An Optimistic Fair E-Commerce Protocol for Large E-Goods

Cagil Can Oniz, Erkay Savas, Albert Levi
Faculty of Engineering and Natural Sciences, Sabanci University
Orhanli – Tuzla, TR-349656 Istanbul, Turkey
cagilo@su.sabanciuniv.edu, {erkays, levi}@sabanciuniv.edu

***Abstract*— Suppose two entities that do not trust each other want to exchange some arbitrary data over a public channel. A fair exchange protocol ensures that both parties get what they want or neither gets anything. In this paper, a fair e-commerce protocol for large e-goods is proposed and implemented. The proposed protocol provides a method for the fair exchange of e-money for e-products, and a method for verifying the contents of the exchanged items. The protocol is optimistic and 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, the customer remains anonymous after the transaction; thus, 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 only a small number of cryptographic operations is sufficient.**

I. INTRODUCTION

The purpose of a fair exchange protocol is to barter data between two entities, as a result of which either both parties get what they want or they both get nothing. Fair exchange protocols have been given special names depending on the contents of exchanged items. Some examples are given below.

- In a *contract signing protocol* [2, 4, 5, 11, 12 and 19], digital signatures of two entities that bind them to the terms stated on a contract are exchanged.
- In a *certified mail protocol* [4, 12 and 16], an e-mail message is exchanged for a receipt of this e-mail. The receipt proves that the intended receiver of the e-mail has obtained the e-mail.
- In an *e-commerce protocol* [6, 7, 8, 9, 10, 13 and 14], payment (a token) is exchanged for an electronic good (e-good) or a service.

Four essential requirements of exchange protocols, namely fairness, quality control, client anonymity, and the number of e-good transfer are explained as follows.

1. *Fairness*: An exchange protocol is considered fair if the protocol has only two possible outcomes: either both entities obtain the items that they expect from the other entity or neither entity obtains any items.
2. *Quality Control*: An exchange protocol must allow participating entities to verify that a declared definition of an item truly defines the contents of that item.
3. *Client Anonymity*: Entities that perform the exchange may decide to remain anonymous. Anonymity ensures that the identity of an entity is not revealed during the transaction. For instance, a customer may not want a merchant to discover his/her pattern of shopping habits after performing a transaction. This is ensured through client anonymity.
4. *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 hence transferring these large items multiple times could be costly. Consequently, large items should be transferred only once per protocol run.

Exchange protocols can be examined in two categories: i) *online third party protocols* [3, 5, 9, 13 and 14] and ii) *baby-step protocols* [11 and 12]. In *online third party protocols*, the exchange is achieved via a trusted third party. Each party submits his/her own item to the trusted third party that forwards the items to the appropriate recipient entity. In *baby-step protocols* where there is no trusted third party, items are divided into smaller 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 as well. If the latter does not send its partial item in return, the algorithm terminates, leaving both parties in a situation where neither obtains the desired item. This act of swapping partial items continues until the items are completely exchanged; hence, the solution is obtained through "baby steps".

Both approaches have drawbacks [9]. *Online third party protocols* require that the third party always be online; therefore, the third party needs bandwidth to handle large amounts of traffic. The 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 expensive measures for continuous service. On the other hand, *baby-step protocols,* which have peer-to-peer architecture, may have a large overhead. In some cases, they provide a lower degree of fairness.

Proceedings of the Seventh IEEE International Symposium on Computer Networks (ISCN' 06)
1-4244-0491-6/06/$20.00 © 2006 **IEEE**

In order to overcome the problems of the *online third party protocols*, another approach, using so-called *optimistic protocol* [4, 7, 8, 9 and 16], has been proposed. In an *optimistic protocol*, 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 honestly by sending its own item, the protocol is complete since both entities obtained the items that they expected. If the initiator does not receive the expected, it 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 gain. Therefore, attempts of cheating and consequently the participation of a trusted third party would be rare. This property of keeping the trusted third party out of normal execution reduces network traffic to the third party.

In this paper, an optimistic fair e-commerce protocol for large e-goods is presented. The proposed protocol provides a method not only for fair exchange of e-money for e-goods, but also for 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; thus, no information about the customer's shopping habits can be gathered through the protocol.

The rest of the paper is structured as follows. We state our system assumptions and give some definitions and notations in Section II. In Section III, we explain some of the preliminary information required to understand the protocol description. In Section IV, we describe our e-commerce protocol. Implementation issues have been explained in Section V and we conclude the paper in Section VI.

II. Assumptions and Notation

The assumptions for the proposed protocol are as follows:

- There are three players involved: the client, the merchant, and the trusted third party.
- The client and the merchant do not trust each other but they both trust the third party.
- The public keys of the merchant and the trusted third party are publicly accessible (e.g. through the use of certificates).
- 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 Purchase Request Message in Secure Electronic Transaction (SET) [1], which is a world wide standard for payment tokens) 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 trusted bank entity in order to verify a token. 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 III.B 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.
- 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 the 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 vice 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 [15, 17, 18, 19, and 20].

Notations used in the protocol are given in Table I.

TABLE I. NOTATIONS

Symbol	Meaning
$CERT_{TP}^{i}$	i^{th} Certificate of a product signed by trusted third party
H(X)	Hash of X
E_K(data)	Encryption of data with key K
e-good	E-product or electronic item such as, database or multimedia file
Price	Price of e-good
Description	String describing contents of product
KU_X	Public key of identity X
KR_X	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

III. PRELIMINARIES

In order to understand the proposed protocol, two preliminary concepts are presented: *chain keys* and *offline e-good certification process*.

A. Chain Keys

In the proposed protocol for each purchase of a product, a separate symmetric key is needed. In order to reduce the total number of keys to be stored, the third party will generate n session keys, KS_i (i = 1… n), using a special method called *chain keys* introduced in [22].

Figure 1 shows the steps involved in producing chain keys. Firstly, the third party generates a random symmetric session key called the *Root Key* or KS_1. Secondly, the third party computes the HMAC [1, 11] of the Root Key using a key, *HMAC Key*, and obtains the second key of the chain, which is KS_2. Thirdly, 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 (i.e. KS_i (i = 1… n)) are produced. The aim of this method is to generate multiple keys derived from the Root Key and a single HMAC Key.

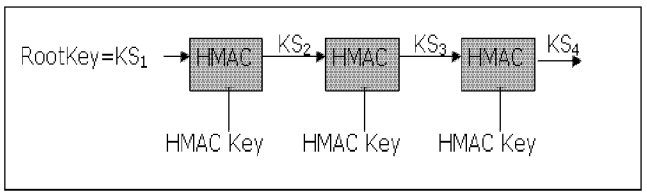

Figure 1. Production of Chain Keys

B. Offline E-Good 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 2 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 using a special method; *Chain Keys* (see Section III.A and [22] for details). In this method a chain of keys is produced using two keys: the *RootKey* and *HMAC Key*. 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 previous fields of the certificate by the trusted third party.

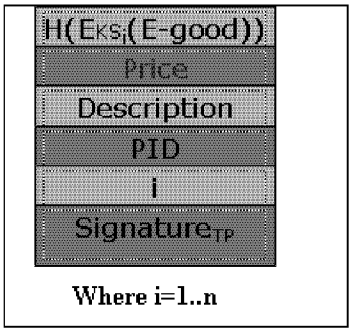

Figure 2. Product Certificate Format

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 III.A for details). At the end of this offline certification process, the trusted third party will send the *n* certificates and keys in the chain (or alternatively the Root Key and the HMAC Key) to the merchant through a secure channel.

IV. E-COMMERCE PROTOCOL DESCRIPTION

Figure 3 shows the steps involved 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 and his/her public key 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 the public-key of the customer in the third message. The customer, receiving this encrypted product decryption key, decrypts it using its private key in the following manner: $D_{KR_C}(E_{KU_C}(KS_i)) = KS_i$. Subsequently, the customer acquires the e-good by decrypting the encrypted e-good using the product decryption key KS_i in the following way: $D_{KS_i}(E_{KS_i}(\text{e-good})) = \text{e-good}$. Up to this point, the normal execution of the protocol has been described. Below the motive for dispute resolution is depicted.

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

1) M➔C: $E_{KS_i}(\text{e-good}) \| CERT_{TP}^{i}$
2) C➔M: token $\| KU_C$
3) M➔C: $E_{KU_C}(KS_i)$

Dispute Resolution Phase

Customer claims that he/she has not received Message 3

4) C➔TP: token $\| i \|$ PID $\| KU_C$
5) TP➔C: $E_{KU_C}(KS_i)$
6) TP➔M: token $\| i \|$ PID $\| KU_C$
7)

Figure 3. 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 has the 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 has been 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 he/she 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 that 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 (see Purchase Request Message in Secure Electronic Transaction (SET) [1] for details); therefore, although the token may be processed two times, the money transferred to the merchant's account will not double. Subsequently, the trusted third party checks the validity of the token. Moreover, it also checks whether the value of the token matches the price of the product. If the token is valid and sufficient to cover the price of the product, in the fifth message, the trusted third party sends the product decryption key, KS_i, 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 its private key as follows: $D_{KR_C}(E_{KU_C}(KS_i)) = KS_i$. 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, upon receiving this message, processes the token and removes the i^{th} certificate from its database.

A malicious user may try to exhaust 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 attacker's aim is to include the third party in the normal execution of the protocol and therefore to increase the traffic directed to 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 a 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. In 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 the way 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.

V. IMPLEMENTATION

The presented protocol has been implemented with C# programming language using Microsoft Visual Studio .Net 2003 [21]. The implementation has been tested on an Intel Celeron 1333 MHz computer with 240 MB RAM.

Tables II, III and IV show the cryptographic operation count for the client, merchant, and third party per protocol run, respectively. In Table III and IV, *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 II. CLIENT CRYPTOGRAPHIC OPERATION COUNT

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

TABLE III. 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 IV. 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 V shows the cryptographic operation count of the third party's certification process per product. In Table V, *n* is the number of requested certificates. Certification times for different file sizes are shown in Table VI. Each of the results obtained in Table VI is the average of 10 runs of the certification process over certain files. The trusted third party has to save only 526 bytes per product. The time of the third party's dispute resolution phase and the time to certify a product consist 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. Note also that certificate production time spent on cryptographic operations can be significantly reduced by using a state-of-the-art implementation of cryptographic library.

TABLE V. THIRD PARTY CERTIFICATION CRYPTOGRAPHIC OPERATION COUNT

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

TABLE VI. THIRD PARTY CERTIFICATION TIMES

File Size (MB)	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 VII and VIII show the comparison of several optimistic protocols with the proposed e-commerce protocol for cryptographic operations that occur during online transactions. Note that in Table VII and VIII the symbol [*] represents the proposed e-commerce protocol. Table VII shows the cryptographic operation count in case of no disputes and Table VIII illustrates the cryptographic operation count in case of disputes. In [4 and 7], the authors omitted the authentication and integrity of each message for the sake of simplicity. For this reason, in Table VII, the cryptographic operation count due to message authentication and the integrity of the proposed e-commerce protocol has also been omitted. Furthermore, all assumptions on digital envelopes [1, 11] have been taken into consideration while calculating the cryptographic operation count, if applicable. In Tables VII and VIII, *i* is the chain key index (see Section III.A for details). Note that in Table VIII, column [7] has different outcomes of the asymmetric encryption/decryption operation count because of the different types of disputes (see [7] for details).

TABLE VII. 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 VIII. 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	2*i* -1	5	4
MAC	2*i* -2	0	0

VI. CONCLUSION AND FUTURE WORK

In this paper, an optimistic fair e-commerce protocol for large e-goods has been presented. The optimistic fair e-commerce protocol is efficient since, even in case 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 experimental results show that the protocol requires low resource usage and therefore has good performance. Moreover, dispute resolution has a 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.

Future work that will increase the efficiency and security of the proposed optimistic fair e-commerce protocol can be outlined as follows: In order to speed up the offline certification process of the proposed protocol, a method for corrupting the file by encrypting some bytes of the product file may be implemented instead of encrypting the entire product file, which is the most time consuming operation in the certification process.

REFERENCES

[1] W. 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, 1997, pp. 1-5.

[4] S. 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", Prentice-Hall, Englewood Cliffs, N.J., 1989.

[6] I. Ray and I. Ray, "Fair Exchange in E-commerce", ACM SIGEcomm Exchange, September 2001.

[7] I. Ray and I. 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] I. Ray and I. Ray, "An Anonymous Fair-exchange E-commerce Protocol", In Proceedings of the 1st International Workshop on Internet Computing and E-Commerce, San Francisco, CA., 2001.

[9] I. Ray and I. 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] I. Ray and I. 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] B. Schneier, "Applied Cryptography", 1996.

[12] 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.

[13] B. Cox, J. D. Tygar, and M. Sirbu, "NetBill Security and Transaction Protocol", In Proceedings of the 1st USENIX Workshop in Electronic Commerce, 1995, pp.77–88.

[14] S. Ketchpel. 1995." Transaction Protection for Information Buyers and Sellers", In Proceedings of the Dartmouth Institute for Advanced Graduate Studies: Electronic Publishing and the Information Superhighway, 1995.

[15] W. Trappe, L. C. Washington, "Introduction to Cryptography with Coding Theory", Prentice Hall, 2001.

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

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

[18] S. G. Akl, "Digital Signatures: A Tutorial Survey", IEEE Computer, vol.16, no. 2, February 1983, pp. 15-24.

[19] U.S. Department of Commerce, "Digital Signature Standard (DSS)", Federal Information Processing Standards Publication 186, 1994.

[20] NIST, "Digital Signature Standard (DSS)", FIPS PUB 186-2, 27 January 2000.

[21] Microsoft, "Visual Studio .Net", http://msdn.microsoft.com/vstudio/

[22] 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.