEMENTS

I would like to thank my advisor, Asst. Prof. Erkay Savaş for his continuous support and his non-wavering trust in me during the past two years. His knowledge and experience gave me motivation in every step and helped me to finish my thesis. I am grateful for his guidance and advice concerning every detail of this thesis.

I would like to express my special thanks to my co-advisor Asst. Prof. Albert Levi, a great mentor, who inspired me with his enthusiasm for the topic and my research throughout this study.

Finally, I want to express my special thanks to my family for their never-ending motivation and positive attitude to my academic career. Knowing that they were behind me at every decision inspired me to start my graduate work and kept me going during the hard days.

TABLE OF CONTENTS

LIST OF SYMBOLS iii

TABLE OF CONTENTS vii

1 INTRODUCTION 1

2 BACKGROUND INFORMATION 5

2.1 Information Security 5

2.1.1 Security Services 6

2.1.1.1 Confidentiality 6

2.1.1.2 Integrity 6

2.1.1.3 Authentication 7

2.1.1.4 Non-Repudiation 7

2.1.2 Security Mechanisms 7

2.2 Overview of Cryptography 8

2.2.1 Symmetric Cryptography 9

2.2.2 Asymmetric Cryptography 11

2.2.2.1 An RSA Example 14

2.2.3 Hash Functions 15

2.2.4 Hash Based Message Authentication Codes (HMAC) 16

2.3 Cryptographic Solutions to Security Problems 16

2.3.1 Confidentiality with Symmetric Encryption 16

2.3.2 Authentication and Integrity with Digital Signatures 17

2.3.3 Digital Envelopes 18

3 RELATED WORK 20

3.1 Exchange Protocols 20

3.1.1 Protocol Using Trusted Third Parties 20

3.1.1.1 Online Trusted Third Party Protocols 21

3.1.1.2 Optimistic Protocols 21

3.1.2 Baby-Step Protocols 22

3.2 Simultaneous Contract Signing Protocols 24

3.3 Digital Certified Mail Protocols 24

3.4 E-Commerce Protocols 25

3.4.1 Desired Properties of E-commerce Protocols 27

4 DESIGN OF FAIR E-COMMERCE PROTOCOL 30

4.1 Assumptions 30

4.2 The Payment Token 32

4.3 Chain Keys 35

4.4 The Offline Certification Process 35

4.5 The Protocol Description 37

5 DESIGN OF FAIR MULTIMEDIA EXCHANGE PROTOCOL 41

5.1 Assumptions 42

5.2 Oblivious Transfer Protocol 43

5.3 The Protocol Description 45

6 IMPLEMENTATION ISSUES 48

6.1 Fair E-Commerce Protocol 48

6.1.1 Cryptographic Methods 48

6.1.1.1 Random Symmetric Key Generator 48

6.1.1.2 Initialization Vector Generator 48

6.1.1.3 Rijndael Encrypt/Decrypt 49

6.1.1.4 RSA Sign/Verify 49

6.1.1.5 Hash 49

6.1.1.6 HMAC 50

6.1.1.7 Get Key from Chain Key 50

6.1.2 Requirements 50

6.1.3 Deployment of the System 52

6.1.4 Performance Issues 53

6.2 Fair Multimedia Exchange Protocol 57

6.2.1 Cryptographic Methods 57

6.2.1.1 Random Symmetric Key Generator 58

6.2.1.2 Initialization Vector Generator 58

6.2.1.3 Rijndael Encrypt/Decrypt 58

6.2.1.4 RSA Sign/Verify 58

6.2.1.5 Create Public/Private Keys 59

6.2.2 Requirements 59

6.2.3 Deployment of the System 59

6.2.4 Performance Issues 60

7 CONCLUSIONS AND FUTURE WORK 66

8 APPENDIX: FORMS 68

8.1 Fair E-Commerce Protcol 68

8.1.1 Trusted Third Party Forms 68

8.1.2 Merchant Forms 70

8.1.3 Client Forms 73

8.2 Fair Multimedia Exchange Protocol 75

8.3 Multimedia Exchange Program 76

8.4 Multimedia E-Commerce Protocol 77

REFERENCES 78

LIST OF FIGURES

Figure 1. Basic Cryptographic Communication Model 9

Figure 2. Symmetric Cryptography Communication Model 10

Figure 3. Usage of Asymmetric Keys 12

Figure 4. Asymmetric Encryption/Decryption Model 13

Figure 5. Authentication with Asymmetric Cryptosystems 13

Figure 6. Digital Signature Model Based on RSA 17

Figure 7. Digital Enveloping 19

Figure 8. An Optimistic Protocol for Fair Certified E-mail 26

Figure 9. SET Purchase Request Message 34

Figure 10. Chain Key Production 35

Figure 11. Product Certificate Format 36

Figure 12. Fair Optimistic E-Commerce Protocol Description 38

Figure 13. Oblivious Transfer Protocol 44

Figure 14. File Division 45

Figure 15. Creation and Grouping of n symmetric keys 45

Figure 16. Encryption of n Pieces of Files with n Symmetric Keys 46

Figure 17. Transmission of Keys Using Oblivious Transfer 47

Figure 18. File Division Time...61

Figure 19. Exchange (No Security) Time with Varying Buffer Size.............................62

Figure 20. Fair Exchange Time (File Size = 600MB, Buffer Size = 4MB)..................63

Figure 21. Multimedia E-Commerce Protocol..64

Figure 22. Trusted Third Party Form, Product Certification 68

Figure 23. Trusted Third Party Form, Settings Page 1 69

Figure 24. Trusted Third Party Form, Settings Page 2 69

Figure 25. Trusted Third Party Form, Settings Page 3 70

Figure 26. Merchant Form, New Certificate Registration 70

Figure 27. Merchant Form, Settings Page 1..71

Figure 28. Merchant Form, Settings Page 2..71

Figure 29. Merchant Form, Settings Page 3..72

Figure 30. Merchant Form, Settings Page 4..72

Figure 31. Client Form, Product Search and Purchase 73

Figure 32. Client Form, Settings Page 1 73

Figure 33. Client Form, Settings Page 2 74

Figure 34. Fair Multimedia Exchange Form...75

Figure 35. Quality Control Confirm Dialog Box 75

Figure 36. Exchange Complete Declaration Dialog Box 75

Figure 37. Multimedia Exchange Form 76

Figure 38. Exchange Complete Declaration Dialog Box 76

Figure 39. Multimedia E-Commerce Merchant Form 77

Figure 40. Multimedia E-Commerce Client Form 77

LIST OF TABLES

Table 1. Asymmetric Cryptosystems Functionality 11

Table 2. Comparison of Several E-Commerce Protocols 29

Table 3. Merchant Table 51

Table 4. TP (Third Party) Table 52

Table 5. Client Cryptographic Operation Count 54

Table 6. Merchant Cryptographic Operation Count 54

Table 7. Third Party Cryptographic Operation Count 55

Table 8. Third Party Certification Cryptographic Operation Count 55

Table 9. Third Party Certification Times 55

Table 10. Comparison of Several Protocols for Cryptographic Operation Count (No Disputes) 56

Table 11. Comparison of Several Protocols for Cryptographic Operation Count (Disputes) 56

Table 12. Fair Multimedia Exchange Protocol Cryptographic Operation Count 60

DESIGN AND DEVELOPMENT OF CRYPTOGRAPHIC FAIR EXCHANGE PROTOCOLS

by

ÇAĞIL CAN ÖNİZ

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
November 2004

DESIGN AND DEVELOPMENT OF
CRYPTOGRAPHIC FAIR EXCHANGE PROTOCOLS

APPROVED BY:

Asst. Prof. Erkay Savaş (Thesis Advisor)	………………………….
Asst. Prof. Albert Levi (Thesis Co-Advisor)	………………………….
Asst. Prof. Özgür Gürbüz	………………………….
Asst. Prof. Selim Balcısoy	………………………….
Asst. Prof. Cem Güneri	………………………….

DATE OF APPROVAL: ………………………….

ABSTRACT

In this thesis, the problem of fair exchange on specific cases is addressed. The main idea of fair exchange is as follows: Two entities that do not trust each other want to exchange some arbitrary data over a communication network. Since they do not trust each other, neither party wants to transmit their own data before receiving the other entity's data. Even though either party could prove an unjust situation after termination of the protocol, if they are in different countries, solving disputes may require time and money due to the bureaucracy of international laws.

In this thesis, a special application of fair exchange, a fair *e-commerce* protocol for large e-goods is designed and implemented. The proposed protocol provides a method for fair exchange of e-money to e-products, and a method for verifying the contents of the exchanged items. The presented protocol is efficient such that when none of the parties tries to cheat, only three messages are sufficient. In case of disputes, three more messages are needed. Furthermore, in most of the previously proposed protocols in the literature, e-goods are transferred multiple times among some entities. This situation is too costly when e-goods are large. In the presented protocol, e-goods are transferred only once. Another important property of the protocol is the anonymity of the customer; no information about the customers shopping habits can be gathered through the protocol. The implementation results show that the protocol is efficient and secure and that small number of cryptographic operations is sufficient.

In addition to the fair e-commerce protocol, another special application of fair exchange, a *fair multimedia exchange protocol* using a different method is designed and implemented. This protocol is designed due to different requirements of different applications. In the fair multimedia exchange protocol, two entities want to exchange some multimedia files such as video or audio files. This protocol requires lower security and has a different a lower degree of fairness as compared to the fair e-commerce protocol. Fair multimedia exchange protocol uses a *baby-step* approach in which the

probability of protocol completion is gradually increased over several cycles. In baby-step approach protocols, entities exchange pieces of the items, which they want to barter. At protocol completion, the complete items are formed by using the pieces exchanged.

ÖZET

Bu tez, adil takas problemini bazı özel durumlar için ele almaktadır. Genel olarak adil takas problemi, birbirlerine güvenmeyen iki tarafın rastgele seçtikleri verileri takas etme sorunu olarak tanımlanabilir. Bu iki taraf birbirlerine güvenmedikleri için, almayı bekledikleri verileri elde etmeden kendi verilerini yollamak istemezler. Bu iki tarafın farklı ülkelerde bulunması ve taraflardan birinin haksızlığa uğraması halinde, uluslar arası hukuk bürokrasisi yüzünden bu anlaşmazlığı çözmek para ve zaman gerektirebilir.

Bu tezde adil takas probleminin özel bir uygulaması olan, büyük boyutlardaki elektronik mallar için adil e-ticaret protokolü tasarlanmış ve uygulanmıştır. Önerilen bu protokol elektronik para karşılığında elektronik malları adil bir şekilde takas eder. Aynı zamanda takas edilen elektronik malların kalitesi ve içeriğinin kontrolünü de yapar. Sunulan bu protokol verimli bir şekilde çalışmaktadır. Öyleki, taraflardan hiçbiri hile yapmayı denemezse, sadece üç mesaj yeterlidir. Taraflardan biri hile yapmayı denerse, anlaşmazlığı çözmek için üç mesaja daha ihtiyaç olacaktır. Literatürde daha önce yapılan başka çalışmalarda önerilen protokollerde, elektronik mallar taraflar arasında birçok kez transfer edilmiştir. Bu durum elektronik malların büyük boyutlarda olması halinde yüksek maliyetlere sebep olmaktadır. Bu tezde önerilen protokolde elektronik mallar sadece bir kez transfer edilmektedir. Bu protokolun başka önemli bir özelliği ise müşterilerin kimliklerinin anonim bırakılmasıdır. Öyleki, protokol akışı sırasında müşterilerin alışveriş alışkanlıkları hakkında hiçbir bilgi toplanamamaktadır. Uygulama sonuçları, adil e-ticaret protokolünün verimli, güvenilir ve az sayıda kriptografik operasyona ihtiyaç olduğunu göstermektedir.

Bu tezde sunulan e-ticaret protokolü dışında yine adil takas probleminin özel bir uygulaması olan, ancak farklı bir yöntemle tasarlanmış ve uygulanmış bir adil çoğulortam takas protokolü sunulmaktadır. Bu protokolü tasarımının ardındaki amaç farklı tipdeki uygulamaların farklı yöntem gereksinimleridir. Adil çoğulortam takas protokolünde iki birey birbiri ile bazı çoğulortam dosyalarını (ör: görüntü veya ses dosyaları) takas etmek isterler. Bu protokol adil e-ticaret protokolüne göre daha az

güvenlik gerektirmekte ve daha düşük derecede adalet sağlamaktadır. Adil çoğulortam takas protokolünde *bebek-adımları* yöntemi kullanılmıştır. Bu yöntemde protokolünün başarılı bir biçimde tamamlanma olasılığı her adımda artmaktadır. Taraflar değişmek istedikleri elektronik malları parçalara ayırıp birbirlerine sırayla bu parçaları yollarlar. Protokol sona erdiğinde elektronik mallar elde edilen parçalar birleştirerek oluşturulur.

LIST OF SYMBOLS

$CERT_{TP^{i}}$: ith Certificate (in terms of price, description, and contents) of a product signed by the trusted third party

H : Hash function (i.e. SHA-1)

E_X (data) : Encryption of data with key X

E-good : An e-product or electronic item such as, database or multimedia file

Price : Price of the e-good

Description : A string describing contents of the product

KUX : Public key of identity X

KRX : Private key of identity X

SIG_X(Data) : Data signed by identity X; equivalent to $E_{KR^{X}}$ (H(Data))

TP : Trusted third party

M : Merchant

C : Customer or Client

|| : Concatenation Operation

PID : Product Identifier

Token : Electronic money; it contains credit card information such as credit card brand, credit card number, amount of money to be transferred, destination account number and PID . Confidential information of the token is only readable by the bank

A$\rightarrow$B:X : A sends data X to B

To my family…

ACKNOWLEDGEMENTS

I would like to thank my advisor, Asst. Prof. Erkay Savaş for his continuous support and his non-wavering trust in me during the past two years. His knowledge and experience gave me motivation in every step and helped me to finish my thesis. I am grateful for his guidance and advice concerning every detail of this thesis.

I would like to express my special thanks to my co-advisor Asst. Prof. Albert Levi, a great mentor, who inspired me with his enthusiasm for the topic and my research throughout this study.

Finally, I want to express my special thanks to my family for their never-ending motivation and positive attitude to my academic career. Knowing that they were behind me at every decision inspired me to start my graduate work and kept me going during the hard days.

TABLE OF CONTENTS

LIST OF SYMBOLS iii

TABLE OF CONTENTS vii

1 INTRODUCTION 1

2 BACKGROUND INFORMATION 5

2.1 Information Security 5

2.1.1 Security Services 6

2.1.1.1 Confidentiality 6

2.1.1.2 Integrity 6

2.1.1.3 Authentication 7

2.1.1.4 Non-Repudiation 7

2.1.2 Security Mechanisms 7

2.2 Overview of Cryptography 8

2.2.1 Symmetric Cryptography 9

2.2.2 Asymmetric Cryptography 11

2.2.2.1 An RSA Example 14

2.2.3 Hash Functions 15

2.2.4 Hash Based Message Authentication Codes (HMAC) 16

2.3 Cryptographic Solutions to Security Problems 16

2.3.1 Confidentiality with Symmetric Encryption 16

2.3.2 Authentication and Integrity with Digital Signatures 17

2.3.3 Digital Envelopes 18

3 RELATED WORK 20

3.1 Exchange Protocols 20

3.1.1 Protocol Using Trusted Third Parties 20

3.1.1.1 Online Trusted Third Party Protocols 21

3.1.1.2 Optimistic Protocols 21

3.1.2 Baby-Step Protocols 22

3.2 Simultaneous Contract Signing Protocols 24

3.3 Digital Certified Mail Protocols 24

3.4 E-Commerce Protocols 25

3.4.1 Desired Properties of E-commerce Protocols 27

4 DESIGN OF FAIR E-COMMERCE PROTOCOL 30

4.1 Assumptions 30

4.2 The Payment Token 32

4.3 Chain Keys 35

4.4 The Offline Certification Process 35

4.5 The Protocol Description 37

5 DESIGN OF FAIR MULTIMEDIA EXCHANGE PROTOCOL 41

5.1 Assumptions 42

5.2 Oblivious Transfer Protocol 43

5.3 The Protocol Description 45

6 IMPLEMENTATION ISSUES 48

6.1 Fair E-Commerce Protocol 48

6.1.1 Cryptographic Methods 48

6.1.1.1 Random Symmetric Key Generator 48

6.1.1.2 Initialization Vector Generator 48

6.1.1.3 Rijndael Encrypt/Decrypt 49

6.1.1.4 RSA Sign/Verify 49

6.1.1.5 Hash 49

6.1.1.6 HMAC 50

6.1.1.7 Get Key from Chain Key 50

6.1.2 Requirements 50

6.1.3 Deployment of the System 52

6.1.4 Performance Issues 53

6.2 Fair Multimedia Exchange Protocol 57

6.2.1 Cryptographic Methods 57

6.2.1.1 Random Symmetric Key Generator 58

6.2.1.2 Initialization Vector Generator 58

6.2.1.3 Rijndael Encrypt/Decrypt 58

6.2.1.4 RSA Sign/Verify 58

6.2.1.5 Create Public/Private Keys 59

6.2.2 Requirements 59

6.2.3 Deployment of the System 59

6.2.4 Performance Issues 60

7 CONCLUSIONS AND FUTURE WORK 66

8 APPENDIX: FORMS 68

8.1 Fair E-Commerce Protcol 68

8.1.1 Trusted Third Party Forms 68

8.1.2 Merchant Forms 70

8.1.3 Client Forms 73

8.2 Fair Multimedia Exchange Protocol 75

8.3 Multimedia Exchange Program 76

8.4 Multimedia E-Commerce Protocol 77

REFERENCES 78

LIST OF FIGURES

Figure 1. Basic Cryptographic Communication Model 9

Figure 2. Symmetric Cryptography Communication Model 10

Figure 3. Usage of Asymmetric Keys 12

Figure 4. Asymmetric Encryption/Decryption Model 13

Figure 5. Authentication with Asymmetric Cryptosystems 13

Figure 6. Digital Signature Model Based on RSA 17

Figure 7. Digital Enveloping 19

Figure 8. An Optimistic Protocol for Fair Certified E-mail 26

Figure 9. SET Purchase Request Message 34

Figure 10. Chain Key Production 35

Figure 11. Product Certificate Format 36

Figure 12. Fair Optimistic E-Commerce Protocol Description 38

Figure 13. Oblivious Transfer Protocol 44

Figure 14. File Division 45

Figure 15. Creation and Grouping of n symmetric keys 45

Figure 16. Encryption of n Pieces of Files with n Symmetric Keys 46

Figure 17. Transmission of Keys Using Oblivious Transfer 47

Figure 18. File Division Time...61

Figure 19. Exchange (No Security) Time with Varying Buffer Size.............................62

Figure 20. Fair Exchange Time (File Size = 600MB, Buffer Size = 4MB)..................63

Figure 21. Multimedia E-Commerce Protocol..64

Figure 22. Trusted Third Party Form, Product Certification 68

Figure 23. Trusted Third Party Form, Settings Page 1 69

Figure 24. Trusted Third Party Form, Settings Page 2 69

Figure 25. Trusted Third Party Form, Settings Page 3 70

Figure 26. Merchant Form, New Certificate Registration 70

Figure 27. Merchant Form, Settings Page 1..71

Figure 28. Merchant Form, Settings Page 2..71

Figure 29. Merchant Form, Settings Page 3..72

Figure 30. Merchant Form, Settings Page 4..72

Figure 31. Client Form, Product Search and Purchase 73

Figure 32. Client Form, Settings Page 1 73

Figure 33. Client Form, Settings Page 2 74

Figure 34. Fair Multimedia Exchange Form...75

Figure 35. Quality Control Confirm Dialog Box 75

Figure 36. Exchange Complete Declaration Dialog Box 75

Figure 37. Multimedia Exchange Form 76

Figure 38. Exchange Complete Declaration Dialog Box 76

Figure 39. Multimedia E-Commerce Merchant Form 77

Figure 40. Multimedia E-Commerce Client Form 77

LIST OF TABLES

Table 1. Asymmetric Cryptosystems Functionality 11

Table 2. Comparison of Several E-Commerce Protocols 29

Table 3. Merchant Table 51

Table 4. TP (Third Party) Table 52

Table 5. Client Cryptographic Operation Count 54

Table 6. Merchant Cryptographic Operation Count 54

Table 7. Third Party Cryptographic Operation Count 55

Table 8. Third Party Certification Cryptographic Operation Count 55

Table 9. Third Party Certification Times 55

Table 10. Comparison of Several Protocols for Cryptographic Operation Count (No Disputes) 56

Table 11. Comparison of Several Protocols for Cryptographic Operation Count (Disputes) 56

Table 12. Fair Multimedia Exchange Protocol Cryptographic Operation Count 60

1 INTRODUCTION

The bottom line of an exchange protocol is to barter data fairly between two entities. The contents of the exchanged items may differ but the main idea is the same. Exchange protocols get special names according to the contents of exchanged items. Below, some special cases for exchange of some specific items are depicted:

- In a contract signing protocol, digital signatures that bind two entities to the terms stated on a contract are exchanged.

- In a certified mail protocol, an e-mail message is exchanged for a receipt. The receipt proves that the receiver of the e-mail has obtained the e-mail.

- In an e-commerce protocol, payment is exchanged for an electronic good (e-good).

The four important requirements of exchange protocols are fairness, quality control, client anonymity and the number of e-good transfer.

- **Fairness:** An exchange protocol is considered as fair if the protocol has only two possible outcomes: either both entities obtain the items they expect or neither entity obtains any items.

- **Quality Control:** An exchange protocol must prove that a claim of a definition of an item truly defines the contents of that item.

- **Client Anonymity:** Entities that perform exchange may decide to remain anonymous. Anonymity provides that identity of an entity not to be revealed during the transaction. For instance, a customer would not want a merchant to discover his/her pattern of shopping habits after performing a transaction.

- **Number of E-Good Transfer:** There must not be any assumptions on the size of exchanged items. Therefore, the exchanged items may be very large and transferring these large items multiple times could be costly. For this reason, large items should be transferred only once per protocol run.

Exchange protocols can be examined in two categories: *online third party protocols* [3, 5, 9, 14 and 15] (see section 3.1.1.1 for details) and *baby-step protocols* [12 and 13] (see section 3.1.2 for details). In online third party protocols, the exchange is achieved via a trusted third party. Each party submits their own item to the trusted third party that forwards the items to the appropriate recipient entity. In baby-step approach protocols, items are divided into small partial items. Two entities achieve the exchange by swapping multiple partial items one by one. In other words, one of the entities sends one of his/her partial item to the other entity and waits for the other side to send a partial item. This act of swapping partial items continues until the items are completely exchanged, hence the solution is obtained with "baby steps".

Both approaches have some drawbacks [9]: Online third party protocols require that the third party always be at service; therefore, we need bandwidth to handle large amounts of traffic. This large amount of traffic routed to the third party creates a bottleneck in the network. Online third party protocols have a *client/server* architecture, which causes a single point of failure and requires costly precautions for continuous service. On the other hand, baby-step protocols may have a large overhead. In some cases, they provide a lower degree of fairness.

In order to overcome the problems stated in the online third party protocols, another approach, called *optimistic protocols* [4, 7, 8, 9 and 19], has been proposed. In optimistic protocols, two entities want to perform an exchange. The entity that starts the protocol is the initiator and the other one is the receiver. First, the initiator takes a risk by sending its own item to the receiver. Second, the receiver either sends its own item or tries to cheat by not sending anything. If the receiver behaves honest, by sending its own item, the protocol is complete since both entities obtained the items they expected. If the initiator does not receive the item it expected, the initiator contacts the trusted third party for dispute resolution. The trusted third party resolves this dispute by sending the correct item to the initiator. Optimistic protocols use a trusted third party only in case of disputes. Optimistic protocols discourage cheating, since cheating is of no purpose. Therefore, attempts of cheating and consequently participation of a trusted third party would be rare. This property of keeping the trusted third party out of normal execution reduces network traffic at the third party.

In this thesis, an optimistic fair e-commerce protocol for large e-goods is presented. The proposed protocol provides a method for not only fair exchange of e-money to e-goods, but also the verification of the contents of the exchanged items for quality control purposes. The proposed protocol is efficient; when none of the parties tries to cheat, only three messages are sufficient. To resolve disputes, if any, only three more messages are needed. In most of the previously proposed protocols in the literature, e-goods are transferred multiple times, which is too costly when the e-goods are large. In the presented protocol, e-goods are transferred only once. Another important property of the protocol is the anonymity of the customer; no information about the customer's shopping habits can be gathered through the protocol.

In addition to the optimistic fair e-commerce protocol, a fair multimedia exchange protocol using a baby-step approach is designed in order to show that in some situations, baby-step protocols may be more convenient than an optimistic fair exchange protocol. In fair multimedia exchange protocol, two entities want to exchange some multimedia files such as video or audio files. The fair exchange protocol has client-*server* architecture. However, fair multimedia exchange protocol works in *peer-to-peer* fashion.

The structure of the thesis is as follows:

In Section 2, background information including the fundamentals of cryptography, security problems and solutions are discussed. In Section 3, related work on exchange protocols is presented. In section 4, the proposed optimistic fair e-commerce protocol is discussed. In section 5, the proposed multimedia exchange protocol is described. In section 6, implementation details of the two proposed protocols are depicted. In section 7, conclusions and future works are presented. Section 8 is appendix in which forms used in both of the implemented systems may be found.

2 BACKGROUND INFORMATION

In today's world, information security is an important and mandatory concept. There are many examples that show security can prevent costly problems or even shame. The most typical of these examples include disclosure of private data, malicious scripts, viruses, trojans, unauthorized access of resources, bogus messages such as TCP hijacking and repudiation on an agreement. Electronic systems may perform critical operations that require information security. Below, some background information related to the proposed thesis is presented.

2.1 Information Security

Information security is about taking actions in order to prevent the corruption of resources, to detect who corrupted, how a resource is corrupted, and to take action to recover corrupted resources. For instance, during an e-commerce transaction, one must prevent a credit card number from being revealed, detect an unauthorized transaction, and resolve any kind of dispute related to an unauthorized transaction.

Two important aspects of information security are *security services* and *security mechanisms.*

2.1.1 Security Services

Security services are tools used to prevent or detect attacks. They are used to improve security by imitating the roles of physical tools such as signatures, seals, and dates on letters.

There exist six types of security services: *confidentiality*, *integrity*, *authentication*, and *non-repudiation* [1].

2.1.1.1 Confidentiality

Confidentiality is an important issue when sensitive information must be protected from opponents or unauthorized entities. This concept is closely related to the concept of privacy. Mostly, confidentiality is performed with encryption/decryption operations. Section 2.3.1 addresses the problem of confidentiality.

2.1.1.2 Integrity

One must be sure that the content of sensitive information has not been changed or corrupted, injected, erased and disordered. Integrity means that information consistency is provided and that the information is tamper-proof. This feature of security may be provided through *Hash Based Message Authentication Codes* (H*MAC*) [1], *Hash functions* [1] and *Digital Signatures* [1]. Hash functions, HMAC and Digital Signatures are discussed in Sections 2.2.3, 2.2.4 and 2.3.2 respectively.

2.1.1.3 Authentication

Another major issue in security is authentication, which is the process of proving the identity of an individual or system. The reader is advised to read [31] for more information about authentication.

2.1.1.4 Non-Repudiation

Non-repudiation is a proof of existence of an agreement between identified entities. In other words, it can be confirmed that the sender and the receiver of an electronic message is, in fact, the parties who claimed to send or receive the message.

2.1.2 Security Mechanisms

Security mechanisms are cryptographic tools or techniques used to form security services in order to prevent, detect, and recover attacks. Some security mechanisms defined in [1] are as follows:

- **Encryption/Decryption:** The process of cloaking a *plaintext* (clear text) message in such a way as to hide its contents is encryption. An encrypted message is called *ciphertext*. The process of turning ciphertext back into plaintext is called decryption.

- **Cryptographic Hash Function:** The output of a *one-way* function (*hash* function) that takes a variable length input and converts it to fixed length output which is called Message Digest. A hash function is a function that works in one direction. It is easy to compute the Message Digest of arbitrary data but it is not easy compute the data given the Message Digest. Hash functions are the building block for many security services.

- **Digital Signature:** A special transformation of data, in order to provide a mechanism that proves the source and integrity of data. The reader is advised to read [32, 33, and 34] for more information on digital signatures. In section 2.3.2, digital signatures have been discussed in detail.

- **Authentication Exchange**: A mechanism used to ensure the identity of an entity by swapping data.

- **Notarization**: A mechanism in which a trusted third party is used in order to provide particular aspects of data exchange.

Security mechanisms are further depicted in section 2.2.

2.2 Overview of Cryptography

Millions of people using insecure-media for communication, such as the internet, require privacy and security. Cryptography is a tool that provides privacy and security. The word cryptography means the study of secret writing. In history, cryptography has been used for military applications in order to prevent the capture of sensitive information by enemies. Nowadays, advanced cryptography techniques are used as security mechanisms in order to provide secure electronic communication for civil or military applications.

Figure 1 shows the basic secure communication model used in cryptography. The secure communication model consists of three entities: Alice, Bob, and Mallory. In this scenario, Alice has a secret message or plaintext that she wants to send to Bob through an insecure channel and Mallory is a malicious user that wants to read Alice's messages to Bob. Alice encrypts her plaintext with an encryption key that both Bob and Alice have previously agreed on. The encrypted plaintext is called ciphertext, which is data not readily intelligible. Alice sends this ciphertext to Bob through an insecure channel in which the malicious user Mallory eavesdrops on the channel to get the packets sent by Alice. Even though Mallory may know the encryption/decryption algorithm used, without knowing the correct decryption key he cannot correctly decrypt the ciphertext. On the other hand, Bob receiving the ciphertext has the decryption key. Bob decrypts correctly the ciphertext and obtains the original plaintext created by Alice. Therefore, privacy of the plaintext is provided by means of encryption and decryption.

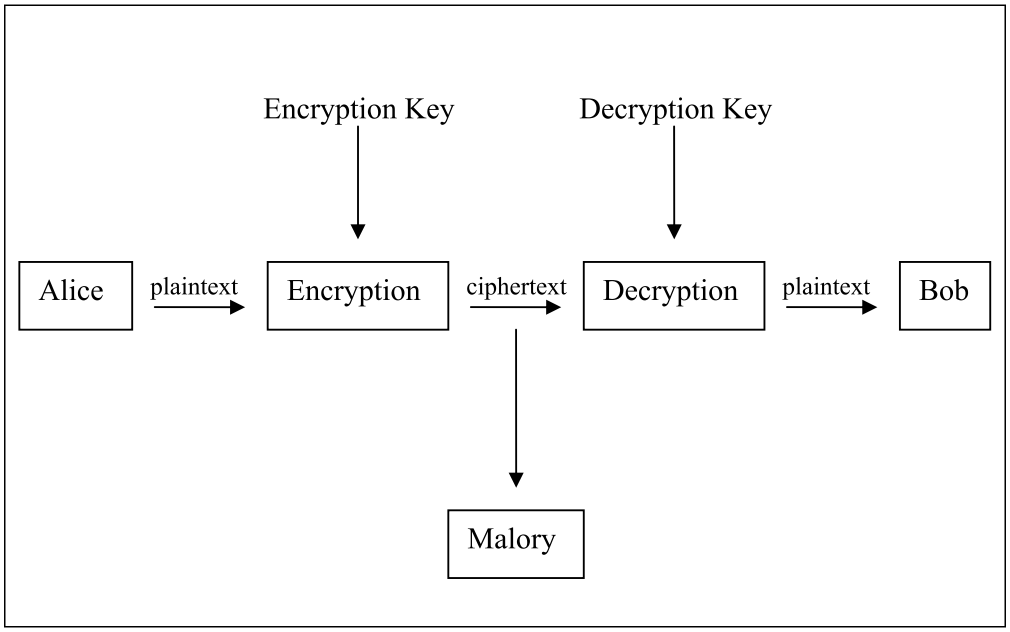


Figure 1. Basic Cryptographic Communication Model

There are mainly two types of approach: *Symmetric Cryptography* and *Asymmetric Cryptography* [1]. Asymmetric cryptography is sometimes called *Public Key Cryptography*.

2.2.1 Symmetric Cryptography

Symmetric cryptography, which is also known as classical, conventional, private-key, single-key cryptography, is the traditional one. In symmetric cryptography, both encryption and decryption algorithm use the same key. Other than encryption/decryption, symmetric cryptography can be used to provide authentication and integrity services by using Message Authentication Codes (MAC) [1] or Hash Based Message Authentication Codes (HMAC) [1].

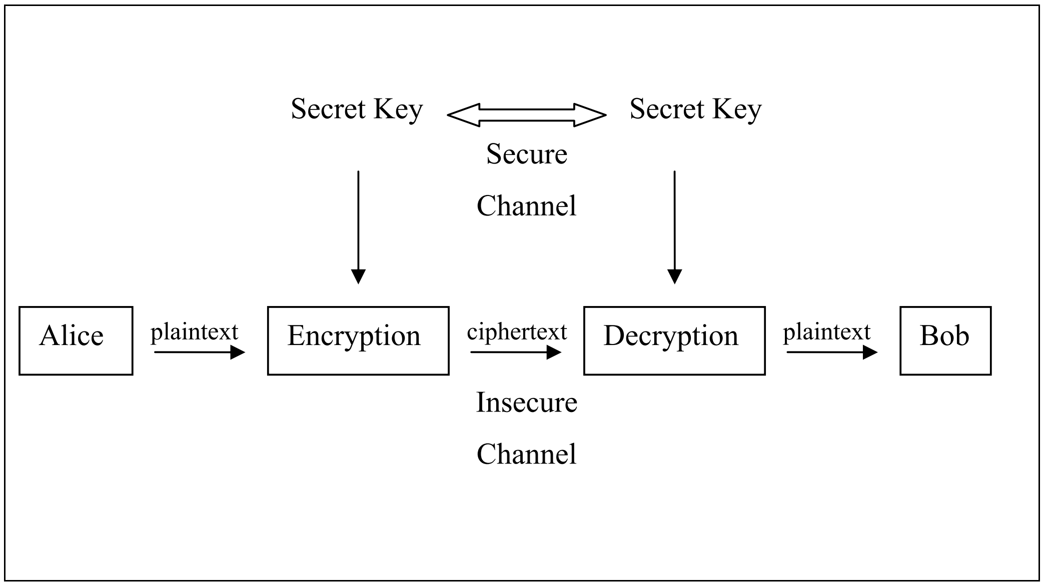


Figure 2. Symmetric Cryptography Communication Model

Figure 2 shows a typical communication model with symmetric cryptography. Alice wants to send a plaintext message to Bob while providing privacy. Alice encrypts her message using an encryption algorithm with a symmetric secret key. Subsequently, Alice sends the encrypted message, the ciphertext, to Bob. Bob receiving the ciphertext decrypts this message with the symmetric secret key that Alice has used for encrypting the message.

The encryption algorithm and the decryption algorithm are mathematically related such that the decryption algorithm does inverse operations of the encryption algorithm if the same symmetric key is used on both algorithms.

As it can easily be seen from the Symmetric Cryptography Communication Model, the two communicating parties must have previously exchanged the symmetric secret key through a secure channel. Therefore, symmetric cryptography has the problem of key distribution or management.

Some symmetric cryptography algorithms are the Rijndael (AES) [20], the International Data Encryption Algorithm (IDEA) [21], RC6 and RC5 [22] and the Data Encryption Standard (DES) [23].

2.2.2 Asymmetric Cryptography

In symmetric cryptography, the sender and the receiver must agree on a secret key before communication starts. This problem of key agreement or management leads to the invention of asymmetric cryptography in other words *public-key cryptography*. Whitfield Diffie and Martin Hellman invented this new concept asymmetric cryptography [24] in 1976.

Asymmetric cryptography is used to provide digital signatures, encryption/decryption and key exchange functionality. Table 1 shows some asymmetric cryptosystem comparisons for their functionality.

Table 1. Asymmetric Cryptosystems Functionality

Algorithm	**Encryption/Decryption**	**Digital Signature**	**Key Exchange**
RSA	Yes	Yes	Yes
Elliptic-Curve	Yes (Rarely Used)	Yes	Yes
Diffie-Helman	No	No	Yes
DSS	No	Yes	No
ElGamal	Yes	Yes	No

In asymmetric cryptography systems, two keys are used: *public-key* and *private-key*. Public-keys may be known by anybody. Public-keys may be used in order to encrypt messages, to verify digital signatures (detailed explanations on digital signatures may be found in Section 2.3.2), and to exchange secret keys.

A private-key is only known by its owner. Private-keys are usually used to decrypt messages and to sign (create) digital signatures. Public-keys and the private-keys are mathematically related to each other; however, it is not feasible to derive a private key from a public key. It is computationally easy to encrypt/decrypt messages when the relevant keys are known. Figure 3 shows the usage of asymmetric keys.

ciphertext = E_{KU}(plaintext), easy to compute when KU and plaintext are known

plaintext = D_{KR}(ciphertext) , easy to compute when KR and ciphertext are known

E : Encryption algorithm

D: Decryption algorithm

KU: Public Key

KR: Private Key

Figure 3. Usage of Asymmetric Keys

A typical asymmetric-key encryption/decryption model is shown in Figure 4. First, Alice encrypts her plaintext message with Bob's freely accessible public key and sends the encrypted message (ciphertext) to Bob. Bob receiving this ciphertext decrypts it using his private key and obtains the original plaintext. Since no one except Bob owns Bob's private key, only Bob can correctly decrypt this message and obtain the original plaintext.

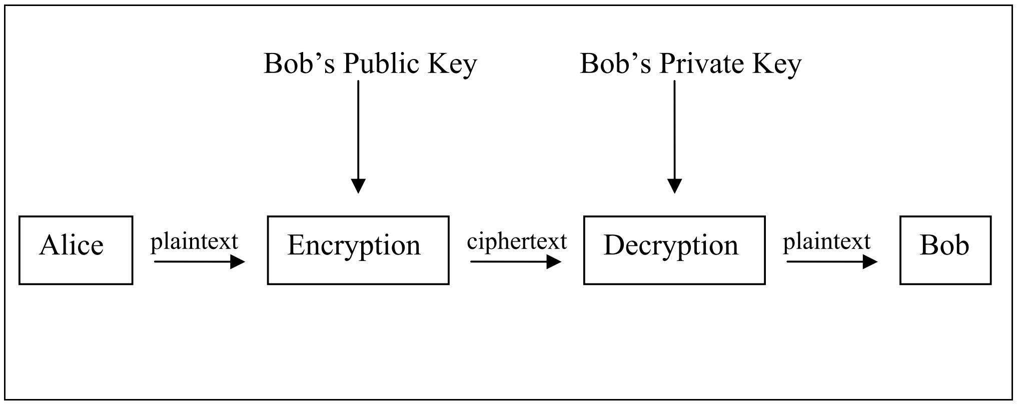


Figure 4. Asymmetric Encryption/Decryption Model

Figure 5 shows how authentication is achieved using asymmetric cryptosystems. Alice encrypts a plaintext message with her private key and sends the ciphertext to Bob. Bob receiving this encrypted message decrypts it with Alice's public key. Since, Alice's public key is known publicly, anyone who captures Alice's ciphertext message may decrypt it and therefore read the original plaintext. Consequently, this scheme does not provide confidentiality, but provides authentication. Since, Alice's private key is only known by Alice, no one other than Alice could create this ciphertext. Therefore, when Bob decrypts this ciphertext with Alice's public key and obtains an intelligible message, Bob authenticates Alice's message.

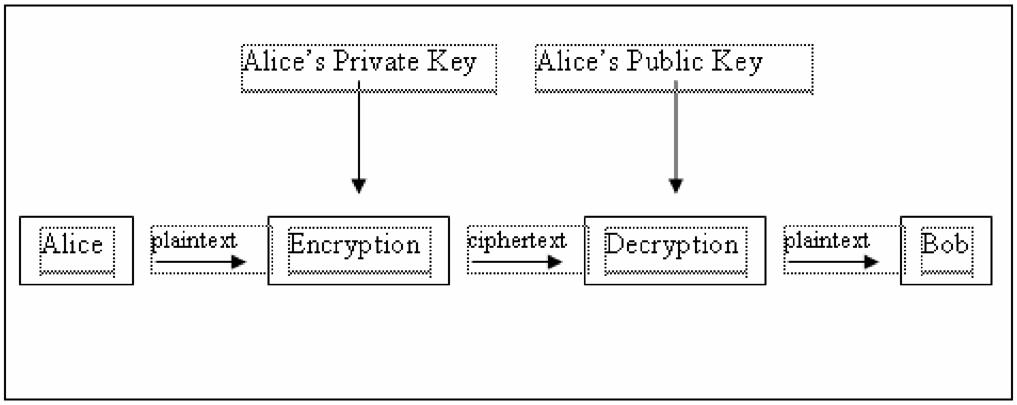


Figure 5. Authentication with Asymmetric Cryptosystems

As it can be seen from Figures 4 and 5, public and private keys are used such that if one is used for encryption the other is used for decryption.

Asymmetric cryptography is much slower than symmetric cryptosystems. For this reason, asymmetric cryptosystems should not be used for encrypting large amounts of data. Asymmetric cryptography does not replace symmetric cryptography and it is not considered more secure than symmetric cryptography. Furthermore, key distribution in asymmetric cryptography is not trivial since making keys public is also a hard problem. Therefore, key distribution is easier but not trivial.

The most popular public key cryptosystem is RSA [25]. Elliptic Curve [26], ElGamal [27], Diffie-Helman [24] and Digital Signature Standard (DSS) [33] are other widely used public-key cryptosystems. Moreover, comparisons of several asymmetric cryptosystems may be found in [28].

2.2.2.1 An RSA Example

In this section an example of an asymmetric cryptosystem, RSA is presented.

1. Two unequal prime number p and q are randomly selected:

$$p = 47$$

$$q = 73$$

2. The product of p and q is calculated:

$$n = p * q = 3431$$

3. Euler totient Ø of the two primes p and q is calculated:

$$\text{Ø} = (p - 1) * (q - 1) = 3312$$

4. A number e is randomly selected such that $1 < e < n$ and $\gcd(e, Ø) = 1$

$$e = 425$$

5. The modular inverse of e is calculated

$$d = e^{-1} \bmod Ø = 1769$$

6. Private-key = { d } and public-key = {e , n}

7. Assume plaintext is represented as a number

$$m = 707$$

8. Encryption operation is computed as follows: $c = m^{e} \pmod{n}$

$$\text{ciphertext} = c = 707^{425} \pmod{3431} = 2142$$

9. Decryption operation is computed as follows: $m = c^{d} \pmod{n}$

$$\text{plaintext} = m = 2142^{1769} \pmod{3431} = 707$$

2.2.3 Hash Functions

Hash functions are one-way functions that get variable size input and produce fix size small output, which are also called *message digests*. Given the digest, it must be computationally infeasible to find the original message. Furthermore, hash functions must be *collision resistant*; finding out two messages that produce the same hash result must be impractical. Some of the most commonly used hash functions are Message

Digest-5 (MD5) [29] and Secure Hash Algorithm-1 (SHA1) [1]. MD5 produces 128-bit digest, SHA-1 produces 160-bit digest.

2.2.4 Hash Based Message Authentication Codes (HMAC)

HMAC [1, 12] is a technique that uses a secret key to generate a small fixed-size block of data from arbitrary large messages. HMAC is not necessarily reversible. A secret key must be shared between the sender and receiver. HMAC is also called *cryptographic checksum* and is usually appended at the end of a message. The receiver of this message usually performs the same HMAC operation on the message and compares the result to the senders HMAC value. This way the receiver gets assurance that a message has not been altered during transmission and further, the receiver authenticates the sender of this message. However, HMAC is not a signature since both the receiver and the sender can generate the same HMAC. Furthermore, by knowing a message and its HMAC value, it should be computationally infeasible to find another message with same HMAC value.

2.3 Cryptographic Solutions to Security Problems

In this section, cryptographic solutions to some security problems are discussed. The problem of confidentiality, integrity, digital signatures, and digital envelopes are discussed.

2.3.1 Confidentiality with Symmetric Encryption

Using symmetric encryption is one of the most frequently used methods for providing confidentiality. Two entities that have previously agreed on a secret key can provide confidentiality of their messages by using symmetric encryption. If an adversary intercepts the encrypted messages, he/she cannot restore it without knowing the correct symmetric key. Therefore, each entity receiving a message can be assured that the message is sent in a confidential way.

2.3.2 Authentication and Integrity with Digital Signatures

Digital signature [25, 32, 33, and 34] is a mechanism for non-repudiation, authentication, and integrity. The basic idea behind digital signature is to use the private key on the message to produce a piece of information that can only be produced by a single entity. Furthermore, this signature can be verified by decryption with the public-key; as a result, the verification can be carried out by anybody. Generally, digital signatures are produced and verified over hash of the message in order to provide integrity.

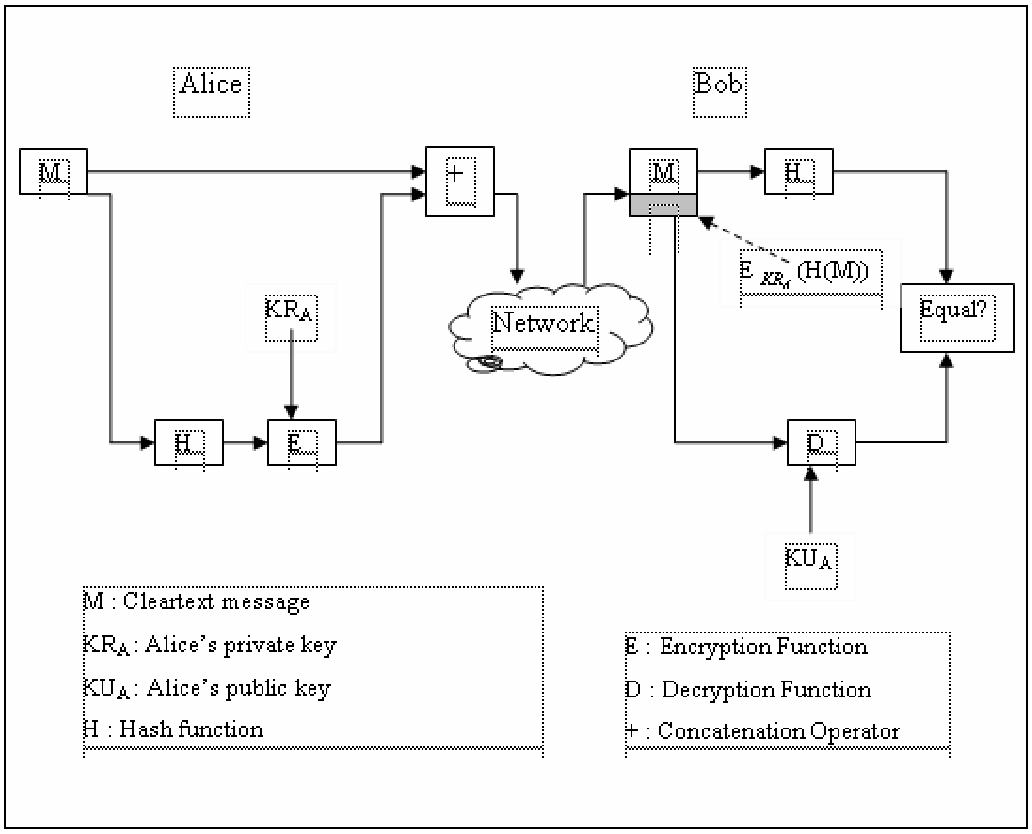


Figure 6. Digital Signature Model Based on RSA

Figure 6 shows RSA digital signature model [25] in which two entities Alice and Bob communicate. In this communication, digital signatures are used in order to provide authentication and integrity. First, Alice generates a message M; second, she produces the hash value of the message H(M); and third, she encrypts this hash value with her private key as follows: E_{KR_A} (H(M)). Finally, she sends the signature E_{KR_A} (H(M)) to Bob together with the original message M. Bob receiving the message M and the signature E_{KR_A} (H(M)), first generates the hash of the message M. Second, he decrypts the signature with Alice's public key and obtains the digest of the message M as follows: D_{KU_A} (E_{KR_A} (H(M))) = H(M). If the digests obtained by Bob in the first two steps are equal, Bob can be sure that message M is produced by Alice and that the contents of the message have not been altered.

2.3.3 Digital Envelopes

Digital Enveloping [1, 12] is a nice solution for fast message exchanging, which uses the speed of symmetric cryptography and security of asymmetric cryptography. A digital envelope has two parts: a message encrypted with a symmetric session key, and the symmetric session key encrypted with the public-key of the recipient.

Figure 7 shows a typical digital enveloping mechanism. In this figure, there are two entities: Alice and Bob. Alice wants to send a confidential message M to Bob. Alice has Bob's public-key KU_B. Alice, does not want to encrypt whole message with Bob's public key since public key encryption is slow so Alice encrypts a randomly generated symmetric session key KS with Bob's public-key as follows: E_{KU_B} (KS). Subsequently, Alice encrypts her message M with symmetric session key KS is done in the following way: E_{KS}(M). Lastly, Alice sends the encrypted session key E_{KU_B} (KS) and the encrypted message E_{KS}(M) to Bob. Having received these two fields, Bob first, decrypts the session key with his private key as shown below: D_{KR_B} (E_{KU_B} (KS)) = KS. Second, Bob decrypts the message using the symmetric session key KS, obtained in the previous step as in the following manner: $D_{KS}(E_{KS}(M)) = M$.

Digital Enveloping is a mechanism that improves the performance of key exchange by joining the strengths of symmetric cryptography and asymmetric cryptography.

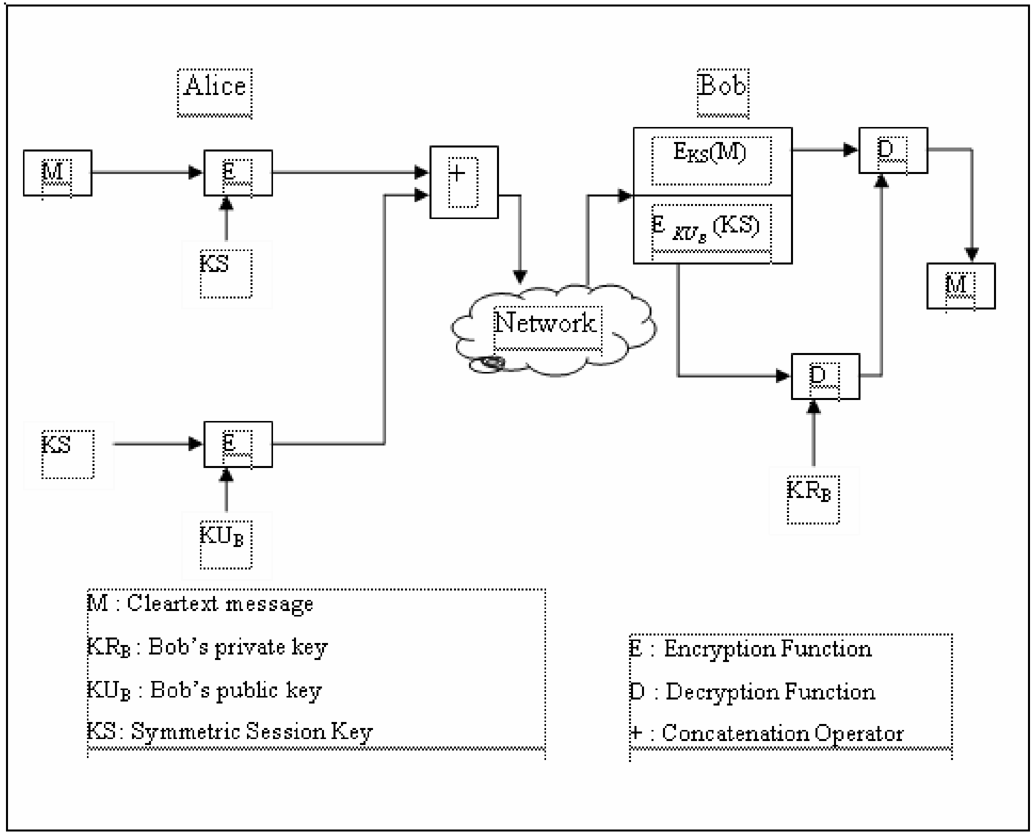


Figure 7. Digital Enveloping

3 RELATED WORK

Assume that there exist two entities that has some electronic good the other one wants. A fair exchange protocol is a protocol guaranteeing that either both entities get what they want, or neither of them gets anything.

In the literature, exchange protocols, contract-signing protocols, digital certified mail protocols, and e-commerce protocols try to solve similar problems. The difference between these protocols is the content of the exchanged items. Some details related to these protocols are found below.

3.1 Exchange Protocols

In Exchange Protocols, two entities want to exchange arbitrary electronic items. These items may be e-money, digital signatures, receipt of mails or any kind of e-goods. Exchange protocols, contract-signing protocols, digital certified protocols and e-commerce protocols can be categorized using two different approaches: *protocols using trusted third parties* and *baby-step protocols* [12].

3.1.1 Protocol Using Trusted Third Parties

In protocols that use trusted third parties, two entities want to perform an exchange by means of a trusted third party. These protocols have two approaches: *online trusted third party* protocols and *optimistic* protocols.

3.1.1.1 Online Trusted Third Party Protocols

In protocols with online trusted third parties, both entities send their items to the trusted third party that makes the exchange by sending the appropriate item to the appropriate entity. However, this approach has some drawbacks:

- **Cost**: Online trusted third parties must always be available. Thus, they must maintain a bandwidth capable of handling enormous traffic.

- **Congestion**: Several messages are routed to and from the online trusted third party, which may create a major bottleneck in the network.

- **Liability**: Trusted third party is actually a single point of failure. A possible crash causes disastrous consequences. Trusted third parties require costly precautions that require large operating costs.

Some protocols with online trusted third parties may be found in [3, 5, 9, 14, and 15].

3.1.1.2 Optimistic Protocols

The second approach using trusted third parties is called optimistic protocols. In optimistic protocols, trusted third parties do not participate in the protocol if both entities are honest. Therefore, with optimistic protocols, congestion and costs are minimized. If one of the entities does not get its item while the other does, this unjust situation is reported to the trusted third party and the trusted third party solves this problem within the protocol. The trusted third party does not need to store any secret of any party or to store anything about an exchange after helping the unjustly treated side. Some optimistic protocols may be found in [4, 7, 8, 9 and 19]. Failure analysis of [7] may be found in [10 and 11].

3.1.2 Baby-Step Protocols

Baby-step protocols are exchange protocols without trusted third parties, where the probability of protocol completion is gradually increased over several cycles. It is better to explain baby-step protocol concept using an example. Assume that Alice and Carol want to exchange their signature on a contract. Firstly, Alice writes the first character "A" of her name on the contract and sends it to Carol. Carol receiving the contract with the first character of Alice's name writes the first character "C" of her name on the contract and sends it back to Alice. Alice receiving the contract adds the second character of her name "Al" to the contract. These sequences of events continue until each entity signs their complete name on the contract; hence, they approach the solution with baby steps.

These types of protocols have a disadvantage: Stopping the protocol before completion may result in a situation where one of the entities is closer to the solution. For example, assume that the contract has "Ali" and "Ca" as signature on it. Carol is closer to the solution. Moreover, assume that these signatures are digital signatures and that Carol tries a brute force attack to obtain Alice's complete signature on the contract. Carol has a one-step advantage to perform a brute force attack as compared to Alice's chance to do it. Carol may successfully complete the brute force attack and she may create an asymmetric situation where Alice is bound to the contract but Carol is not. This situation is contradictory to the fairness property and is a serious breach of security. Even if Carol and Alice have signed the same amount of characters, Carol may have much greater computation power than Alice and therefore she may be closer to the solution. Consequently, equal computation power cannot be assumed.

Although it seems like exchange protocols with trusted third parties are better solutions than the baby-step approach (exchange protocols without trusted third parties), sometimes baby-step approach may be better suited due to different security requirements and because sometimes problems require a *peer-to-peer* approach rather than a *client-server* approach. For example, consider a multimedia exchange protocol similar to *Kazaa* [16] or *Napster* [17] is designed. Kazaa and Napster are peer-to-peer multimedia exchange programs that do not display the fairness property. In these types of systems, several clients communicate with each other in order to download multimedia files.

One of the most important features of a fair multimedia or even a general exchange protocol is to check the quality of the exchanged products. Assume that Alice and Bob want to exchange some movies using a fair multimedia exchange protocol. Alice claims that she has Movie A and Bob claims that he has Movie B. If there is no quality control in the protocol, Alice may create some dummy file, rename this dummy file as Movie A, and send this file to Bob using the protocol. At the end of the protocol, Bob will realize that although he and Alice have fairly exchanged some data, the quality of the data exchanged is not as Alice has claimed. This example shows that, fair multimedia exchange protocols must provide a quality control mechanism. Furthermore, this quality control mechanism requires human inspection; i.e. understanding whether a movie file named Movie A is really Movie A. The quality control in a fair multimedia exchange protocol can be performed by a trusted third party or by client entities. If the protocol is designed to be scalable as Kazaa and Napster, it is not possible for the trusted third party to verify the quality of each exchanged multimedia file. There are too many files for human inspection at a centralized server. For this reason, the quality control mechanism of fair multimedia exchange protocols must be set in motion by the client entities; each client must decide for itself whether the claimed goods are consistent with the actual goods.

Multimedia exchange protocols in which client entities perform quality control can be implemented with the baby-step approach (exchange protocols without trusted third parties). For example, Alice will send some sub part of Movie A to Bob. Bob receiving this sub part of a movie file will watch it and decide whether this is truly some subpart of Movie A. If so, Bob will send some sub part of Movie B to Alice, and so on. If neither of the entities stops the protocol early, each entity will end up with a complete movie. If either part stops the protocol early, both entities will be able to watch approximately the same amount of movie but not the entire movie.

The security needs of a multimedia exchange protocol are much less than the security needs of an e-commerce protocol, since in e-commerce protocols the exchanged items are money and e-goods; however, in multimedia exchange protocols the exchanged items may be video or audio files. Furthermore, the definition of fairness has a different meaning: two entities that want to perform exchange both obtain either the complete items they want or approximately the same amount of data from the item. Some baby-step approach protocols may be found in [12 and 13]. In [6], there is some work on comparisons of different types of exchange protocols.

3.2 Simultaneous Contract Signing Protocols

In simultaneous contract signing protocols, two entities want to exchange digital signatures on a contract simultaneously. Neither entity wants to send its signature before receiving the other entity's signature since the other entity could vanish after receiving a signature. This problem is similar to e-commerce protocols, since instead of exchanging e-money with an e-product, signatures are exchanged between entities. Some simultaneous contract signing protocols may be found in [2, 4, 5, 12, 13 and 19].

3.3 Digital Certified Mail Protocols

In digital certified mail protocols, two parties are involved. The first party Alice wants to send a message to the second party Bob, but she does not want him to read it without signing a receipt. Therefore, Alice wants to exchange her message with Bob's signature on this message. This situation is similar to a conventional mailing system where a mail carrier brings a letter to a destination address. The mail carrier will not deliver the mail until obtaining the signature of the property owner. Digital certified mail protocols are similar to the problem in general exchange and contract signing protocols, only the content of exchanged items differs. Some digital certified mail protocols may be found in [4, 13 and 19].

3.4 E-Commerce Protocols

Two entities are involved in an e-commerce transaction: a customer and a merchant. The merchant provides some service or transmits some e-goods in response to a customer's e-money. Some e-commerce protocols may be found in [6, 7, 8, 9, 14 and 15]. Furthermore, failure analysis of some e-commerce protocols can be found in [10 and 11].

In [4], Silvio Micali proposed several optimistic protocols for certified mail applications and for contract signing applications in which two entities, Alice and Bob, are involved. These two types of protocols guarantee, respectively, a fair exchange of mail in return for a receipt or exchange of digital signatures. Therefore, either both Alice and Bob will get what they want, or neither of them will receive anything. Furthermore, both of these protocols are enhanced by guaranteeing the termination of the protocol at a given cut-off time.

Figure 8 shows one of Micali's proposed protocols "An Optimistic Protocol for Fair Certified E-mail". In certified e-mail protocols, e-mail messages are exchanged for their receipts. In the proposed protocol, Alice (A) exchanges her e-mail message M in return for Bob's (B's) receipt. In the first message of the protocol, Alice sends Bob an e-mail message M, her identity information A, and Bob's identity information B, all encrypted with the trusted third party's public-key. Bob receiving this message cannot read the contents since he lacks the correct decryption key (third party's private key KRTP). Subsequently, in the second message, Bob sends the receipt of the e-mail, which is actually Bob's signature over the first message Z (SIG_B(Z)). Alice receiving the second message checks the validity of Bob's signature; if valid Alice sends the e-mail message M in the third message. If both parties behave honestly, after this step, the protocol is completed since both parties have obtained the items they expected. M received as part of the first message must be equal to M received in the third message. In order to check this equality, Bob performs the encryption E_{KUTP}(A,B,M) using the value M obtained from the third message and compares the result with the first message. If these two values are not equal, Bob concludes that Alice has cheated and therefore he applies to the trusted third party for dispute resolution. In the fourth message, Bob sends Z and the e-mail receipt $SIG_B(Z)$ to the trusted third party. The trusted third party receiving this message checks the validity of Bob's signature. If valid, the third party resolves this dispute by sending the appropriate item to the appropriate entity; in other words, the third party sends the e-mail receipt $SIG_B(Z)$ to Alice and the e-mail message M to Bob. However, in order to do so, the trusted third party first has to retreive the e-mail message from Z by performing the following decryption: $D_{KRTP}(Z) = = (A, B, M)$.

```
1. $A \rightarrow B: Z = E_{KUTP}(A,B,M)$

2. $B \rightarrow A: SIG_B( Z )$

**If ( $SIG_B( Z )$ is correct )**

        3. $A \rightarrow B: M$

          **If $E_{KUTP}(A, B, M) != Z$**

                4. $B \rightarrow TP: Z, SIG_B( Z )$

                **If ( $SIG_B( Z )$ is correct )**

                        5. $TP \rightarrow A: SIG_B( Z )$
                        6. $TP \rightarrow B: M$
```

Symbol	Meaning
A	Alice
B	Bob
TP	Trusted third party
KUTP	Public-Key of TP
KRTP	Private-Key of TP
M	Mail Message
$A \rightarrow B$: X	A sends X to B
E_{KUTP} (X)	Assymetric Encryption of X using the key KUTP
D_{KRTP} (X)	Assymetric Decryption of X using the key KRTP
$SIG_B(X)$	B's digital signature on X

Figure 8. An Optimistic Protocol for Fair Certified E-mail

3.4.1 Desired Properties of E-commerce Protocols

Some desired properties of e-commerce protocols have been stated below:

1. The protocol must be fair. Assume that a merchant has e-products to sell and that a client has e-money. The protocol must have two possible outcomes:

    - Either, the client obtains the e-product and the merchant obtains the e-money, or
    - Both entities obtain nothing.

2. The protocol must not assume that entities have equal knowledge of protocol and equal computation power.

3. The trusted third party will not participate in any execution in which the merchant and the customer are honest. If the customer pays but does not get any e-products then he/she will prove this injustice to the trusted third party and the customer will get the e-products from the trusted third party. This property of keeping the trusted third party out of normal execution avoids congestion at the trusted third party with a minimum cost.

4. Messages within the dispute resolution protocol must be minimized. It must be simple and therefore fast.

5. E-goods must be transferred only once per protocol run. In most of the previous e-commerce protocols proposed in the literature [3, 7, 8, 9, 15 and 19], it is assumed that the merchant's e-goods are small. These protocols transfer e-goods multiple times in order to establish dispute resolution. This situation may be excessively costly if the e-good size is large. An example to a large e-good is a movie file.

6. Disputes must be resolved within the protocol, not by gathering evidence and taking them to a court.

7. The trusted third party must not store any secrets, e-goods or protocol messages (i.e. In [7] both TP and Merchant stored copies of the e-goods).

8. Clients must be anonymous. Anonymity provides that the identity of an entity will not be revealed during an e-commerce transaction. This is especially significant for customers who do not want the merchant to discover their patterns of shopping habits.

9. Quality control of the e-products must be provided. The protocol must guarantee that the e-product contents and prices are as the merchant has claimed them to be. Either a customer may perform the quality control for his/her self or the trusted third party may perform it.

Table 2 shows comparison of several protocols for their desired properties according to the descriptions stated above. If a protocol provides a certain property, the symbol "√" is used. If not, the symbol "X" is used. If a property cannot be evaluated using a certain protocol, the symbol "N" (standing for: not applicable) is used. Finally, "[*]" is the proposed optimistic fair e-commerce protocol.

Table 2. Comparison of Several E-Commerce Protocols

		Desired Properties of e-commerce Protocols								
Exchange Protocols		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
	[3]	√	√	X	N	X	√	√	√	X
	[4]	√	√	√	√	N	√	√	√	N
	[7]	√	√	√	X	X	√	√	X	√
	[8]	√	√	X	X	X	√	√	√	√
	[9]	√	√	X	N	X	√	X	√	√
	[13]	X	X	√	N	√	X	N	√	X
	[14]	√	√	X	√	√	X	√	X	X
	[15]	√	√	X	N	X	N	√	√	√
	[19]	X	√	√	X	X	X	√	X	X
	[*]	√	√	√	√	√	√	√	√	√

4 DESIGN OF FAIR E-COMMERCE PROTOCOL

In this section, this researcher proposes an optimistic fair e-commerce protocol. First, assumptions are presented. Third, preliminary concepts, chain keys and offline certification, are described. Finally, the proposed protocol is described.

4.1 Assumptions

Assumptions for the proposed protocol are as follows:

- The public keys of each player are securely distributed before the protocol run.
- The customer has already browsed the merchant's web page and selected the item to be purchased before the protocol run.
- The merchant and the trusted third party accept tokens (see Section 4.2 for details) as a valid payment method and both can verify whether a token is valid (whether the claimed credit card number exists and has enough money in the account) or not. The merchant contacts a bank entity in order to verify a token. The token is assumed to have the format of the *Purchase Request Message* in *Secure Electronic Transaction (SET)* [1], which is a world wide standard for payment tokens. These types of tokens are idempotent; in other words processing the same token multiple times does not mean that the amount of money to be transferred will also multiply.

- Before the protocol starts, the trusted third party certifies each product in terms of its price, description, and contents (See Section 4.4 for details).

- The integrity and authentication of each message is provided by appending the digital signature of that message. For the sake of simplicity, these signatures are not shown in the protocol.

- Encryptions are strong enough so that it is not possible to decrypt a message without the correct decryption key.

- Bank entities that are involved in the payment procedure (see Section 4.2 for details) and the third party are assumed to be trustworthy.

- Communication between the trusted third party and the other players can be delayed by an arbitrary, but finite amount of time by an attacker. However, the trusted third party will eventually receive messages.

- An attacker may gain complete control of the communications between the merchant and the customer. In other words, the attacker may prevent the customer from sending messages to the merchant and visa versa for an indefinite period.

- Communication failures between the customer and the merchant are considered as misbehavior of an entity and therefore dispute resolution commences. In other words if for any reason the communication between the customer and the merchant is disrupted, the client will assume that the merchant is cheating and therefore, the client will apply to the third party for dispute resolution.

- Each of the players is able to compute and verify digital signatures and to compute collision resistant one-way hash functions. More information regarding digital signatures and hash functions may be found in [18, 25, 32, 33, and 34], Section 2.3.2 and in Section 2.2.3.

4.2 The Payment Token

Figure 9 shows the payment token which has the SET [1] Purchase Request Message format. The PI field (payment information field) contains customer’s credit card number, destination account identifier and amount of money to be transferred. The PI field is intended to be processed only by the bank entity. The OI field (order information field) contains details of e-products such as a product identifier number, description string of the e-product and price of the product. The OI field is intended to be processed only by the merchant entity. The PI and OI fields are linked together in special technique called *Dual Signature* [1].

Stallings [1] discusses the function of dual signatures in the purchase request message. He states that dual signatures serve as a link between two messages that are anticipated for two different recipients. In the proposed protocol, the customer intends to transmit the OI to the merchant and the PI to the bank entity. The merchant has no need to know the customer’s credit card information and the bank does not need to know the customer’s order information. The customer must provide further protection in order to keep these two items separate while providing privacy. On the other hand, the OI and PI must be linked for dispute resolution if required. This link is essential to enable the customer to prove that a certain payment is for a certain order and not for some other orders. In order to understand the necessity for this link, take the case of the customer who transmits two messages, a signed OI and a signed PI, to the merchant, who then passes the PI to the bank entity. If this merchant acquires another OI from this customer, the merchant could argue that this OI is intended for the PI instead of the original OI. A dual signature prevents this situation.

Stallings [1] goes on to argue that in order to create a dual signature the customer gets the hash of the PI and the hash of the OI. Subsequently, these two digests are concatenated and the hash of the result is obtained. In the final step, the dual signature is formed by encrypting the final digest with the customers private key, KRC, as follows: $DS = E_{KRC}(H(PI) \| H(OI))$. Assume that the merchant has received a dual signature (DS), an OI, the message digest of a PI (PIMD) and the customer's public-key. The merchant calculates these two values: $H(PIMD \| H(OI))$ and $D_{KUC}(DS)$. The bank entity has verified the signature if these two quantities are equal.

Stallings [1] also exemplifies the security of dual signatures. In the example case, the merchant wants to replace the legitimate OI with another OI in this transaction to benefit himself/herself. The merchant then has to create another OI whose hash corresponds to the existing OIMD. When secure hash functions are used, this is a costly process. Thus, the merchant is unable to link another OI with this PI.

Note that the *Digital Envelopes* [1, 12] (see Section 2.3.3 for details) are used in order to prevent asymmetric bulk encryption of PI, dual signature and OIMD.

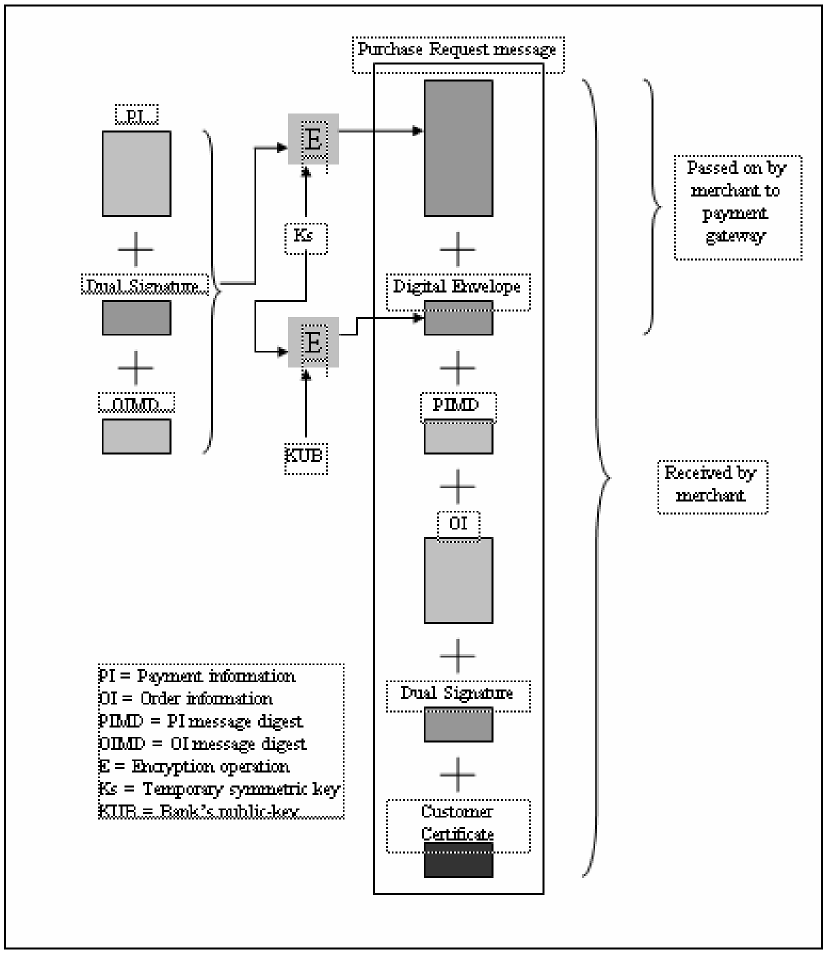


Figure 9. SET Purchase Request Message

4.3 Chain Keys

In the proposed protocol for each purchase of a product, one symmetric key will expended. In order to reduce the total numbers of keys to be stored, the third party will generate the n session keys $KS_{i=1..n}$ with a special method called *Chain Keys* , which is a concept introduced in [38].

Figure 10 shows the steps involved in producing chain keys. First, the third party generates a random symmetric session key called the *Root Key* or in other words KS_1. Second, the third party computes the HMAC [1, 12] (See Section 2.2.4 for details) of the Root Key using a key, *HMAC Key*, and obtains the second key of the chain, which is KS_2. Third, the third party computes the HMAC of KS_2 again using the same HMAC Key and obtains KS_3. This process continues until all keys ($KS_{i=1..n}$) are produced. The aim of this method is to generate multiple keys derived the Root Key and a single HMAC key.

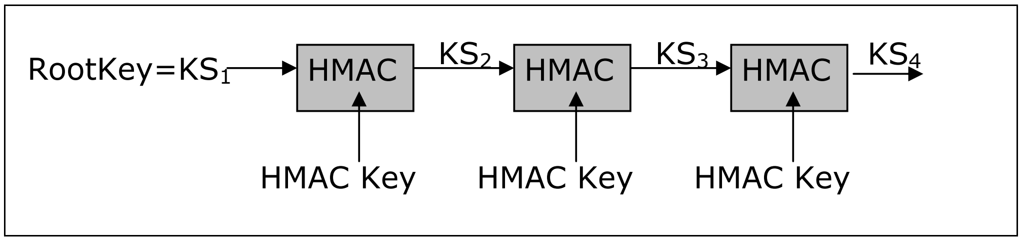


Figure 10. Chain Key Production

4.4 The Offline Certification Process

Before the protocol runs, an offline certification process is employed in order to certify each product in terms of its price, description, and contents. In the offline certification process, the merchant demands *n* certificates from the trusted third party for a certain e-product.

Figure 11 shows the product certificate format. The first field in the certificate is the hash of the encrypted e-product. This field is employed in order to certify the contents of the product. The encryption is performed with a symmetric session key KS_i. This symmetric session key is produced with a special method; *Chain Keys* (see Section 4.3 for details). In this method a chain of keys are produced using two keys *RootKey* and *HMAC Key*. The HMAC and the RootKey key must be known by the entities that will generate the chain key. Therefore, they must be known by both the merchant and the third party. Each one of the *n* certificates of a certain e-product will contain a different hash value of the encrypted product since the product will be encrypted with a different session key in each copy. The second field is a numeric value indicating the price of the product. The third field is a string that describes the product. The fourth field is a unique identifier for each product. The fifth field is the chain key index. The last field is the signature of the trusted third party on the previous fields of the certificate.

In order to resolve disputes the third party does not have to save the copies of the certificates but has to save a Root Key and a HMAC Key per product (See Section 4.5 for details). At the end of this offline certification process, the trusted third party will send the *n* certificates, the Root Key and the HMAC Key to the merchant through a secure channel.

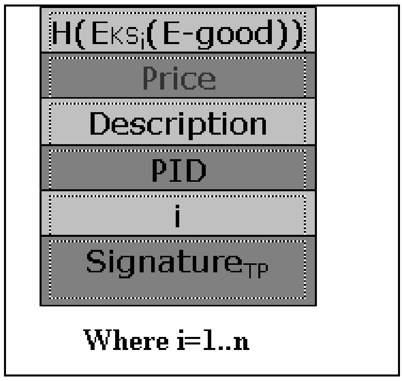


Figure 11. Product Certificate Format

4.5 The Protocol Description

Figure 12 shows the steps involved in the protocol. The integrity and authentication of each message of the protocol is provided by appending the digital signature of that message. For the sake of simplicity, these signatures are not shown in the protocol.

The protocol starts with the merchant sending the encrypted e-good and the certificate of that product. The customer receiving this message wants to be sure that he/she is really buying the product that the merchant has promised and that the merchant has not changed the price or contents. In order to do so, the client will check the certificate for the *Price* and *Description* fields. If satisfied, the encrypted product is hashed and compared to the first field of the certificate. If these two digests (hash values) are equal, then the customer is sure that the merchant is providing the correct product. Knowing that the merchant is not cheating, the client sends the token to the merchant in the second message. Subsequently, the merchant checks the validity of this token. If the token is valid, the merchant sends the product decryption key KS_i encrypted with public-key of the customer in the third message. The customer receiving this encrypted product decryption key decrypts it using it private key as follows: $D_{KRC}(E_{KUC}(KS_i)) = = KS_i$. Subsequently, the customer acquires the e-good by decrypting the encrypted e-good using the product decryption key KS_i as follows: $D_{KS_i}(E_{KS_i}(\text{E-good})) = = \text{E-good}$. Up to this point the normal execution of the protocol is described, below the motive for dispute resolution is depicted.

$CERT_{TP_i} = H(E_{KS_i}(\text{E-good})) || Price || Description || PID || i || Signature$

1) M➔C: $E_{KS_i}(\text{E-good}) || CERT_{TP_i}$
2) C➔M: token
3) M➔C: $E_{KUC}(KS_i)$

Dispute Resolution Phase

- *Customer claims that he/she has not received Message 3*

4) C➔TP: token || i || PID || KUC
5) TP➔C: $E_{KUC}(KS_i)$
6) TP➔M: token || i || PID || KUC

Symbol	**Meaning**
$CERT_{TP}{}^{i}$	ith Certificate of a product signed by the trusted third party
H(X)	Hash of X
E_X (data)	Encryption of data with key X
E-good	An e-product or electronic item such as, database or multimedia file
Price	Price of the e-good
Description	A string describing contents of the product
KUX	Public key of identity X
KRX	Private key of identity X
SIG_X(Data)	Data signed by identity X; equivalent to E_{KR^X} (H(Data))
TP	Trusted third party
M	Merchant
C	Customer or Client
||	Concatenation Operation
PID	Product Identifier

Figure 12. Fair Optimistic E-Commerce Protocol Description

Assume that the normal execution of the protocol runs and that the merchant sends the first message. After receiving the first message if the client does not send the second message the fairness property of the protocol is not violated. This fact is true since in the first message, the e-good is sent encrypted and therefore neither customer has received the e-good he/she expected nor did the merchant receive a token. However, if the customer sends the second message (token) in a legitimate way and the merchant does not reply with the encrypted product decryption key (third message) the fairness property is violated. This fact is true since, although the merchant has received the token the customer has not received the e-good he/she expected. In order to solve this problem the customer applies to the third party for dispute resolution. Below the dispute resolution phase is described.

Assume that the normal execution flows and that after the customer sends the token in the second message, the customer waits and does not get the product decryption key or it gets a wrong product decryption key. The customer applies to the trusted third party for dispute resolution. In the fourth message, the customer sends the token, the certificate index i, the product identifier PID and his/her public key to the trusted third party. In this message, the customer specifies the product decryption key he/she expects by sending the PID and i pair, since the PID specifies the product and i specifies the product decryption key corresponding to that PID. Furthermore, the third party must also obtain the token in this message. This fact is true since the third party is willing to perform the exchange by sending the appropriate item to the appropriate entity. It is important to remember that tokens are idempotent, therefore although the token may be processed two times the money transferred to the merchants account will not double (see Section 4.2 for details). Subsequently, the trusted third party checks the validity of the token. If the token is valid, in the fifth message the trusted third party sends the product decryption key related to the product identified by the PID and the chain key index i to the customer encrypted with the customers public key. The customer receiving this encrypted product decryption key decrypts it using it private key as follows: $D_{KRC}(E_{KUC}(KS_i)) == KS_i$. Subsequently, the customer acquires the e-good by decrypting the encrypted e-good using the product decryption key KS_i as follows: $D_{KS_i}(E_{KS_i}(\text{E-good})) == \text{E-good}$. Finally, the merchant must be informed about this transaction. In order to do so, in the sixth message, the third party forwards the token and the purchased product information (fifth message) to the merchant. The merchant receiving this message processes the token and removes the i^{th} certificate from its database.

A malicious user may try to spend the product decryption keys by skipping the normal execution of the protocol and by directly applying to the third party for dispute resolution. However, in order to do so the malicious user must send a legitimate token. In other words, the malicious user must pay in order to perform this attack. This attack is to the benefit of the merchant and is undesirably costly and infeasible for the attacker.

Another attack may be performed by a malicious user in order to create a bottleneck in the third party's network. The attackers aim is to include the third party in the normal execution of the protocol and therefore to increase the traffic directed the third party. First, the attacker downloads the first message from the merchant. Second, the attacker skips the second and third messages and directly applies to the third party for dispute resolution. Subsequently, the attacker sends the fourth message in legitimate way; in other words, the token, i, and PID values are consistent with the first message and the customer's public-key field is appropriate. This way, the third party is included in the normal execution of the protocol and traffic to the third party is increased. However, this attack is similar to the previous type of attack in that it also requires payment by the malicious user and benefits the merchant. Furthermore, the most costly operation is downloading the encrypted e-good (first message). The cost required to perform dispute resolution is much lower than the cost required for downloading the first message. In reality, in the proposed protocol, the merchant and customer exchange the product decryption key for the token (not the e-good for the token). Excluding the download operation of the encrypted e-good (which can be large) from the dispute resolution phase minimizes the damage of this attack.

5 DESIGN OF FAIR MULTIMEDIA EXCHANGE PROTOCOL

In this section, this researcher proposes a fair multimedia exchange protocol using a baby-step approach [12, 13] (see Section 3.1.2 for details). In a fair multimedia exchange protocol, there are large numbers of clients that communicate in order to barter fairly multimedia files in a peer-to-peer fashion. An important point to remember is the definition of fairness in baby-step protocols: two entities that want to perform exchange both obtain either the complete items they want or approximately the same amount of data from the item.

In a fair multimedia exchange protocol in addition to the problem of fair exchange the problem of quality control must be solved. In other words, entities involved in a transaction must verify claims of other untrustworthy entities using a quality control mechanism. This quality control mechanism requires human inspection; i.e. understanding whether a movie file named Movie A is in fact Movie A. The quality control in a fair multimedia exchange protocol can be performed by a trusted third party or by client entities. If the protocol is designed to be scalable, it is not possible for the trusted third party to verify the quality of each exchanged multimedia file. Trusted third parties may not have human inspection functions. For this reason, the quality control mechanism of fair multimedia exchange protocols must be set in motion by the client entities; each client must decide for itself whether the claimed goods are consistent with the actual goods.

In this section, first, assumptions are presented. Second, a preliminary process, which is the offline certification process, is presented. Lastly, the protocol is described.

5.1 Assumptions

The assumptions for the proposed protocol include the following:

- There are two players in the protocol: Alice and Bob.
- The public keys of Alice and Bob are securely exchanged before the protocol runs.
- Before the protocol starts, both Alice and Bob have already decided on what multimedia files to exchange.
- The integrity and authentication of each message is provided by appending the digital signature of that message. For the sake of simplicity, these signatures are not shown in the protocol.
- Encryptions are strong enough so that it is not possible to decrypt a message without the correct decryption key.
- An attacker may gain complete control of the communications between the entities involved in a transaction. In other words, the attacker may prevent messages to be sent for an indefinite period.
- Communication failures are considered as misbehavior of an entity.
- If for any reason the communication between the entities involved is disrupted, it is assumed that an entity is cheating. The protocol prevents violation of the fairness property.

5.2 Oblivious Transfer Protocol

The fair multimedia exchange protocol is based on the oblivious transfer protocol described in [12]. In the oblivious transfer protocol, an entity A sends two messages to another entity B. However, only one of the messages that entity A sends will be received by entity B. Furthermore, entity A will not know which one of these messages are received by entity B. An oblivious transfer protocol is similar to flipping a coin and showing the result to another person without learning what the outcome was. Below, an oblivious transfer protocol has been described.

Figure 13 shows steps involved in the oblivious transfer protocol. There are two players of the oblivious transfer protocol: Alice and Bob. Firstly, Alice generates two public/private key pairs KU1/KR1 and KU2/KR2 and sends the public keys KU1 and KU2 to Bob. Bob receiving the public keys chooses one of them, say KU1 and creates a symmetric key K. Subsequently, Bob encrypts the symmetric key K with the chosen public key KU1 and sends the result of encryption $E_{KU1}(K)$ back to Alice. Alice receiving this encryption does not know which one of her public keys was used for this encryption. Alice decrypts Bob's keys twice with both of her private keys and obtains two keys. The result of one of these decryptions is the symmetric key K created by Bob: $D_{KR1}(E_{KU1}(K)) = K$ and the other decryption is gibberish data that looks like a symmetric key: $D_{KR2}(E_{KU1}(K))$ = K_Wrong. Alice does not know which one is the original symmetric key created by Bob. Alice creates two messages msg1 and msg2. Subsequently, Alice encrypts these two messages with the symmetric keys K and K_Wrong in the following way: E_K(msg1) and E_{K_Wrong}(msg2) and sends them to Bob. Bob decrypts these messages with his symmetric key K as follows: $D_K(E_K(\text{msg1}))$ = msg1 and $D_K(E_{K\_Wrong}(\text{msg2}))$ = msg_Wrong. Bob can read only one of these messages since the other cannot be decrypted correctly. Moreover, Alice does not know which message Bob can read.

The only way that Alice could cheat is to create both messages msg1 and msg2 equal (msg1 = msg2). This way Alice would know what message Bob has obtained, since both messages are the same. In order to prevent this situation, at the end of the protocol Alice must give her private keys KR1 and KR2 to Bob. Since Bob wants to discover the contents of msg1 and msg2 and since these two messages are contained in E_K(msg1) and E_{K_Wrong}(msg2), Bob must find the symmetric key K and K_Wrong. In order to find K and K_Wrong, Bob must perform the following decryptions, which require the private keys KR1 and KR2: $D_{KR1}(E_{KU1}(K)) = K$ and $D_{KR2}(E_{KU1}(K)) = K\_Wrong$.

1. A➔B: KU1 || KU2
2. B➔A: E_{KU1}(K)
3. A➔B: E_K(msg1) || E_{K_Wrong}(msg2) || KR1 || KR2

Symbol	**Meaning**
A	Alice
B	Bob
A➔B: X	A sends X to B
KU	Public-Key
KR	Private-Key
||	Concatenation Operation
E_K(X)	Encryption of X with key K

Figure 13. Oblivious Transfer Protocol

5.3 The Protocol Description

Two entities Alice and Bob want to exchange their multimedia files. Alice has file A, and Bob has file B. In step one, both Alice and Bob divides their multimedia files into n pieces grouped as in Figure 14:

1. Piece 1	Piece 2
2. Piece 3	Piece 4
. . .	. . .
. . .	. . .
. . .	. . .
n/2. Piece n-1	Piece n

Figure 14. File Division

In step two, both Alice and Bob creates n symmetric keys grouped in pairs as in Figure 15.

1. K1	K2
2. K3	K4
. . .	. . .
. . .	. . .
. . .	. . .
n/2. Kn-1	Kn

Figure 15. Creation and Grouping of n symmetric keys

In step three, both Alice and Bob encrypts their n pieces of files with their n symmetric keys as in Figure 16:

1.	E_{K1}(Piece 1)	E_{K2}(Piece 2)
2.	E_{K3}(Piece 3)	E_{K4}(Piece 4)
	. . .	. . .
	. . .	. . .
	. . .	. . .
n/2.	E_{Kn-1}(Piece n-1)	E_{Kn}(Piece n)

Figure 16. Encryption of n Pieces of Files with n Symmetric Keys

In step four, both Alice and Bob sends each other their n encrypted messages. The integrity and authentication of each message of the protocol is provided by appending the digital signature of that message. For the sake of simplicity, these signatures are not shown in the protocol.

In step five, Alice and Bob send each other their symmetric key pairs using the oblivious transfer protocol. Among each key pair, only one will be received by both entities. Therefore, in total n/2 keys will be received by each side. Figure 17 shows an example of the first phase of this step in which Alice sends K1 and K2 through the Oblivious Transfer Protocol to Bob who receives K1. Subsequently, Bob sends K9 and K10 through the protocol in the same manner to Alice who receives K10.

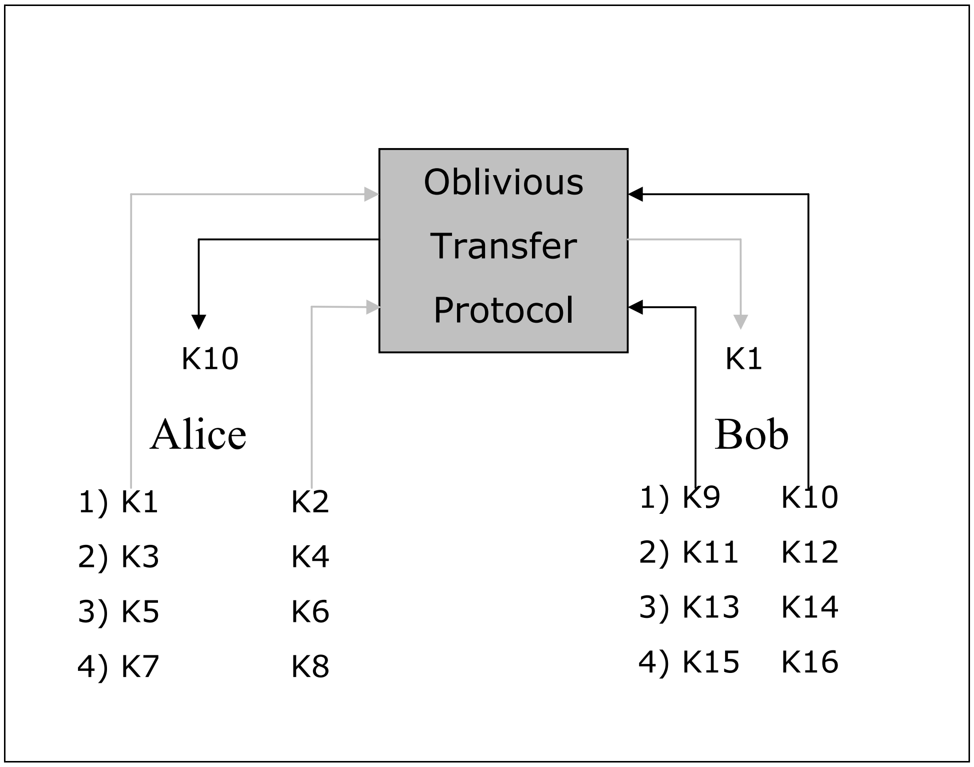


Figure 17. Transmission of Keys Using Oblivious Transfer

In step six, Alice and Bob decrypts the pieces they can. There will be n/2 decrypted pieces at each party. Each party will investigate the decrypted pieces and will decide whether these pieces are truly sub-parts of multimedia files A and B. If so, in step seven, both Alice and Bob will send each other the first bits of all n symmetric keys. Due to the oblivious transfer, n/2 bits of these bits have already been received; therefore, each party can check whether the other side is trying to cheat. Since both sides do not know which keys the other side has obtained among each pair of keys, the probability of cheating by sending wrong bits is $1/2^{(n/2)}$. Therefore, the number of pairs has a direct relation with the probability that an entity cheats. The more pairs exist, the lower probability for an entity to cheat. Subsequently, Alice and Bob send the second bits of n symmetric keys, and this continues until all bits of all keys are sent. These bits sent can be encrypted if Alice and Bob decide to do so.

Lastly, both Alice and Bob decrypt the remaining n/2 pieces and obtain the complete multimedia file.

6 IMPLEMENTATION ISSUES

The presented protocols are implemented with C# programming language using Microsoft Visual Studio .Net 2003 [35]. The implementation is tested on an Intel Celeron 1333 MHz computer with 240 MB RAM.

6.1 Fair E-Commerce Protocol

The implementation consists of three modules: Client, Merchant, and Third Party. In the following sections, cryptographic methods, system requirements, deployment of system, and performance measurements of the implementation are described.

6.1.1 Cryptographic Methods

The part below describes some important cryptographic methods deployed such as random symmetric key generator, initialization vector generator, rijndael encrypt/decrypt, rsa sign/verify, hash, HMAC and get key from key chain.

6.1.1.1 Random Symmetric Key Generator

This method generates 128-bit random symmetric keys to be used for HMAC and Rijndael Encryption/Decryption operations.

6.1.1.2 Initialization Vector Generator

This method generates random 128-bit initialization vectors for Rijndael Encryption/Decryption Operations.

6.1.1.3 Rijndael Encrypt/Decrypt

The Encrypt method performs Rijndael encryption. This method gets an input file name (name of the file to be encrypted), an output file name (encrypted file name), an initialization vector and a rijndael key as parameter.

The Decrypt method performs Rijndael decryption. This method gets an input file name (encrypted file name), an output file name (decrypted file name), an initialization vector and a rijndael key as parameter.

6.1.1.4 RSA Sign/Verify

The method RSA Sign generates RSA signatures using SHA1 for 160-bit message digest generation. This method gets a private-key and the message to be signed as parameter. Furthermore, the signature is returned.

The method RSA Verify verifies RSA signatures using SHA1 for 160-bit message digest generation. This method gets a public-key, a message and signature of a message as parameter. Furthermore, this method returns true if the signature is verified, false if the signature is not verified.

6.1.1.5 Hash

This method performs 128-bit md5 message digests. As parameter, this method gets the name of the file to be hashed and returns the 128-bit digest.

6.1.1.6 HMAC

This method performs 160-bit hash based message authentication code using SHA1 as a one-way hash function. This method gets a message, and a symmetric key as parameter. Furthermore, this method returns the 160-bit message authentication code.

6.1.1.7 Get Key from Chain Key

This method gets a key from a chain key at a given index position according to the method described in Section 4.3. This method gets a RootKey, an HMAC key and an index value as parameter. Furthermore, this method returns the computed key from the chain key.

6.1.2 Requirements

In order to run the programs, it is enough to have Microsoft .Net Framework 1.1 [36] and Microsoft SQL Server 2000 [37]. The Merchant and Third Party programs must be run on a fast computer since these servers must be able to operate for a large number of clients and have database connectivity with a Microsoft SQL Server. The Client program does not require Microsoft SQL Server.

The merchant program connects to a Microsoft SQL Server table: Merchant. Table 3 shows the fields, descriptions and data types used in the merchant table.

Table 3. Merchant Table

Field Name	**Description**	**Data Type**
PID	Product identifier	Unique identifier
Name	Product File Name	VarChar
Description	Description of the product	VarChar
Price	Price of the product	Numeric
Location	Product File Location	VarChar
LastUsedIndex	Last used chain key index	Numeric
NoCertificates	Total number of certificates	Numeric

The third party program connects to a Microsoft SQL Server table; TP. Table 4 shows the fields, descriptions, and data types used in the TP table.

Table 4. TP (Third Party) Table

Field Name	Description	Data Type
PID	Product identifier	Unique identifier
Name	Product File Name	VarChar
Description	Description of the product	VarChar
Price	Price of the product	Numeric

6.1.3 Deployment of the System

Implementation software is available at the project web page [30]. In order to use the project, the Client, Merchant, and Third Party programs must be downloaded. Microsoft .Net Framework 1.1 must be installed. Furthermore, Microsoft SQL Server must be installed on the computers that run Merchant and Third Party programs. Initially, the Trusted Third party program must be run in order to certify a product. Subsequently, the Merchant program must be run in order to register the certificate obtained in the previous step. Finally, the Client program may be run in order to start e-commerce transactions. Client, Merchant, and Third Party programs all have a practical and easy use GUI.

In order to run the client program, firstly, some setting parameters such as merchant IP, merchant port, merchant search query port, merchant public-key filename, trusted third party IP, trusted third party dispute resolution port and trusted third party public key filename must be provided. These settings can be loaded either from a settings file or by manually typing them. After the settings are changed, the client program must be restarted for these new settings.

In order to run the merchant program, some setting parameters such as the third party public-key filename, SQL data source, SQL initial catalog, SQL user name, SQL password, SMTP BCC, SMTP from, SMTP server address, SMTP server port, SMTP username, SMTP password, merchant port, merchant dispute resolution port, merchant search query port, merchant administrator e-mail address, merchant public-key filename, merchant private-key filename and IsTokenAlwaysValid field must be provided. These settings can be loaded either from a settings file or by manually typing them. After these settings are changed, the merchant program must be restarted for these new settings.

In order to run the third-party program, some setting parameters such as third-party private key filename, third-party public key filename, third party server port, SQL data source, SQL initial catalog, SQL user name, SQL password, merchant public-key filename, merchant dispute resolution port and merchant IP must be provided. These settings can be loaded either from a settings file or by manually typing them. After the settings are changed, the third-party program must be restarted for these new settings.

After completing the steps mentioned above, the system is ready to commence e-commerce transactions.

6.1.4 Performance Issues

The implementation is tested on an Intel Celeron 1333 MHz computer with 240 MB RAM. Tables 5, 6 and 7 show the cryptographic operation count for the client, merchant, and third party respectively per protocol run. In Table 6 and Table 5, i is the chain key index. The trusted third party's dispute resolution phase takes 0,3 seconds on average of ten runs of the protocol. Table 8 shows the cryptographic operation count of the third party's certification process per product. In Table 8, n is the number of requested certificates. Certification times for several files are shown in Table 9. Each of the results obtained in Table 9 is average of 10 runs of the certification process over certain files. The trusted third party has to save 526 bytes per product. The time of the third party's dispute resolution phase and the time to certify a product consists of cryptographic operations, file input/output operations, time to read/write from/to a remote Microsoft SQL Server 2000 Table and network socket operations (read/write). Note that, since the e-goods are large, most of the time is spent on file input/output operations.

Table 5. Client Cryptographic Operation Count

	Normal Protocol Run	**Dispute Resolution Run**
RSA Signature Generation	2	1
RSA Signature Verification	3	1
MD5 Hash	1	0
Rijndael Decryption	1	1
RSA Decryption	1	1

Table 6. Merchant Cryptographic Operation Count

	Normal Protocol Run	**Dispute Resolution Run**
RSA Signature Generation	2	0
RSA Signature Verification	2	1
SHA1 HMAC Generation	i-1	0
MD5 Hash	i-1	0
RSA Encryption	1	0

Table 7. Third Party Cryptographic Operation Count

	Dispute Resolution Run
RSA Signature Generation	1
RSA Signature Verification	1
SHA1 HMAC Generation	i-1
MD5 Hash	i-1
RSA Encryption	1

Table 8. Third Party Certification Cryptographic Operation Count

Operation	**Count**
RSA Signature Generation	n
SHA1 HMAC Generation	n-1
MD5 Hash	n
Rijndael Encryption	n

Table 9. Third Party Certification Times

File Size (Mega Bytes)	**I/O Read Write Time (Seconds)**	**Certificate Production time (Seconds)**	**Total Time (Seconds)**
716	277.97	185.63	463.6
470	223.3	126.94	350.24
250	122.21	77.41	199.62
157	79.21	55.79	135

Table 10 and 11 shows comparison of several optimistic protocols to the proposed e-commerce protocol for cryptographic operations that occurs during online transactions. Note that in Table 10 and 11 the symbol [*] represents the proposed e-commerce protocol. Table 10 shows cryptographic operation count in case of no disputes and table 11 shows cryptographic operation count in case of disputes. In [4 and 7] the authors omitted authentication and integrity of each message for sake of simplicity. For this reason, in table 10, the cryptographic operation count due to message authentication and integrity of the proposed e-commerce protocol is also omitted. Furthermore, all assumptions on digital envelopes [1, 12] (See Section 2.3.3 for details) are taken in to consideration while calculating cryptographic operation count if applicable. In table 10 and 11 i is the chain key index (see Section 4.3 for details). Note that in table 11 the column [7] has different outcomes of asymmetric encryption/decryption operation count because of the different types of disputes (see [7] for details).

Table 10. Comparison of Several Protocols for Cryptographic Operation Count (No Disputes)

	[*]	**[4]**	**[7]**
Symmetric Enc/Dec	1	3	1
Asymmetric Enc/Dec	2	5	13
Hash	i	4	4
MAC	i -1	0	0

Table 11. Comparison of Several Protocols for Cryptographic Operation Count (Disputes)

	[*]	**[4]**	**[7]**
Symmetric Enc/Dec	2	4	1
Asymmetric Enc/Dec	4	7	12 OR 13 OR 15
Hash	2i -1	5	4
MAC	2i -2	0	0

6.2 Fair Multimedia Exchange Protocol

During the development of the Fair Multimedia Exchange Protocol, the initial plan had been to use AVI [39] formatted movie files for multimedia files. The reason for this choice was the popularity of this format and that it yields highly compressed audio and video streams, which are of high quality. The Fair Multimedia Exchange Protocol required that AVI formatted movie files be programmatically divided (see Chapter 5 for details) in to smaller pieces on the end of one entity while the same movie pieces were programmatically concatenated by another entity on the other end. However, problems arose during the development of concatenation; the ready-made AVI methods failed to function. For this reason, instead of using AVI formatted movie files text files that represent AVI files have been deployed. The aim of developing the Fair Multimedia Exchange Protocol was to calculate the cost of the protocol; therefore using text files instead of AVI formatted movie files has no negative effect on the results.

In the following sections, cryptographic methods, system requirements, deployment of system, and performance measurements of the implementation are described.

6.2.1 Cryptographic Methods

The part below describes some important cryptographic methods deployed such as random symmetric key generator, initialization vector generator, rijndael encrypt/decrypt, rsa sign/verify and create public/private keys.

6.2.1.1 Random Symmetric Key Generator

This method generates 128-bit random symmetric keys to be used for HMAC and Rijndael Encryption/Decryption operations.

6.2.1.2 Initialization Vector Generator

This method generates random 128-bit initialization vectors for Rijndael Encryption/Decryption Operations.

6.2.1.3 Rijndael Encrypt/Decrypt

The Encrypt method performs Rijndael encryption. This method gets an input file name (name of the file to be encrypted), an output file name (encrypted file name), an initialization vector and a rijndael key as parameter.

The Decrypt method performs Rijndael decryption. This method gets an input file name (encrypted file name), an output file name (decrypted file name), an initialization vector and a rijndael key as parameter.

6.2.1.4 RSA Sign/Verify

The method RSA Sign generates RSA signatures using SHA1 for 160-bit message digest generation. This method gets a private-key and the message to be signed as parameter. Furthermore, the signature is returned.

The method RSA Verify verifies RSA signatures using SHA1 for 160-bit message digest generation. This method gets a public-key, a message and signature of a message as parameter. Furthermore, this method returns true if the signature is verified, false if the signature is not verified.

6.2.1.5 Create Public/Private Keys

This method generates random RSA public and private keys and stores them in separate files.

6.2.2 Requirements

In order to run the programs, it is enough to have Microsoft .Net Framework 1.1 [36].

6.2.3 Deployment of the System

Implementation software is available at the project web page [30]. In order to use the project the Fair Exchange program must be downloaded. Microsoft .Net Framework 1.1 must be installed.

In order to run the Fair Exchange program, firstly, some setting parameters such as remote computers IP number, remote computer port number, public-key filename, private-key filename, remote computer public-key filename and the location of the text file to be exchanged (text file represents a multimedia file) must be provided. These settings can be loaded either form a settings file or by manually typing them. After the settings are provided the server, which listens for connections, may be started. In order to do so, the *Apply Settings* button must be pressed. In a similar way the client program, which initiates the protocol, may be started by pressing the *Send* button. A confirmation dialog box for quality control appears on each exchange (for both client and server). At this step, half of the expected text files are downloaded. The user must investigate these text files for quality control. If the user confirms the quality the rest of the text files are downloaded. If not the protocol is terminated without violation of the fairness property.

In order to create easily text files with different sizes a group box named *Create Text File* containing a text box and a button is designed. While running the program one must input the desired size (in terms of megabytes) to the text box and press the *Create Text File* button to create a text file.

6.2.4 Performance Issues

Table 12 shows cryptographic operation count of the proposed Fair Multimedia Exchange Protocol. P represents the number of pieces that the multimedia file is divided. Rijndael symmetric keys are 128 bit. As explained in Section 5.3, in the last step of the protocol, the two entities involved in the transaction exchange bits of all keys. However, this means that each entity sends 128 messages. This amount of messages creates unnecessary traffic. In order to overcome this problem, instead of transmitting bit-by-bit, the first 64 bits are transferred in groups of 8 bits, the next 16 bits are transferred in groups of 2 bits and the last 48 bits are transferred bit-by-bit. Doing so, instead of transmitting 128 messages at this step 64 messages are transmitted.

Table 12. Fair Multimedia Exchange Protocol Cryptographic Operation Count

Rijndael Encryption	**Rijndael Decryption**	**RSA Signature Verification**	**RSA Signature Generation**	**RSA Encryption**	**RSA Decryption**
P	2P	1 + 2P + 64	1 + 2P + 64	P/2	P

Figure 18 shows the time required to divide and encrypt files, which is an offline process that is performed before transmission of any network message. The values in this figure are the average of 10 runs of the protocol. As it can be seen from the figure, the time required to divide and encrypt files is not affected by the number of pieces.

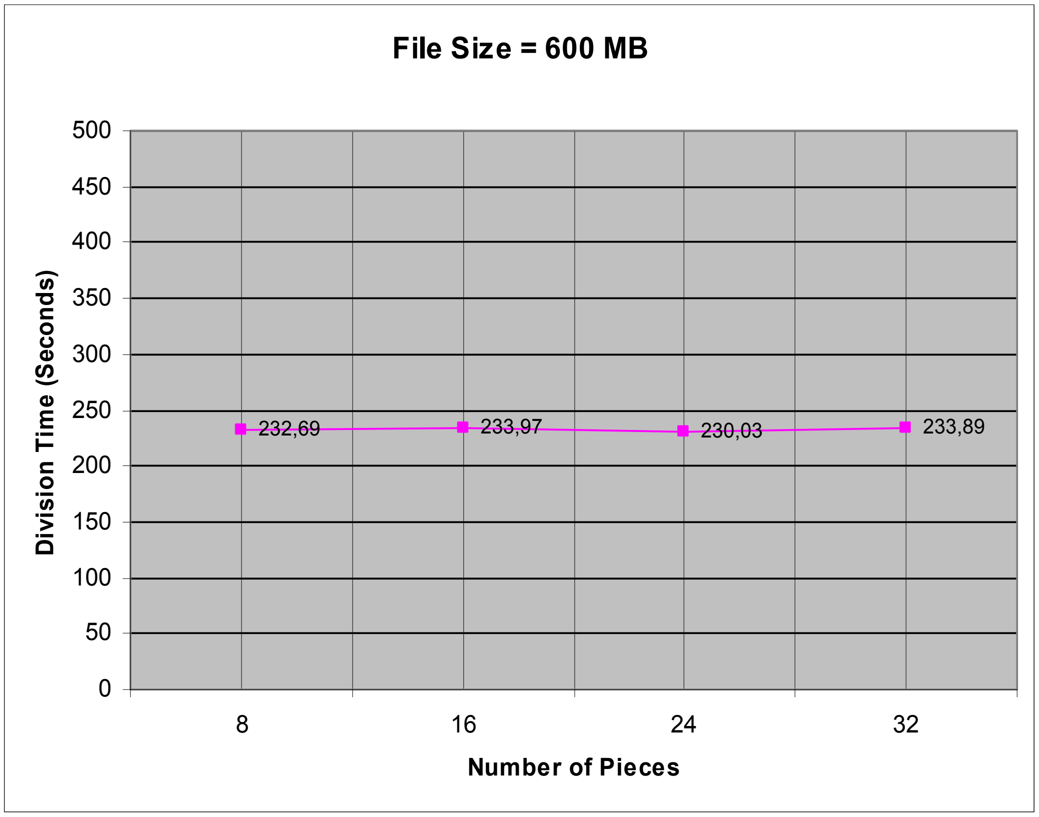


Figure 18. File Division Time

One of the most important parameters is the network buffer size. In other words, the network buffer size is the maximum amount of data to be stored in a byte array variable, which is subsequently sent through a network socket. If the network buffer is too large, the primary memory is overloaded. In Microsoft Windows Operating System [40] when the memory is overloaded, the system starts paging. In other words, the system treats the hard disk as a virtual memory and writes excessive data to secondary memory (hard disk). Although this property provides extra memory, the execution time of an operation diminishes since secondary memory operates much slower than primary memory. If the network buffer is too small, the overhead increases because the total amount of commands and controls increases. In order to investigate optimum buffer size and to calculate the overhead of the fair exchange protocol an exchange program is developed. This program has no security features and it is used for file exchange. Figure

19 shows time to perform exchange (with no security) of a 600 MB text file with varying network buffer sizes. The values in this figure are the average of 10 runs of the program. As it can be seen from the figure, the optimum network-buffer size is 4 MB. Therefore, the size of the network-buffer in the fair exchange application is fixed to 4 MB.

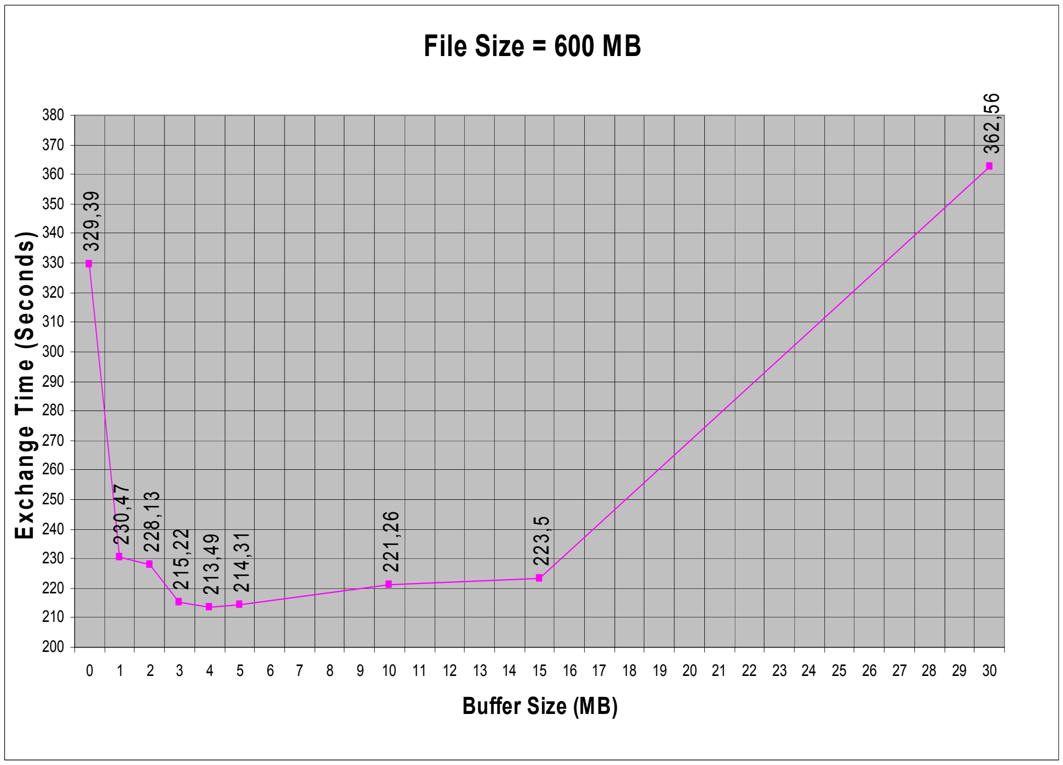


Figure 19. Exchange (No Security) Time with Varying Buffer Size

Figure 20 shows time to perform fair exchange of 600 MB files with variable number of pieces. The values in this figure are the average of 10 runs of the protocol. As expected, if the number of pieces increases the exchange becomes more secure (see Section 5 for details) however, the overhead increases and therefore the total amount of time to finish the exchange increases. The overhead increases since the number of Rijndael encryption/decryption, RSA encryption/decryption and signature generation/verification is directly proportional to the number of pieces as depicted in Table 12. Furthermore, an increase in the number of pieces augments the number of commands and controls due to the programming of the system.

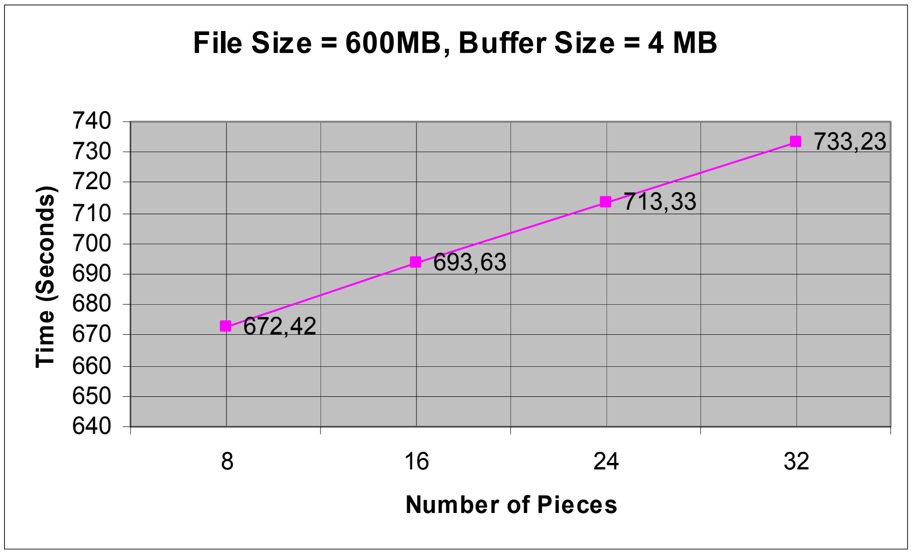


Figure 20. Fair Exchange Time (File Size = 600MB, Buffer Size = 4MB)

The overhead of the fair exchange protocol is calculated as follows. The exchange protocol, which has no security features takes minimum 213,49 seconds in order to barter 600 MB data using a buffer of size 4 MB, however the fair exchange protocol requires minimum 672,42 seconds in order to barter the same amount of data and with the same amount of buffer size. This shows that the overhead takes 458,93 seconds and that the fair exchange of an 600 MB file is exchanged approximately 3 times slower than the exchange application with no security.

Figure 21 shows an e-commerce protocol in which a client pays money in order to download a multimedia file. It is assumed that the client has selected a multimedia file to download and that using Payment Tokens (see Section 4.2 and SET in [1] for details) is a valid payment method. Furthermore, the client assumes that the merchant will not try to cheat, by ending the protocol prematurely, after he/she has been paid. In other words, the client trusts the merchant.

In the First message, the client sends a payment token, a session key encrypted with the merchant's public-key and the digital signature of the first message to the merchant. The merchant, receiving the first message, first checks the digital signature of that message and then the token for validity by contacting a bank entity. If satisfied, the merchant decrypts the encrypted session key with his/her private key and obtains the session key KS as follows: $D_{KRM}(E_{KUM}(KS)) = = KS$ where D is the decryption operation and KRM is merchant's private-key. Subsequently, in the second message, the merchant sends the multimedia file that the client has requested encrypted with the session key KS and the digital signature of the second message to the client. The client, receiving the second message, first checks the digital signature of the message; second he/she decrypts the encrypted multimedia file using the session key KS and obtains the multimedia file MMedia_File as follows: $D_{KS}(E_{KS}(\text{MMedia\_File})) = = \text{MMedia\_File}$.

1) C➔M: Token|| $E_{KUM}(KS)$ || Signature
2) M➔C : $E_{KS}(\text{MMedia\_File})$ || Signature

Symbol	Meaning
C	Client
M	Merchant
KUM	Public-key of the Merchant
KS	A symmetric session key
MMedia_File	A multimedia file
E_K(Data)	Encryption of Data with key K
Signature	Digital Signature of a message
Token	E-Money

Figure 21. Multimedia E-Commerce Protocol

The Multimedia E-Commerce Protocol, described in Figure 21, is implemented in order to compare the Fair Multimedia Exchange Protocol with a different type of protocol. It should be remembered that in the Fair Multimedia Exchange Protocol, the two entities involved do not trust each other; however, in the Multimedia E-Commerce Protocol, the client must trust the merchant. Furthermore, the architecture of the Fair Multimedia Exchange Protocol is peer-to-peer; however, the architecture of the Multimedia E-Commerce Protocol is client/server. Client-server architecture applications have the disadvantage of single point of failure. Moreover, for this architecture, the cost of maintaining a continuous operation may be large. In the implementation of the Fair Multimedia Exchange Protocol, the total amount of exchanged data is 600*2 = 1200 MB; since each entity transmits a 600 MB multimedia file to the other. However, in the Multimedia E-Commerce Protocol the total amount of exchanged data is 600 MB since only one multimedia file is transmitted by the merchant.

The implementation results show that using a file of size 600 MB takes 318,47 seconds and using a file of size 1200 MB takes 617,29 seconds on average. The best time (with 8 Pieces) of the Fair Multimedia Exchange Protocol was 672,42 seconds. Therefore, although the Fair Multimedia Exchange Protocol has the architectural and trust-relation advantages, the time required to complete the protocol is close to the time required to complete the Multimedia E-Commerce Protocol when the same amount of exchanged data is assumed. However, when assuming a 600 MB file in the Multimedia E-Commerce Protocol, this protocol takes approximately half the time required by the Fair Multimedia Exchange Protocol.

7 CONCLUSIONS AND FUTURE WORK

In this thesis, an optimistic fair e-commerce protocol for large e-goods and a fair multimedia exchange protocol have been presented.

The optimistic fair e-commerce protocol is efficient since, even in cases of disputes, the large product is transferred only once to the customer and a small number and size of messages are needed to establish the protocol. Disputes are resolved by the third party within the protocol, not by gathering evidence and taking them to a court afterwards. The third party does not have to store e-goods or protocol messages even in case of disputes. Furthermore, the client's identity is kept anonymous; no information about the customer's preferences can be gathered through the protocol. The experiment results show that the protocol requires low resource usage and therefore has good performance. Moreover, dispute resolution has low load on the trusted third party in terms of both the amount of data to be stored and the cryptographic operations to be computed. As a result, the trusted third party does not create a bottleneck in the network.

The optimistic fair e-commerce protocol has client-server architecture; however, the fair multimedia exchange protocol requires a peer-to-peer architecture. Due to this architectural difference, both protocols require different fair exchange methods and used for different applications. Client-server architecture applications have the disadvantage of single point of failure. Furthermore, for this architecture the cost of maintaining continuous operation may be large. Therefore, peer-to-peer architectures are preferred over client-server architectures. Although the multimedia exchange protocol has a architectural advantage, the experiment results show that this protocol has large overhead compared to the fair e-commerce protocol.

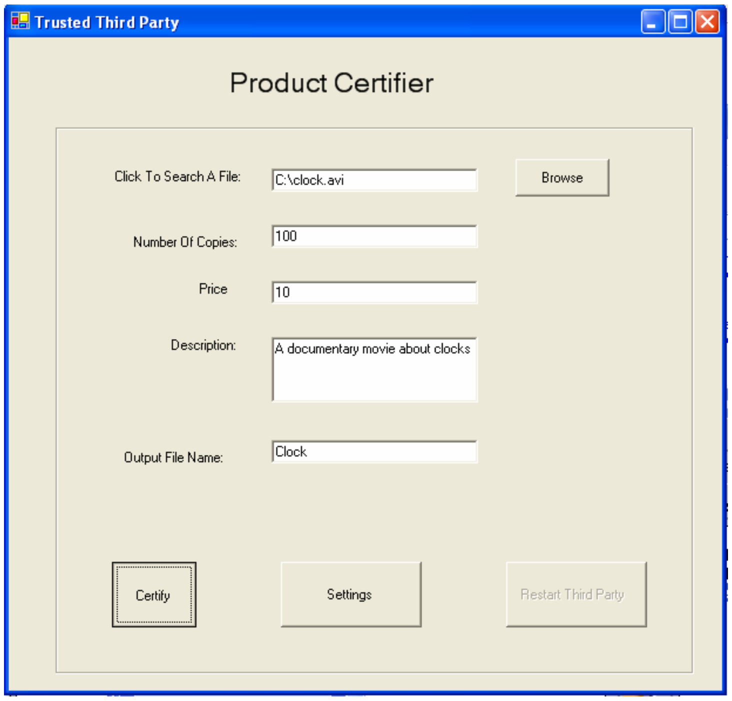
A future work that will increase the efficiency and security of the proposed optimistic fair e-commerce protocol is as follows:

In order to diminish the time of the offline certification process of the optimistic e-commerce protocol, some methods can be implemented. Instead of encrypting the entire product file, a method for corrupting the file by encrypting some bytes of the product file may be implemented.

8 APPENDIX: FORMS

8.1 Fair E-Commerce Protcol

8.1.1 Trusted Third Party Forms



Product Certifier

Click To Search A File: C:\clock.avi

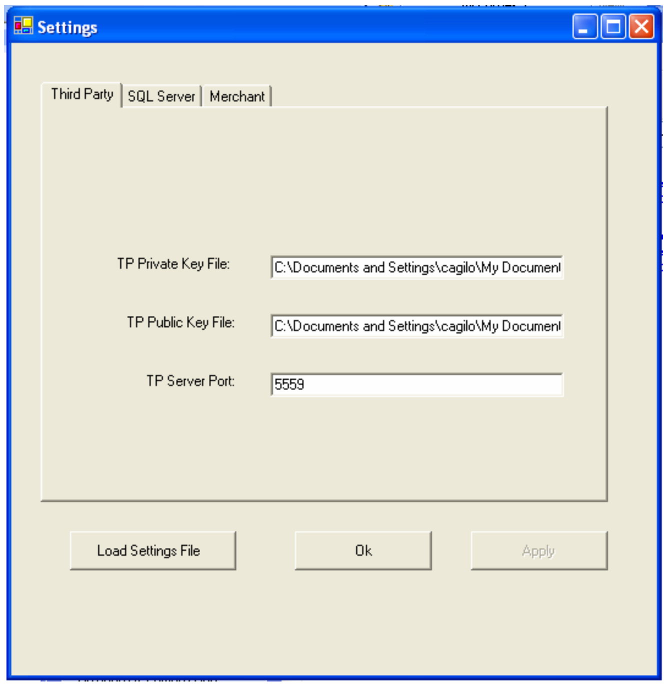
Number Of Copies: 100

Price 10

Description: A documentary movie about clocks

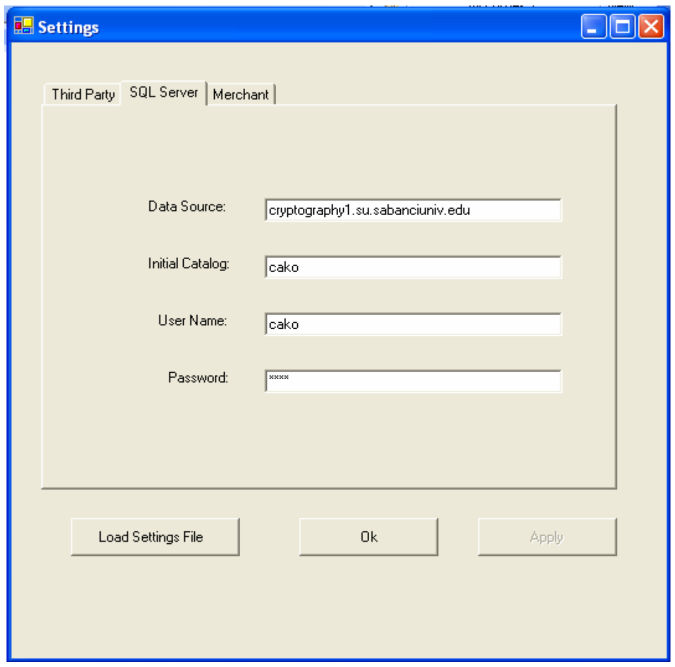
Output File Name: Clock

Figure 22. Trusted Third Party Form, Product Certification



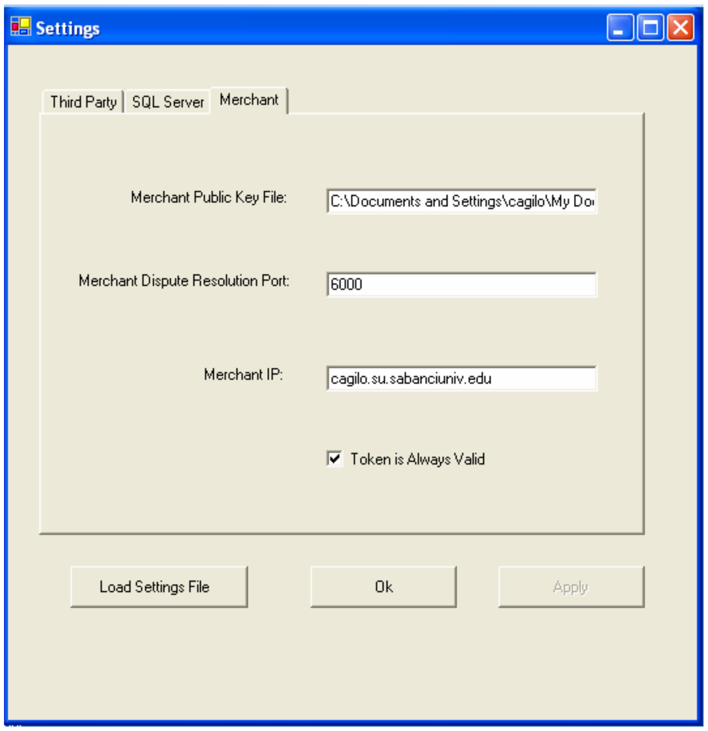
TP Private Key File: C:\Documents and Settings\cagilo\My Document
TP Public Key File: C:\Documents and Settings\cagilo\My Document
TP Server Port: 5559

Figure 23. Trusted Third Party Form, Settings Page 1



Data Source: cryptography1.su.sabanciuniv.edu
Initial Catalog: cako
User Name: cako
Password: ****

Figure 24. Trusted Third Party Form, Settings Page 2



Settings

Third Party | SQL Server | Merchant

Merchant Public Key File: C:\Documents and Settings\cagilo\My Do

Merchant Dispute Resolution Port: 6000

Merchant IP: cagilo.su.sabanciuniv.edu

☑ Token is Always Valid

Figure 25. Trusted Third Party Form, Settings Page 3

8.1.2 Merchant Forms

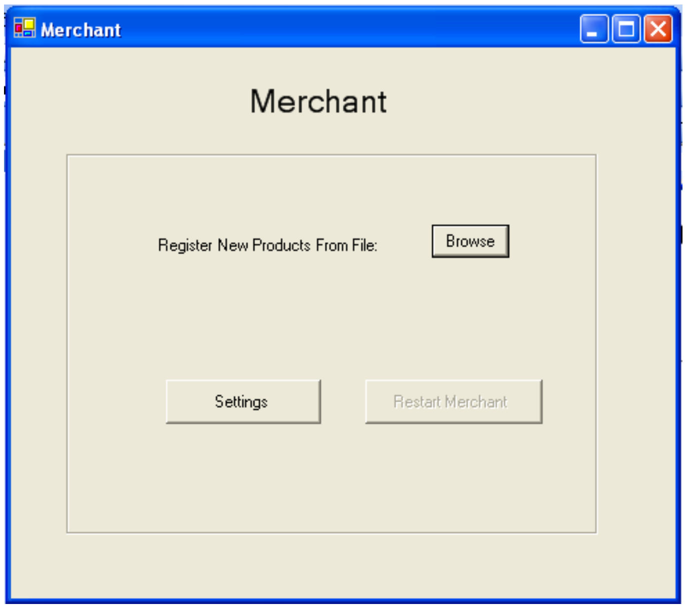
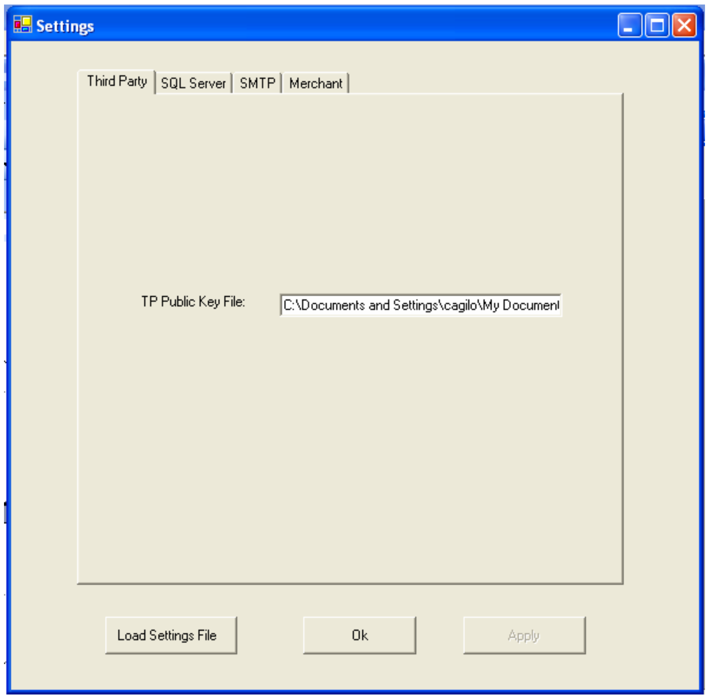
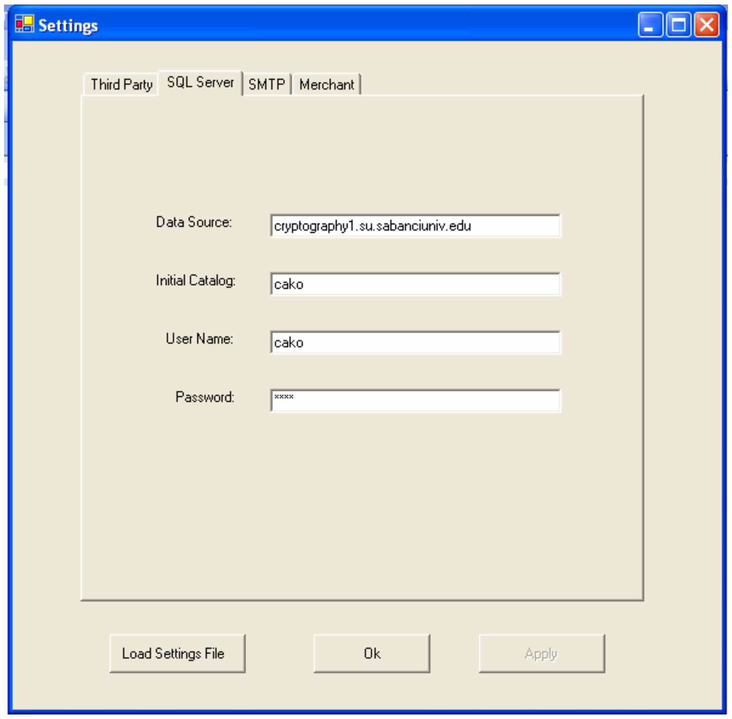


Figure 26. Merchant Form, New Certificate Registration



TP Public Key File: C:\Documents and Settings\cagilo\My Document

Figure 27. Merchant Form, Settings Page 1



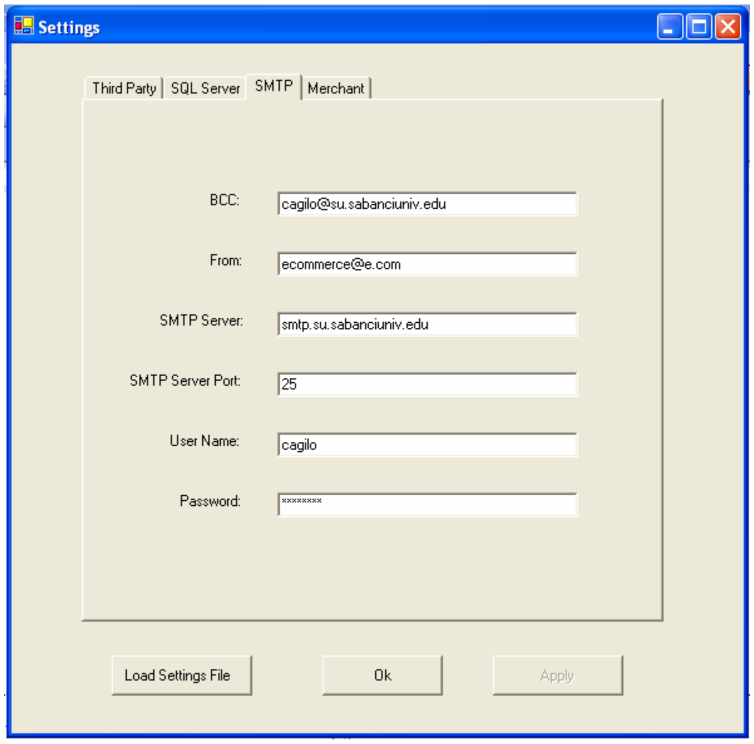
Data Source: cryptography1.su.sabanciuniv.edu

Initial Catalog: cako

User Name: cako

Password: ****

Figure 28. Merchant Form, Settings Page 2



BCC: cagilo@su.sabanciuniv.edu

From: ecommerce@e.com

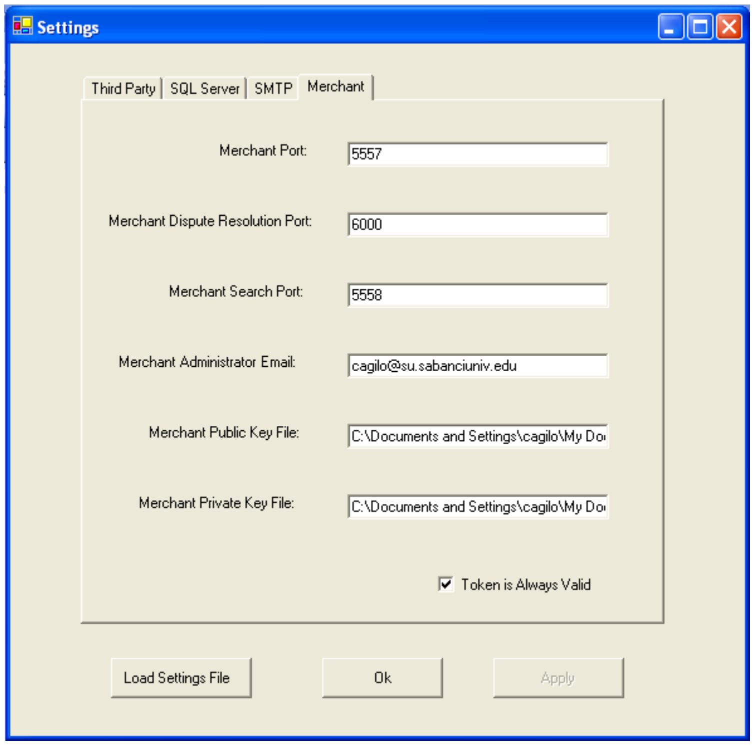
SMTP Server: smtp.su.sabanciuniv.edu

SMTP Server Port: 25

User Name: cagilo

Password: ********

Figure 29. Merchant Form, Settings Page 3



Merchant Port: 5557

Merchant Dispute Resolution Port: 6000

Merchant Search Port: 5558

Merchant Administrator Email: cagilo@su.sabanciuniv.edu

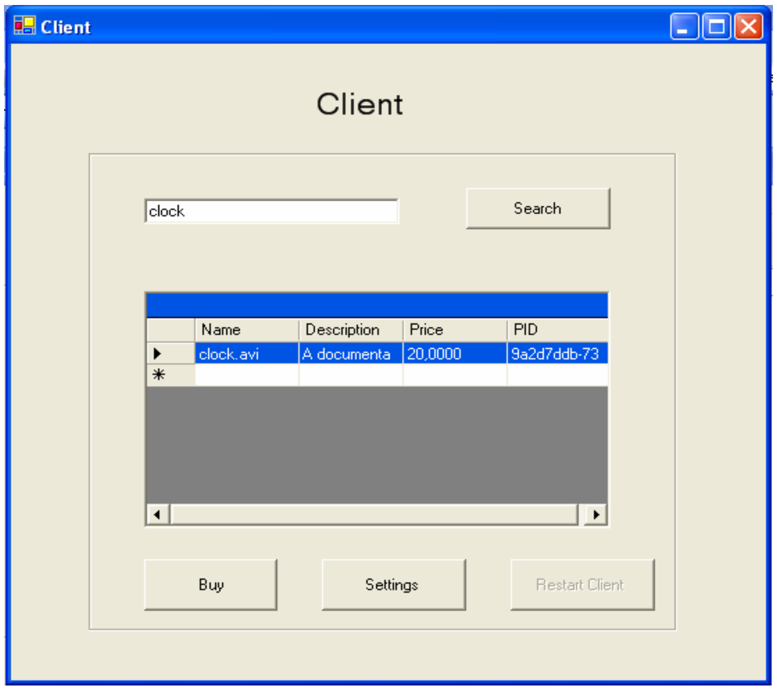
Merchant Public Key File: C:\Documents and Settings\cagilo\My Do

Merchant Private Key File: C:\Documents and Settings\cagilo\My Do

[✓] Token is Always Valid

Figure 30. Merchant Form, Settings Page 4

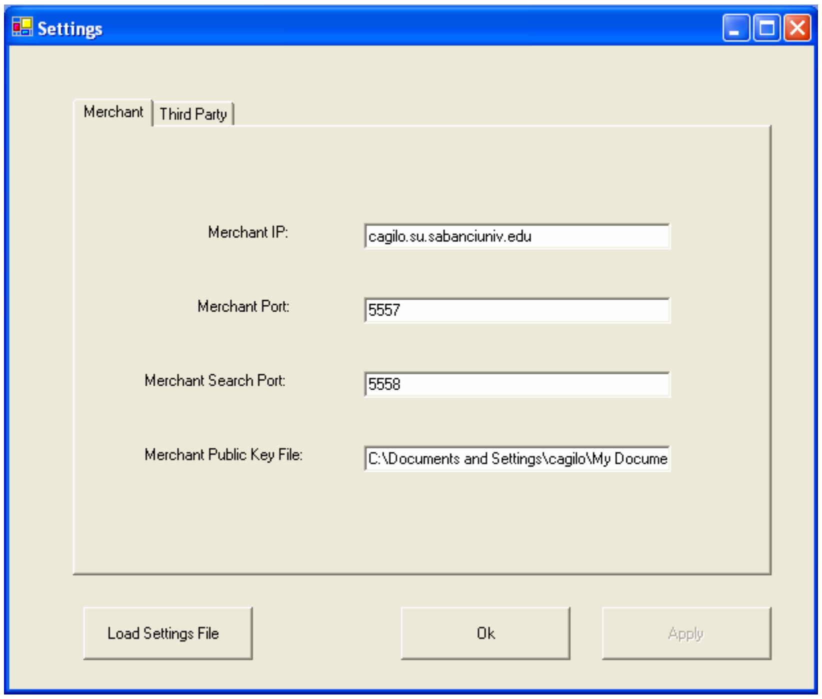
8.1.3 Client Forms



clock

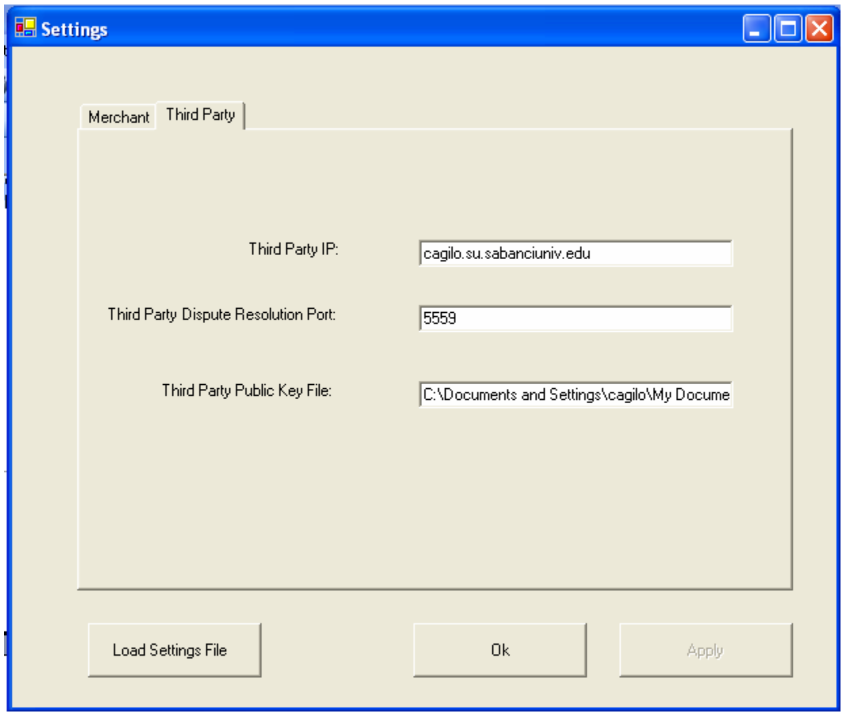
	Name	Description	Price	PID
▸	clock.avi	A documenta	20,0000	9a2d7db-73
*				

Figure 31. Client Form, Product Search and Purchase



Merchant IP:	cagilo.su.sabanciuniv.edu
Merchant Port:	5557
Merchant Search Port:	5558
Merchant Public Key File:	C:\Documents and Settings\cagilo\My Docume

Figure 32. Client Form, Settings Page 1



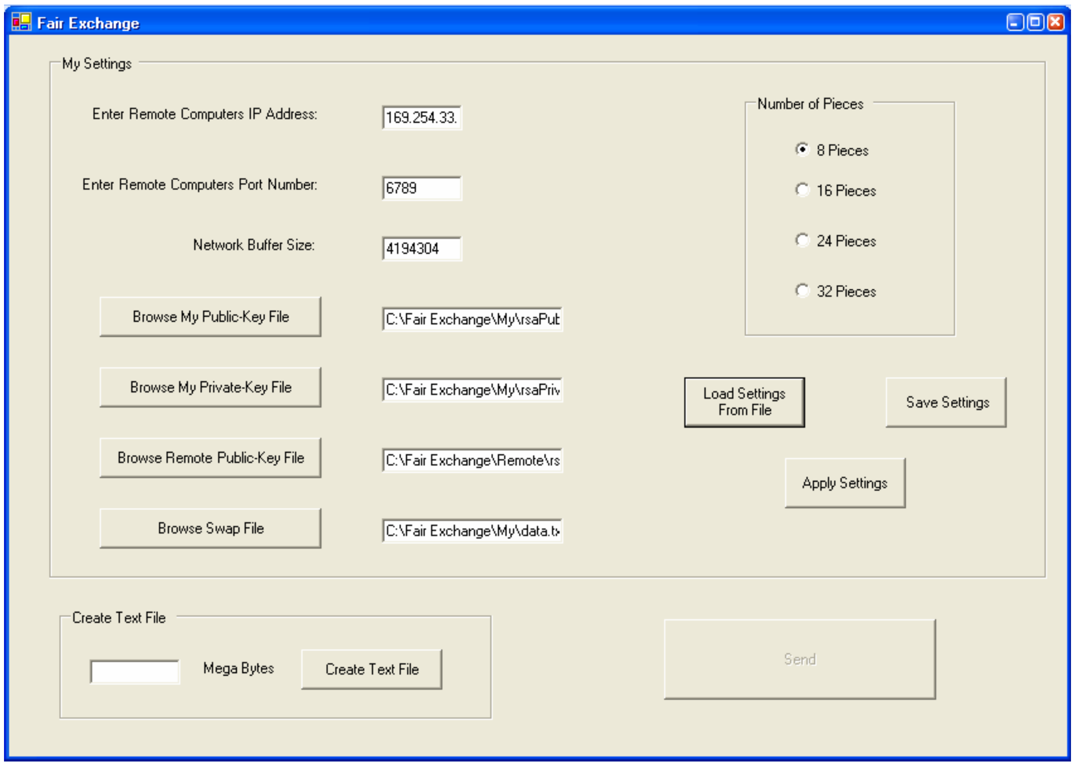
Third Party IP: cagilo.su.sabanciuniv.edu

Third Party Dispute Resolution Port: 5559

Third Party Public Key File: C:\Documents and Settings\cagilo\My Docume

Figure 33. Client Form, Settings Page 2

8.2 Fair Multimedia Exchange Protocol


Fair Exchange

My Settings

Enter Remote Computers IP Address: 169.254.33.

Enter Remote Computers Port Number: 6789

Network Buffer Size: 4194304

Browse My Public-Key File: C:\Fair Exchange\My\rsaPub

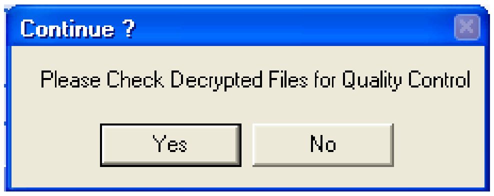
Browse My Private-Key File: C:\Fair Exchange\My\rsaPriv

Browse Remote Public-Key File: C:\Fair Exchange\Remote\rs

Browse Swap File: C:\Fair Exchange\My\data.tx

Number of Pieces: [✓ 8 Pieces] 16 Pieces 24 Pieces 32 Pieces

Figure 34. Fair Multimedia Exchange Form


Continue ?

Please Check Decrypted Files for Quality Control

Figure 35. Quality Control Confirm Dialog Box

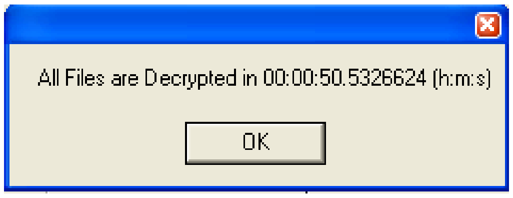
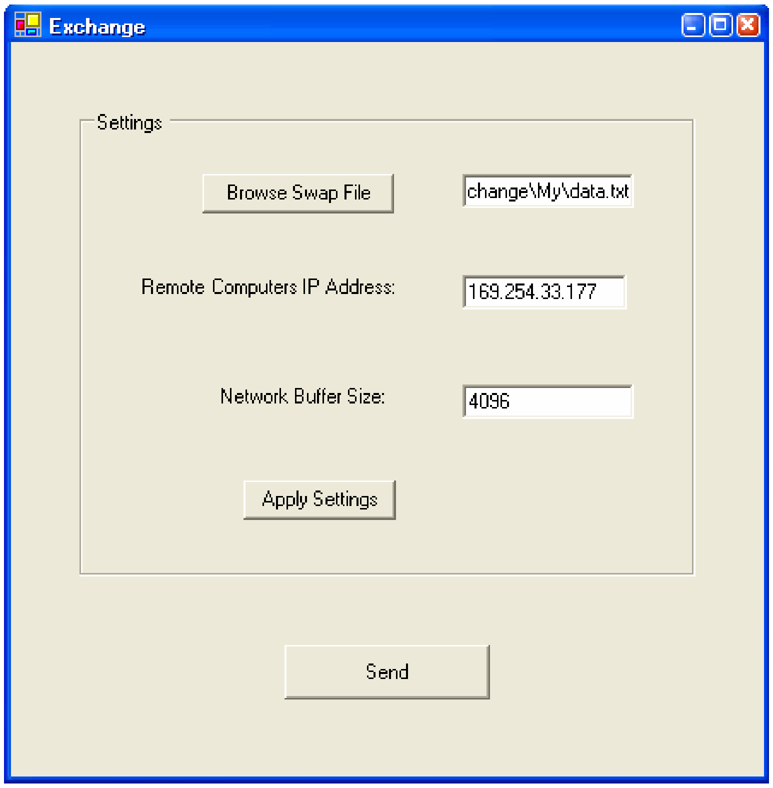

All Files are Decrypted in 00:00:50.5326624 (h:m:s)

Figure 36. Exchange Complete Declaration Dialog Box

8.3 Multimedia Exchange Program



change\My\data.txt

Remote Computers IP Address: 169.254.33.177

Network Buffer Size: 4096

Figure 37. Multimedia Exchange Form

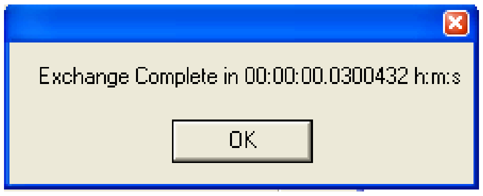
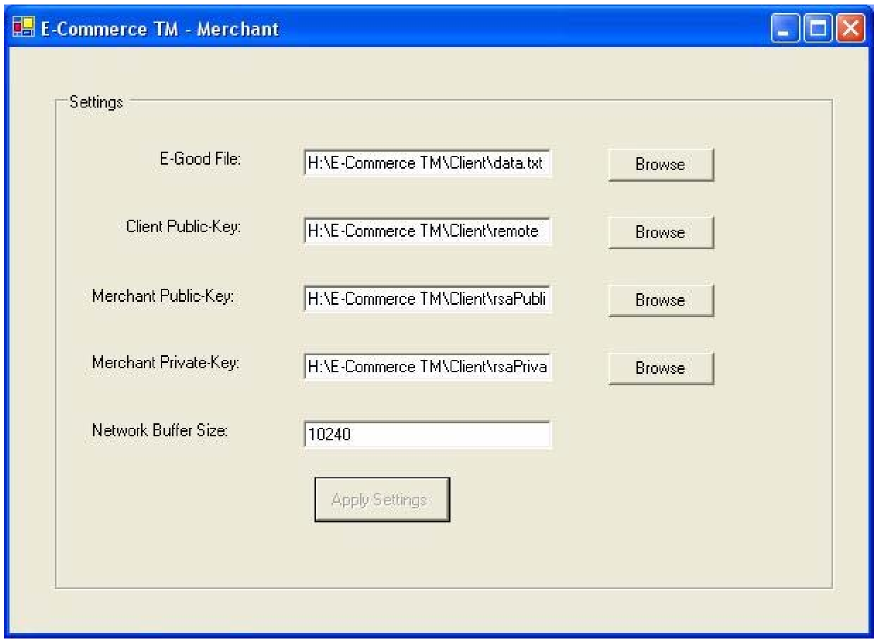

Exchange Complete in 00:00:00.0300432 h:m:s

Figure 38. Exchange Complete Declaration Dialog Box

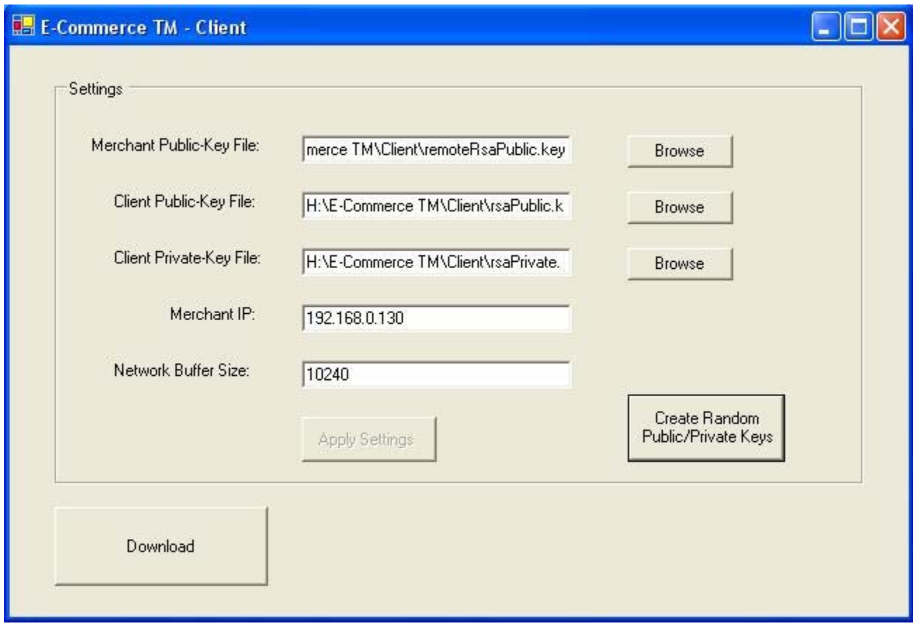
8.4 Multimedia E-Commerce Protocol


E-Commerce TM - Merchant

Settings

E-Good File:	H:\E-Commerce TM\Client\data.txt
Client Public-Key:	H:\E-Commerce TM\Client\remote
Merchant Public-Key:	H:\E-Commerce TM\Client\rsaPubli
Merchant Private-Key:	H:\E-Commerce TM\Client\rsaPriva
Network Buffer Size:	10240

Figure 39. Multimedia E-Commerce Merchant Form


E-Commerce TM - Client

Settings

Merchant Public-Key File:	merce TM\Client\remoteRsaPublic.key
Client Public-Key File:	H:\E-Commerce TM\Client\rsaPublic.k
Client Private-Key File:	H:\E-Commerce TM\Client\rsaPrivate.
Merchant IP:	192.168.0.130
Network Buffer Size:	10240

Figure 40. Multimedia E-Commerce Client Form

REFERENCES

1. William Stallings, “Cryptography and Network Security Principles and Practices”, Third Edition, Prentice Hall, 2003.

2. M. Ben-Or, O. Goldreich, S. Micali, and R.L. Rivest, “A Fair Protocol for Signing Contracts”, IEEE Transactions on Information Theory, v. 36, n.1, Jan 1990, pp.40-46.

3. M. K. Franklin and M. K. Reiter, Fair exchange with a semi-trusted third party", 4th ACM Conference on Computer and Communications Security, pages 1-5, 1997.

4. Silvio Micali,”Simple and Fast Optimistic Protocols for Fair Electronic Exchange”, Annual ACM Symposium on Principles of Distributed Computing, 2003, pp. 12- 19.

5. C.P. Pfleeger, Security in Computing, Englewood Cliffs, N.J.: Prentice-Hall, 1989.

6. Indrakshi Ray and Indrajit Ray, "Fair Exchange in E-commerce", ACM SIGEcomm Exchange.Accepted for publication, Sep 2001

7. Indrakshi Ray and Indrajit Ray, "An Optimistic Fair-exchange E-commerce Protocol with Automated Dispute Resolution", Proceedings of the First International Conference on Electronic Commerce and Web Technologies, Greenwich, UK, September 2000.

8. Indrakshi Ray and Indrajit Ray, ”An Anonymous Fair-exchange E-commerce Protocol”, In Proceedings of the 1st International Workshop on Internet Computing and E-Commmerce, San Francisco, CA., 2001

9. Indrakshi Ray and Indrajit Ray, “A Fair-Exchange Protocol with Automated Dispute Resolution”, In Proceedings of the 14th Annual IFIP WG 11.3 Working Conference on Database Security. Schoorl, The Netherlands, 2000

10. Indrakshi Ray and Indrajit Ray, " Failure Analysis of an E-commerce Protocol Using Model Checking", Proceedings of the Second International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, Milpitas, CA, June 2000.

11. Indrakshi Ray and Indrajit Ray, " Failure Analysis of an E-commerce Protocol Using Model Checking Extended Paper", Proceedings of the Second International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, Milpitas, CA, June 2000.

12. Bruce Schneier, “Applied Cryptography”, 1996.

13. S. Even, O. Goldreich, and A.Lempel, “A Randomizing Protocol for Signing Contracts,” Communications of the ACM, v.28, n.6, Jun 1985, pp. 637-647).

14. COX, B., TYGAR, J. D., AND SIRBU, M. 1995, “NetBill Security and Transaction Protocol”, In Proceedings of the 1st USENIX Workshop in Electronic Commerce. 77–88).

15. KETCHPEL, S. 1995.” Transaction Protection for Information Buyers and Sellers”, In Proceedings of the Dartmouth Institute for Advanced Graduate Studies ’95: Electronic Publishing and the Information Superhighway.

16. Kazaa, "KaZaA is a completely distributed peer-to-peer file sharing service", www.kazaa.com.

17. Napster,"Napster a legal music service", www.napster.com.

18. Wade Trappe, Lawrence C. Washington, "Introduction to Cryptography with Coding Theory", Prentice Hall, 2001.

19. Asokan, N., Shoup, V., and Waidner, M. 1998a. Asynchronous protocols for optimistic fair exchange, In Proceedings of the IEEE Symposium on Research in Security and Privacy (May 1998), pp. 86--99.

20. P. Chown, "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.

21. S. Teiwes, P. Hartmann, D. Kuenzi, "Use of the IDEA Encryption Algorithm in CMS", RFC 3058, February 2001.

22. R. Baldwin, R. Rivest, "The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms", RFC 2040, October 1996.

23. Trappe W., L.C. Washington, "Introduction to Cryptography with Coding Theory", 1997

24. Diffie, W., and M. E. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, vol. IT-22, no. 6, pp. 644-654, November 1976.

25. Rivest, R., A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, vol. 21, no. 2, pp. 120-126, February 1978.

26. V.S. Miller, “Use of elliptic curves in cryptography”, Advances in Cryptology - Crypto '85, Springer-Verlag, 417-426, 1986.

27. T. ElGamal, “A public-key cryptosystem and a signature scheme based on discrete logarithms”, IEEE Transactions on Information Theory 31, 469-472, 1985.

28. Wiener, M. J., “Performance Comparison of Public-Key Cryptosystems”, RSA Laboratories’ Cryptobytes, vol. 4, no. 1, pp. 1 – 5, 1998.

29. Rivest, R., “The MD5 Message Digest Algorithm”, RFC 1321, April 1992

30. Optimistic Fair E-commerce Project, http://students.sabanciuniv.edu/~cagilo/ecommerce/

31. U.S. Department of Commerce, “Computer Data Authentication”, Federal Information Processing Standards Publication 180-1,1995.

32. Akl, S. G., “Digital Signatures: A Tutorial Survey”, IEEE Computer, vol.16, no. 2, pp. 15-24, February 1983.

33. U.S. Department of Commerce, “Digital Signature Standard (DSS)”, Federal Information Processing Standards Publication 186, 1994.

34. NIST, “Digital Signature Standard (DSS)”, FIPS PUB 186-2, 2000 January 27.

35. Microsoft, ”Visual Studio .Net”, http://msdn.microsoft.com/vstudio/

36. Microsoft, “Microsoft .Net Framework”, http://msdn.microsoft.com/netframework/technologyinfo/howtoget/default.aspx

37. Microsoft, “SQL Server 2000”, http://www.microsoft.com/sql/

38. A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "SPINS: Security protocols for sensor networks," in Proceedings of Mobile Networking and Computing 2001, 2001.

39. John F. McGowan, "AVI Overview", http://www.jmcgowan.com/avi.html

40. Microsoft, "Microsoft Operating System", http://www.microsoft.com

REFERENCES

1. William Stallings, “Cryptography and Network Security Principles and Practices”, Third Edition, Prentice Hall, 2003.

2. M. Ben-Or, O. Goldreich, S. Micali, and R.L. Rivest, “A Fair Protocol for Signing Contracts”, IEEE Transactions on Information Theory, v. 36, n.1, Jan 1990, pp.40-46.

3. M. K. Franklin and M. K. Reiter, Fair exchange with a semi-trusted third party", 4th ACM Conference on Computer and Communications Security, pages 1-5, 1997.

4. Silvio Micali,”Simple and Fast Optimistic Protocols for Fair Electronic Exchange”, Annual ACM Symposium on Principles of Distributed Computing, 2003, pp. 12- 19.

5. C.P. Pfleeger, Security in Computing, Englewood Cliffs, N.J.: Prentice-Hall, 1989.

6. Indrakshi Ray and Indrajit Ray, "Fair Exchange in E-commerce", ACM SIGEcomm Exchange.Accepted for publication, Sep 2001

7. Indrakshi Ray and Indrajit Ray, "An Optimistic Fair-exchange E-commerce Protocol with Automated Dispute Resolution", Proceedings of the First International Conference on Electronic Commerce and Web Technologies, Greenwich, UK, September 2000.

8. Indrakshi Ray and Indrajit Ray, ”An Anonymous Fair-exchange E-commerce Protocol”, In Proceedings of the 1st International Workshop on Internet Computing and E-Commmerce, San Francisco, CA., 2001

9. Indrakshi Ray and Indrajit Ray, “A Fair-Exchange Protocol with Automated Dispute Resolution”, In Proceedings of the 14th Annual IFIP WG 11.3 Working Conference on Database Security. Schoorl, The Netherlands, 2000

10. Indrakshi Ray and Indrajit Ray, " Failure Analysis of an E-commerce Protocol Using Model Checking", Proceedings of the Second International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, Milpitas, CA, June 2000.

11. Indrakshi Ray and Indrajit Ray, " Failure Analysis of an E-commerce Protocol Using Model Checking Extended Paper", Proceedings of the Second International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, Milpitas, CA, June 2000.

12. Bruce Schneier, “Applied Cryptography”, 1996.

13. S. Even, O. Goldreich, and A.Lempel, “A Randomizing Protocol for Signing Contracts,” Communications of the ACM, v.28, n.6, Jun 1985, pp. 637-647).

14. COX, B., TYGAR, J. D., AND SIRBU, M. 1995, “NetBill Security and Transaction Protocol”, In Proceedings of the 1st USENIX Workshop in Electronic Commerce. 77–88).

15. KETCHPEL, S. 1995.” Transaction Protection for Information Buyers and Sellers”, In Proceedings of the Dartmouth Institute for Advanced Graduate Studies ’95: Electronic Publishing and the Information Superhighway.

16. Kazaa, “KaZaA is a completely distributed peer-to-peer file sharing service”, www.kazaa.com.

17. Napster,”Napster a legal music service”, www.napster.com.

18. Wade Trappe, Lawrence C. Washington, “Introduction to Cryptography with Coding Theory”, Prentice Hall, 2001.

19. Asokan, N., Shoup, V., and Waidner, M. 1998a. Asynchronous protocols for optimistic fair exchange, In Proceedings of the IEEE Symposium on Research in Security and Privacy (May 1998), pp. 86--99.

20. P. Chown, “Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)”, RFC 3268, June 2002.

21. S. Teiwes, P. Hartmann, D. Kuenzi, “Use of the IDEA Encryption Algorithm in CMS”, RFC 3058, February 2001.

22. R. Baldwin, R. Rivest, “The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms”, RFC 2040, October 1996.

23. Trappe W., L.C. Washington, “Introduction to Cryptography with Coding Theory”, 1997

24. Diffie, W., and M. E. Hellman, “New Directions in Cryptography”, IEEE Transactions on Information Theory, vol. IT-22, no. 6, pp. 644-654, November 1976.

25. Rivest, R., A. Shamir and L. Adleman, “A Method for Obtaining Digital Signatures and Public Key Cryptosystems”, Communications of the ACM, vol. 21, no. 2, pp. 120-126, February 1978.

26. V.S. Miller, “Use of elliptic curves in cryptography”, Advances in Cryptology - Crypto '85, Springer-Verlag, 417-426, 1986.

27. T. ElGamal, “A public-key cryptosystem and a signature scheme based on discrete logarithms”, IEEE Transactions on Information Theory 31, 469-472, 1985.

28. Wiener, M. J., “Performance Comparison of Public-Key Cryptosystems”, RSA Laboratories’ Cryptobytes, vol. 4, no. 1, pp. 1 – 5, 1998.

29. Rivest, R., “The MD5 Message Digest Algorithm”, RFC 1321, April 1992

30. Optimistic Fair E-commerce Project, http://students.sabanciuniv.edu/~cagilo/ecommerce/

31. U.S. Department of Commerce, “Computer Data Authentication”, Federal Information Processing Standards Publication 180-1,1995.

32. Akl, S. G., “Digital Signatures: A Tutorial Survey”, IEEE Computer, vol.16, no. 2, pp. 15-24, February 1983.

33. U.S. Department of Commerce, “Digital Signature Standard (DSS)”, Federal Information Processing Standards Publication 186, 1994.

34. NIST, “Digital Signature Standard (DSS)”, FIPS PUB 186-2, 2000 January 27.

35. Microsoft, ”Visual Studio .Net”, http://msdn.microsoft.com/vstudio/

36. Microsoft, “Microsoft .Net Framework”, http://msdn.microsoft.com/netframework/technologyinfo/howtoget/default.aspx

37. Microsoft, “SQL Server 2000”, http://www.microsoft.com/sql/

38. A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "SPINS: Security protocols for sensor networks," in Proceedings of Mobile Networking and Computing 2001, 2001.

39. John F. McGowan, "AVI Overview", http://www.jmcgowan.com/avi.html

40. Microsoft, "Microsoft Operating System", http://www.microsoft.com