An Anonymous Fair Exchange E-commerce Protocol*

Indrakshi Ray Indrajit Ray
Department of Computer and Information Science
University of Michigan-Dearborn
4901 Evergreen Road, Dearborn, MI 48128
Email: {iray, indrajit}@umich.edu

Abstract

In this paper we propose an e-commerce protocol for trading digital products over the Internet. The novel features of our protocol include: (1) ensuring fair exchange, (2) not requiring manual dispute resolution in case of unfair behavior by any party, (3) assuring each party that the item he is about to receive is indeed the correct one, (4) not requiring the active involvement of a trusted third party unless a problem occurs, and (5) ensuring anonymity for both the customer and the merchant. No existing e-commerce protocol that we know of has all these features.

1 Introduction

Researchers have identified a number of desirable characteristics that must be satisfied by e-commerce protocols: (i) should ensure fair exchange, (ii) should not require manual dispute resolution in case of unfair behavior by one party, (iii) each party should be assured that the item he is about to receive is indeed the correct one, (iv) should not require the active involvement of an online trusted third party, and (v) should ensure anonymity for the customer and optionally for the merchant. No existing e-commerce protocol that we know of satisfies all of these requirements simultaneously. We propose a protocol that satisfies all these properties. Before we outline our approach, we elaborate on each of these requirements.

Fair exchange: Ideally fair exchange requires that either both the parties involved in the transaction receive each other's items or none do. However, researchers [9, 10, 21] have often used the term in a weaker sense: the protocol gathers enough evidence during execution so that, in case one party behaves unfairly and obtains the other's item without sending his, the misbehaving party can be prosecuted. If a dispute occurs, a judge looks at the evidence and delivers his judgment. The dispute resolution is performed after the protocol execution, that is, after the customer has obtained his product or the merchant his money. However, such "after-the-fact" protection [14, 15] may be inadequate in an e-commerce environment where the customer and the merchant may be unreachable after the transaction.

Avoiding manual dispute resolution: The need for manual dispute resolution, when one party behaves unfairly, does not arise if protocols provide *true fair exchange* – under all circumstances[1] either both the parties receive each other's items or none do. Protocols providing true fair exchange [14] typically use an online trusted third party. The third party receives the information from each party involved in the e-commerce transaction and then forwards it to the other party. As a result if any party misbehaves or prematurely quits, no harm is caused to the other party.

Ensuring correct item will be received: Before exchanging the items, each party must have the confidence that the item he is about to receive from the other party will indeed be the correct one. The use of an online trusted third party helps to meet this requirement as well. For example, in the protocol proposed by Ketchpel [14] the third party verifies the contents of each item before forwarding it to the respective parties. Although using a trusted third party helps meet some requirements, the third party is a source of bottleneck for these protocols. Not only is the performance of the third party an issue, but also its vulnerability to denial of service attacks.

Reducing involvement of the third party: Several protocols have been proposed [1, 2, 3, 4] that do not use the third party unless a problem, such as, a party misbehaving or prematurely aborting, occurs. Such protocols are termed optimistic [1, 2, 3]. Most of these protocols do not ensure true fair exchange [1].

*This work has been partially supported by the National Science Foundation under grant EIA 9977548 and by a Summer Research Grant from the University of Michigan-Dearborn.

[1] This includes any party misbehaving or prematurely quitting.

Anonymity: One problem not addressed by existing true fair exchange protocols is anonymity. Anonymity ensures that the identity of a customer and, optionally, that of a merchant is not revealed during an e-commerce transaction. A customer, for example, may not want outsiders to compile a pattern of his spending habits. Thus the customer may want anonymity. A merchant, similarly, may also want to remain anonymous. Although anonymity is addressed by several payment mechanism schemes [6, 18, 16, 17], none of these schemes are satisfactory. Some of them [16, 17] actively use an online trusted third party; the anonymity is compromised if the third party colludes with the others. Some [6] provide anonymity but allows the customer to create money. Others [11] rely on tamper proof hardware devices.

We propose a protocol that ensures anonymity of the customer and the merchant. Money is transferred electronically from one bank to another in the form of a payment token that is generated by a bank. The individual parties cannot generate or duplicate a payment token. Moreover, the customer (the merchant) is not revealed the identity of merchant's (customer's) bank – this prevents collusion of different parties and compromising anonymity.

Besides anonymity, our protocol allows the customer to verify that the product he is about to receive is the one he is about to pay for. This verification is done by the customer using the theory of cross validation that was developed in a related paper [20]. Note that in other works [14], the trusted third party performs the verification and gives an assurance that the items the two parties are going to interchange will be the correct ones.

We do, however, rely on the trusted third party for ensuring true fair exchange. The merchant escrows the encrypted product and a pair of keys with the third party. Thus, if the merchant disappears after receiving the payment, the third party can always give the customer the keys for decrypting the product. But the third party is not involved unless a problem, such as, a party misbehaving or prematurely aborting occurs. Thus, the use of the third party is kept to a minimum level.

The rest of the paper is organized as follows. Section 2 gives an informal description of the protocol. Section 3 describes the theory of cross validation on which our protocol is based. Section 4 describes the basic protocol. Section 5 provides the extensions necessary to provide fair exchange when a party misbehaves or aborts. Section 6 analyzes how anonymity is assured by our protocol. Finally, section 7 concludes the paper.

2 Informal Description of Our Protocol

A merchant M who wishes to sell an electronic product m registers itself with the third party TP. M sends the product, its description which includes the cost, and a key pair (K_1, K_1^{-1}) to TP. TP encrypts the product with key K_1 and advertises it on the web. If M wants to remain anonymous, he also sends his one time public key, M_{ipub}, to TP.

The protocol begins with the customer, C, downloading an encrypted product from TP. C then sends a purchase order to M, using a pseudo identifier, together with a one time public key, C_{ipub}, that is to be used in the transaction. M responds by sending the product encrypted with a key, denoted by $K_1 \times K_2$, together with encrypted information about the account that he wishes to be credited. The key $K_1 \times K_2$ has a mathematical relation with the key K_1. Using the theory of cross-validation, C is able to verify that the product he is about to receive is the one he will be paying for.

When C is satisfied, he asks his bank CB to generate the payment token. The payment token can only be encashed by the encrypted account. CB generates the payment token, signs it with a signature common to all banks, and forwards it to C. C then forwards it to M who forwards it to his bank MB. MB after successfully incrementing M's account, sends a message to M. M after receiving this message sends the decrypting key to C.

3 Theory of Cross Validation

Before presenting our protocol we outline the theory of cross-validation on which the protocol is based and then show how this theory is used in ensuring that the product the customer is about to receive will be the correct one. For details the reader is referred to [20].

Definition 1 *The set of* messages $\mathcal{M}$ *is the set of non negative integers m that are less than an upper bound N, i.e.*

$$\mathcal{M} = \{m | 0 \leq m < N\} \tag{1}$$

Definition 2 *A* key K *is defined to be the ordered pair* $< e, N >$*, where N is a product of distinct primes, e is relatively prime to the Euler's totient function* $\phi(N)$*; e is the* exponent *and N is the* base *of the key K.*

Definition 3 *The* encryption *of a message* m *with the key K* $= < e, N >$*, denoted as [m,K], is defined as*

$$[m, < e, N >] = m^e \mod N \tag{2}$$

Definition 4 *The* inverse *of a key* $K = < e, N >$*, denoted by* K^{-1}*, is an ordered pair* $< d, N >$*, satisfying* $ed \equiv 1 \mod \phi(N)$.

Theorem 1 *For any message m.*

$$\left[[m, K], K^{-1}\right] = \left[\left[m, K^{-1}\right], K\right] = m \tag{3}$$

where $K = < e, N >$ *and* $K^{-1} = < d, N >$.

Definition 5 *Two keys* $K_1 =< e_1, N_1 >$ *and* $K_2 =< e_2, N_2 >$ *are said to be* compatible *if* $e_1 = e_2$ *and* N_1 *and* N_2 *are relatively prime.*

Definition 6 *If two keys* $K_1 =< e, N_1 >$ *and* $K_2 =< e, N_2 >$ *are compatible, then the* product *key,* $K_1 \times K_2$*, is defined as* $< e, N_1 N_2 >$.

Theorem 2 *For any two messages* m *and* $\hat{m}$*, such that* $m, \hat{m} < N_1, N_2$*,*

$$[m, K_1 \times K_2] \equiv [\hat{m}, K_1] \bmod N_1 \text{ if and only if } m = \hat{m} \quad (4)$$

$$[m, K_1 \times K_2] \equiv [\hat{m}, K_2] \bmod N_2 \text{ if and only if } m = \hat{m} \quad (5)$$

where K_1 *is the key* $< e, N_1 >$*,* K_2 *is the key* $< e, N_2 >$ *and* $K_1 \times K_2$ *is the product key* $< e, N_1 N_2 >$.

3.1 Ensuring Correct Product will be Received

We claim that a customer C is able to ensure that the product m he is about to receive from the merchant M, is the same as the one he ordered, before C pays for or receives the product. This property is achieved using the results of theorem 2. M sends the product m to the third party TP to be encrypted with a key K_1 and placed at a public place, as an advertisement for m. When C decides to purchase m from M, he acquires $T = [m, K_1]$ from TP and keeps it for future validation of the product received.

To sell m to C, M selects a second set of keys (K_2, K_2^{-1}) such that K_2 is compatible with K_1 according to definition 5. M provides C with $T' = [m, K_1 \times K_2]$.

C verifies that $[m, K_1]$ and $[m, K_1 \times K_2]$ are encryption of the same message m by verifying: $T \equiv T' \bmod N_1$, as per equation (4).

When satisfied, C sends the payment token. M in return, sends the decrypting key K_2^{-1}. C obtains m using $m = \left[T', K_2^{-1}\right]$. The proof of correctness follows from the above theorems.

3.2 Security

In the theory presented in section 3, if e is chosen small and C can guess e correctly, we can have a security problem[2]. Assume that the exponent e is small, say e=3. C starts as if he is buying the same product m three times, but always stops after having received $[m, K_1 \times K_2]$, $[m, K_1 \times K_3]$, $[m, K_1 \times K_4]$, where $K_2 =< e, N_2 >$, $K_3 =< e, N_3 >$ and $K_4 =< e, N_4 >$.

Let $N = N_1 \times N_2 \times N_3 \times N_4$. Knowing $m^e \bmod N_i$, for i $= 1 \ldots 4$, the attacker can, using the Chinese remainder theorem [19], compute m. Thus C can get the product, without paying for it. Note that this attack is similar to the low exponent attack on the RSA cryptosystem [13]. However, since C does not know the value of e, this problem will not arise. Below we provide an additional mechanism using which the security will not be compromised even if C can guess e correctly.

For every transaction that M performs, M chooses a random number r such that r is relatively prime to N_2. C downloads $[m, K_1]$ from TP. Rather than sending $[m, K_1 \times K_2]$ to C, M sends the following: $[m.r, K_1 \times K_2]$, $[r, K_1]$, where $m.r$ is the product of m with r. To validate the product, C multiplies $[m, K_1]$ with $[r, K_1]$ and the resulting product is compared with $[m.r, K_1 \times K_2]$. If both match, then C is confident that the product he is about to receive is the one he is going to pay for. Finally, instead of sending just K_2^{-1}, M now sends K_2^{-1} and r^{-1} where r^{-1} is the multiplicative inverse of r modulo N_2. Using the decrypting key K_2^{-1}, C obtains $m.r \bmod N_2$. Multiplying this by r^{-1}, C can retrieve m.

4 The Basic Protocol

Table 1 lists the notations used in the description of the protocol. We make the following assumptions in the protocol: (i) Encrypted messages cannot be decrypted without proper keys. Digital signatures cannot be forged. Cryptographic checksums ensure integrity of messages. (ii) All parties use the same algorithm for encryption as well as for generating cryptographic checksums. (iii) Customer and the merchant each have a bank account. (iv) Identity of any party cannot be revealed from the IP address alone. (v) A constant time out period known to all parties is used when a party waits for a message from another party.

Prelude

Before the protocol begins, we assume that the following steps have already executed that sets up the environment in which the protocol operates. Note that even after these steps have executed the e-commerce transaction may not execute.

1. M **registers with** TP**.** For every product, m, that M wants to advertise with TP, he sends the following: (i) m and its description, and (ii) the key pair (K_1,K_1^{-1}). TP performs the encryption before uploading $[m, K_1]$ on the catalog. In this manner TP is able to certify that the product meets its claim.

2. M **keeps a public key with** TP**.** M generates a public/private key pair, (M_{ipub}, M_{iprv}), and provides the public key, M_{ipub}, to TP. M requires C to use the public key M_{ipub} for this transaction. M may choose to

[2] Although we use an asymmetric cryptographic system in this protocol, unlike public key cryptosystems we do not disclose the exponent e.

Symbol	Interpretation
C, M, and TP	Customer, Merchant, and Third Party
CB and MB	Customer's Bank and Merchant's Bank
C_{acct}	Customer's bank account with CB
M_{acct}	Merchant's bank account with MB
m	Product the customer is purchasing
PO	Purchase order used to order product m
T_i	Transaction involving purchase of m
A_{prv}, A_{pub}	A's private and public keys
A_{iprv}, A_{ipub}	A's private and public keys used only in T_i
B_{cprv}, B_{cpub}	Common private and public keys for banks
$A \Longrightarrow B : X$	A sends X to B
$[X,K]$	Encryption of X with key K
$CC(X)$	Cryptographic checksum of X
K_1	Key given to the TP by M
K_1^{-1}	Decrypting key of K_1 held by third party
K_2	Key that is compatible with K_1
K_2^{-1}	Decrypting key corresponding to K_2
r	Random number chosen by merchant for T_i
r^{-1}	Multiplicative inverse of r modulo N_2
$\mathcal{P}$	Payment token used for paying for the product

Table 1. Symbols used in protocol description

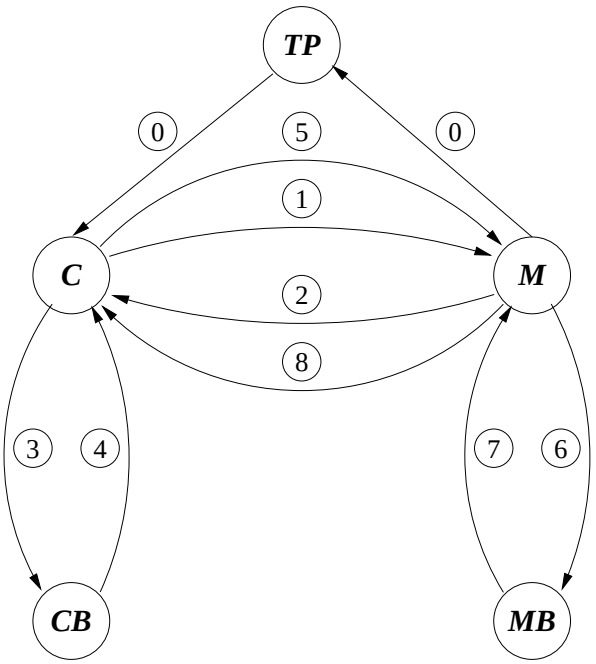

Figure 1. The Basic Protocol

change this public key after each transaction or periodically. This step may be omitted if M does not wish to remain anonymous and his public key can be obtained from elsewhere.

3. C **selects a product to purchase.** C downloads $[m,K_1]$ and M_{ipub} from TP. Note that C does not actually have the product m, because he does not have the decrypting key K_1^{-1}. This $[m,K_1]$ will be used later by C to validate the product received from M.

4. C **generates a one time public/private key.** C generates a public/private key pair, (C_{ipub},C_{iprv}), which he wishes to use during the current e-commerce transaction T_i.

Protocol Description

The basic protocol involves the following steps when no party misbehaves or prematurely quits. The messages exchanged in the protocol are shown in figure 1. Note that only the circled message numbers are shown in the figure. The message numbered 0 corresponds to steps that have been performed in the prelude.

Message 1

$C \Longrightarrow M : PO, [CC(PO),C_{iprv}], [C_{ipub},M_{ipub}]$

C initiates the e-commerce transaction by sending M three things: (i) a purchase order, PO, (ii) a signed cryptographic checksum of the purchase order, $[CC(PO),C_{iprv}]$, and (iii) the public key of C encrypted with M's public key, $[C_{ipub},M_{ipub}]$.

The purchase order, PO, contains the details of the order, such as, the product C is purchasing, the price C is paying, the pseudo identities of C and M, and other necessary information.

C generates a cryptographic checksum of the PO and signs it. This way M can verify if the PO was received correctly. C's signature forestalls debate over whether C expressed intention to purchase the product.

C also sends his public key, C_{ipub}, that is to be used for this transaction. To prevent intruders from observing this key, it is encrypted with M's public key M_{ipub}.

Message 2

$M \Longrightarrow C : [Abort, M_{iprv}]$

OR

$M \Longrightarrow C : [CC(PO),M_{iprv}],$
$[m.r,K_1 \times K_2], \quad [CC([m.r,K_1 \times K_2]),M_{iprv}],$
$[r,K_1],[CC([r,K_1]),M_{iprv}],$
$[M_{acct},MB_{pub}],[CC([M_{acct},MB_{pub}]),M_{iprv}]$

M after receiving **Message 1** checks if the purchase order is to his satisfaction.

If M is not satisfied, he sends an abort message to C and aborts the transaction.

If M is happy with the purchase order and wants to continue with the protocol he sends the following things to C: (i) signed cryptographic checksum of the PO, $[CC(PO), M_{iprv}]$, (ii) encrypted product, $[m.r, K_1 \times K_2]$, (iii) signed cryptographic checksum of the encrypted product, $[CC([m.r, K_1 \times K_2]), M_{iprv}]$, (iv) the random number, r, encrypted with K_1, $[r, K_1]$, (v) its signed cryptographic checksum $[CC([r, K_1]), M_{iprv}]$, (vi) encrypted account information, $[M_{acct}, MB_{pub}]$, and (vii) signed checksum of the encrypted account, $[CC([M_{acct}, MB_{pub}]), M_{iprv}]$.

M's endorsement on the purchase order forestalls debate over whether the purchase order was received correctly or not and whether M agreed to the terms of the current transaction.

M also sends the encrypted product and a signed cryptographic checksum of the encrypted product. In this and the following messages, the purpose of a signed cryptographic checksum of a message is twofold: (a) it can be used as evidence of what the sender has actually sent and (b) it also ensures the integrity of the message while in transit.

To remain anonymous, M should not reveal his bank account information to C. So he sends his account information encrypted with the public key of his bank, MB_{pub}. C will not be able to decrypt this information and get M's account number.

Message 3

$C \Longrightarrow CB : [[MTI, C_{prv}], CB_{pub}]$

After receiving **Message 2** from M, C checks to see if it is an abort message or the encrypted product. If it is an abort, C aborts the transaction.

If C has received the encrypted product from M, he validates the product as outlined in Section 3.2. If the two compare and C is still interested in buying the product, he sends the bank a signed money transfer instruction, MTI, encrypted with CB's public key.

The money transfer instruction, MTI, consists of the following: (a) the amount of money that is being transfered, (b) C's account that is to be debited C_{acct}, and (c) M's encrypted account that is to be credited $[M_{acct}, MB_{pub}]$.

If, on the other hand, the product is not validated C skips **Message 4** and sends an abort in **Message 5**.

Message 4

$CB \Longrightarrow C : [[\mathcal{P}, B_{cprv}], C_{pub}]$

OR

$CB \Longrightarrow C : [Failure, C_{pub}]$

After receiving **Message 3**, CB, first checks to see if C's account has enough money. Then it sends an appropriate response to C – either the payment token $\mathcal{P}$ appropriately signed and encrypted or a failure message.

The payment token $\mathcal{P}$ created by CB contains the following information: (a) the amount that is being credited, (b) the encrypted account that will be credited, (c) a nonce to prevent replay attacks.

In order for MB to honour the payment token, it must be signed by a bank. However, to prevent collusion and compromisation of anonymity, we do not want MB to know who CB is. To solve this problem, we assume that all banks share a common public, private key pair (B_{cprv}, B_{cpub}). If a bank signs a message using the key B_{cprv}, then the recipient of the message will be able to verify that the message has been signed by a bank but will not be able to guess which bank signed it. This idea is similar to using a group signature scheme [7, 8]. A group signature is publicly verifiable but it maintains anonymity of the signer.

Message 5

$C \Longrightarrow M : [[\mathcal{P}, B_{cprv}], M_{ipub}]$.

OR

$C \Longrightarrow M : [Abort, C_{iprv}]$.

Depending on whether or not C is interested in successfully completing the transaction, he sends either the signed and encrypted payment token that it got from CB in **Message 4** or an abort message to M.

Message 6

$M \Longrightarrow MB : [[\mathcal{P}, B_{cprv}], MB_{pub}]$.

If M receives an abort message in **Message 5**, he aborts the transaction. On the otherhand if he receives the payment token signed by CB, he forwards it to his own bank MB after encrypting it with MB's public key.

Message 7

$MB \Longrightarrow M : [ack, MB_{prv}]$

MB after receiving the payment token in **Message 6** decrypts the message to get the account number that is to be credited. Then it credits the appropriate account and sends an acknowledgement to M.

Message 8

$M \Longrightarrow C : [K_2^{-1}, C_{ipub}], [CC(K_2^{-1}), M_{iprv}],$
$[r^{-1}, C_{ipub}], [CC(r^{-1}), M_{iprv}]$

After receiving the acknowledgement in **Message 7**, M sends the product decryption key, K_2^{-1} and the multiplicative inverse of r modulo N_2, namely, r^{-1}, to C.

Using these C can decrypt the product as outlined in Section 3.2.

5 Extensions to Ensure True Fair Exchange

True fair exchange requires that either both parties obtain each other's items or none do. In the context of our protocol fair exchange will be ensured if either both M receives the payment token and C the product or none do. C is said to have received the product if he receives both the encrypted product and the required decrypting key. M is said to have received the payment token once he has verified that it is the correct payment token. Now when all the parties behave honestly and executes the protocol presented in Section 4, true fair exchange is ensured. In the following paragraphs we outline the extensions necessary to ensure true fair exchange even if any party misbehaves or prematurely aborts.

In this protocol, the parties interact by sending and receiving messages. One party when expecting a message from the other party cannot wait for an indefinite amount of time.[3] This problem is solved by associating a time out period with each message. Both the sender and the receiver of the message is aware of this time out period. If the receiver does not hear from the sender in this time period, he first sends a message to the sender saying that he has not received the message and again waits to hear from the sender. If the sender responds before the time out period, the protocol proceeds as before. If the sender does not respond, the receiver assumes that the sender does not want to continue with the protocol. Depending on the stage of execution of the protocol, the receiver takes the appropriate step. For example, if the protocol is in the early stages of execution then the receiver can choose to discontinue with the protocol. On the other hand, if the protocol is in the late stages of execution, then the receiver can take this up with TP.

Before proceeding further, we wish to draw attention to one fact. Fair exchange is compromised if M chooses to discontinue with the protocol after receiving the payment token. However, if any party discontinues with the protocol before C sends the payment token, fair exchange is not compromised.

Note that since M merchant sends the decryption key only after it has received payment in a satisfactory matter, it will always be the case that C initiates the extended protocol. The extended protocol involves interaction with TP and is initiated by C by sending the messages and the signed checksums it has received from M – evidences of M misbehaving – and the payment token.

TP verifies the payment token. TP then gets in touch with M and asks him to send the product decryption key. TP then starts a timer. M on receiving the message can send the product decryption key to TP. TP can forward this to C. Alternatively, M may claim that he did not send the product decryption key because he has not received the payment token from C. TP, in this case, sends the payment token to M – M can then continue with the rest of the protocol starting with **Message 6** of the Basic Protocol in section 4.

If M does not respond within the timeout period, TP can verify the payment token from any bank – that it is of the correct amount and it is indeed signed by a bank – and then forward the message containing the key, ${K_1}^{-1}$, to C. C can use ${K_1}^{-1}$ to decrypt $[m, K_1]$ that he downloaded from TP at the beginning of the protocol.

Analysis of True Fair Exchange

Both C and M behave properly: It is easy to see that in this case C obtains the product and M the payment token.

M behaves improperly: This includes the following:

1. M receives the correct payment token in **Message 5** but does not send the product decryption key in **Message 8**.

   C in this case initiates the extended protocol by presenting all the messages received from the M. TP gets in touch with M and requests for the product decryption key. If M does not comply or has disappeared, TP sends C the decryption key K_1^{-1} that is escrowed with him, and asks C to decrypt $[m, K_1]$ with this key.

2. M sends the wrong product decryption key.

   This is handled as in the above case.

3. M falsely claims that he has not received the payment token or has not received the correct payment token.

   Note that if M does not receive the payment token or receives incorrect payment token, he does not send the product decryption key. Now when C initiates the extended protocol, he also gives TP the payment token. So if M claims not receiving the payment token, TP can verify the payment token, forward it to M and request him for the product decrytion key. A final point is that M will not benefit by depositing the payment token multiple times.

C behaves improperly: There are three ways in which C can behave improperly.

1. C falsely claims that it has sent the payment token but has not received the product decryption key.

   In this case, C initiates the extended protocol by sending the necessary evidence together with the

[3] This may happen if the other party has simply disappeared.

payment token. TP then gets in touch with M and sends him the payment token, M then continues with **Message 6** of the basic protocol.

2. C has send inadequate amount in payment token.
   Note that M can verify the payment token send by the C. If it contains inadequate amount, M does not deposit the payment token but gets in touch with C. If C does not make ammends, he just keeps the copy of the payment token and does not send the product decryption key. Later on, if contacted by TP, he can present the payment token as evidence of unfair behavior by C. TP can then send the correct payment token which he obtained from C.
3. C falsely claims that he has received the wrong product decryption key.
   In any abnormal scenario C gets in touch with TP and presents all the messages it received from M. So if he makes a false claim, then TP detects it.

6 Ensuring Anonymity

One of the primary objectives of this protocol is to protect the anonymity of C under all possible scenarios. To get assurance of anonymity we must ensure that (i) no single party has enough information to link C to M and (ii) it will not be possible for all the parties to collude and get this information. The protocol proposed by Low et al. [16] ensures (i) but not (ii).

To analyze anonymity we follow the approach of Low et al.[16, 17] and tabulate the information that each party knows in Table 2. The entries Y, N, and M stand for Yes, No, and Maybe respectively. The table is interpreted as follows. Consider first row 1. A Y under column CB indicates that the identity of C is known by CB; N under the columns MB, M and TP indicate that none of the entities MB, M and TP know the identity of C. Consider next row 8. N's under columns CB and MB indicate that neither CB nor MB knows anything about PO; Y under column M indicates that M knows PO; finally a M under column TP indicates that the third party may or may not know anything about PO – depending on whether the extended protocol is executed or not.

Analyzing Collusion

Note that the necessary conditions for two parties to collude are (a) the two parties must know each other's identity and (b) the two parties must have some common piece of information pertaining to the transaction that C carries out with M. From table 2 it is clear that no party alone has enough information to link C and M.

Information	CB	MB	M	TP
C	Y	N	N	N
CB	Y	N	N	N
MB	N	Y	Y	N
M	N	Y	Y	N
TP	N	N	Y	Y
C_{acct}	Y	N	N	N
M_{acct}	N	Y	Y	N
PO	N	N	Y	M
M_{ipub}	N	N	Y	Y
M_{pub}	N	Y	Y	N
C_{ipub}	N	N	Y	M
C_{pub}	Y	N	N	N
$[m.r, K_1 \times K_2]$	N	N	Y	M
$[r, K_1]$	N	N	Y	M
$[K_2^{-1}]$	N	N	Y	M
$[r^{-1}]$	N	N	Y	M
$\mathcal{P}$	Y	Y	Y	M

Table 2. Information Possessed by Each Party

Two party collusion

We use table 2 to derive the information obtained when any two parties collude. The possible two party collusions and the knowledge they obtain after colluding are given below.

TP **and** M**:** M does not learn anything new by colluding with TP. As a result of this collusion, TP will have the knowledge possessed by M.

TP **and** CB**:** TP and CB do not know each other's identity and they will not be able to collude.

TP **and** MB**:** TP and MB do not know each other's identity and so they cannot collude.

M **and** CB**:** M and CB do not know each other's identity and so cannot collude.

M **and** MB**:** M learns nothing new from MB. If they collude, MB will know what M knows.

CB **and** MB**:** CB and MB does not know each other's identity and so they cannot collude.

Summarizing the results above, we can say that the only parties to get new information as a result of two party collusions are TP and MB. However, both of them get the information which is already possessed by M. Since M does not have enough information to link C with M, as a result of this collusion the colluding parties also will not have this information.

Three Party Collusion

From the previous section, it follows that the only possible two party collusions are (i) TP and M and (ii) M and MB. Thus the only three party collusion is between TP, M and MB. Now even if these three parties collude, the only information they can collectively obtain is whatever M already possesses in Table 2. Since M does not have the information linking C to M, anonymity of C is preserved.

Four Party Collusion

Even after the three parties TP, M and MB collude they will not know the identity of the fourth party CB. Hence, no four party collusion is possible in our protocol.

7 Conclusion

In this work we have proposed an e-commerce protocol that has some desirable features. First, it provides fair exchange under all circumstances. Second, the protocol does not require any manual dispute resolution in case any party behaves unfairly. Third, the protocol does use a third party; however, the third party does not become actively involved unless a problem occurs. Fourth, the protocol allows the customer to be confident that he is paying for the correct product before actually paying for it. Fifth, the protocol provides anonymity for the customer and the merchant.

One future work is to optimize the protocol by reducing the number of messages exchanged between the parties. Another important future work is evaluating the correctness of the protocol using formal methods of software verification like model checking [12] and theorem proving [5].

References

[1] N. Asokan, M. Schunter, and M. Waidner. Optimistic Protocols for Fair Exchange . In T. Matsumoto, editor, *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 7–17, Zurich, Switzerland, Apr. 1997.

[2] N. Asokan, V. Shoup, and M. Waidner. Asynchronous Protocols for Optimistic Fair Exchange. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 86–99, Oakland, California, May 1998.

[3] N. Asokan, V. Shoup, and M. Waidner. Optimistic Fair Exchange of Digital Signatures. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Eurocrypt '98*, pages 591–606, Helsinki, Finland, June 1998.

[4] F. Bao, R. H. Deng, and W. Mao. Efficient and Practical Fair Exchange Protocols with Off-line TTP. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, May 1998.

[5] D. Bolignano. Towards the Formal Verification of Electronic Commerce Protocols. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop*, June 1997.

[6] D. Chaum. Security without Identification: Transaction Systems to make Big Brother obsolete. *Communications of the ACM*, 28(10):1030–1044, Oct. 1985.

[7] D. Chaum and E. van Heijst. Group Signatures. In *Advances in Cryptology – EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.

[8] L. Chen and T. P. Pederson. Group Signatures. In *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181. Springer-Verlag, 1995.

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

[10] R. H. Deng, L. Gong, A. A. Lazar, and W. Wang. Practical Protocols for Certified Electronic Mail. *Journal of Network and System Management*, 4(3), 1996.

[11] S. Even, O. Goldreich, and Y. Yacobi. Electronic Wallet . In D. Chaum, editor, *Advances in Cryptology – CRYPTO '83*. North-Holland/Elsevier, 1984.

[12] N. Heintze, J. Tygar, J. Wing, and H. Wong. Model Checking Electronic Commerce Protocols. In *Proceedings of the 2nd USENIX Workshop in Electronic Commerce*, pages 146–164, November 1996.

[13] B. Kaliski and M. Robshaw. The Secure Use of RSA. *CryptoBytes*, 1(3):7–13, 1995.

[14] S. Ketchpel. Transaction Protection for Information Buyers and Sellers. In *Proceedings of the Dartmouth Institute for Advanced Graduate Studies '95: Electronic Publishing and the Information Superhighway*, 1995.

[15] S. Ketchpel and H. Garcia-Molina. Making Trust Explicit in Distributed Commerce Transactions. In *Proceedings of the 16th International Conference on Distributed Computing Systems*, pages 270–281, 1996.

[16] S. Low, N. Maxemchuk, and S. Paul. Anonymous Credit Cards. In J. Stern, editor, *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 108–117, Fairfax, Virginia, Nov. 1994.

[17] S. H. Low, N. F. Maxemchuk, and S. Paul. Anonymous Credit Cards and Their Collusion Analysis. *IEEE/ACM Transactions on Networking*, 4(6), Dec. 1996.

[18] G. Medvinsky and B. C. Neuman. Netcash: A Design for Practical Electronic Currency on the Internet . In V. Ashby, editor, *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 102–106, Fairfax, Virginia, Nov. 1993.

[19] I. Niven and H. S. Zuckerman. *An Introduction to the Theory of Numbers*. John Wiley and Sons, 4th edition, 1980.

[20] I. Ray, I. Ray, and N. Narasimhamurthi. A Fair-Exchange Protocol with Automated Dispute Resolution. Technical report, University of Michigan -Dearborn, 2000.

[21] J. Zhou and D. Gollmann. A Fair Non-repudiation Protocol. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 55–61, Oakland, California, May 1996.