A Methodology for Detecting Violation of Real-Time Constraints in Secure Electronic Commerce Transactions

Srividhya Subramanian and Mukesh Singhal

Dept. of Computer and Information Science
The Ohio State University, Columbus, OH 43210-1277
Tel: , Fax: (614)-292-2911
E-mail: {ssubrama,singhal}@cis.ohio-state.edu

Abstract

In electronic commerce applications, it is often critical to transact within a short stipulated time. In such applications, it is financially damaging to the customer if the transaction does not complete within the stipulated time. In this paper, we describe various situations in the stock market (one of the biggest electronic commerce markets) requiring real-time transactions. We propose a methodology to enhance any electronic commerce protocol to a real-time aware electronic commerce protocol. The methodology consists of two parts: the first part adds a timestamp to some messages in the original protocol and the second part augments the protocol with a mechanism to detect violation of real-time constraints. We show that the methodology does not violate any of the useful properties present in existing electronic commerce protocols. As a result, the methodology may be successfully applied to any existing electronic commerce protocol. The methodology is illustrated by applying it to an existing well known electronic commerce protocol, namely, the NetBill protocol.

Keywords: Electronic commerce, real-time, transaction, security, atomicity, privacy, anonymity, stock market.

1 Introduction

There is a plethora of products already available to us through the World Wide Web or through email. Some of these are electronic commodities such as electronic books, newspapers, reports, typesetting services, ownership titles, shares, software, etc. Others are non-electronic goods that are advertised, ordered and paid for electronically. According to a projection by *Business Week (12 June 1995)*, trade worth 1,650 billion dollars will be conducted over electronic networks by the year 2000. This growing shift towards electronic marketing can be attributed to certain factors; one of the greatest factors being the ability to transfer goods and information quickly and yet inexpensively.

In many situations, it is important that transactions complete within a fixed period of time (real-time transactions). This may be either because the price agreed upon changes rapidly with time (e.g., currency market, stock market, options, etc.), or because the product is useful to the customer only for a certain period of time (e.g newspapers, stock updates, etc.). Since such transactions must complete almost instantaneously, electronic networks seem to be the ideal medium to conduct them over.

There are many ways for a business to provide quick transaction execution - faster networks, more and faster memory, faster machines, better algorithms, more employees, etc. However, even with the fastest resources, a transaction may occasionally take more than the stipulated time. Delays may be due to failures, hot-spots, laziness on part of the business employees or other uncontrollable factors. Under such circumstances, the customer does not receive the full value for the payment made. The customer is, therefore, entitled to a full/partial refund. However, it is not easy to detect if an event took place within a certain time. This is due to the non-existence of global clocks and the ability to tamper with the system clocks of unsecured computers. A mechanism is required to detect if a transaction took more than the time stipulated for its completion. The following examples should illustrate this need.

One of the areas that can immensely benefit from real-time aware protocols is the stock-market. In the stock market, decisions to buy, sell, or short shares of a company are based on current stock prices. It is consequently important for an investor to receive current price quotes whenever requested. The value of a 20 minutes old is low enough that it is usually provided free of cost. In fact, some financial analysts believe that the value of a quote falls exponentially with time. Thus, current quotes requested must be provided within a very short period of time (in the order of seconds), for them to be valuable. If an investor does not receive current quotes within a stipulated period of time, there must be a mechanism to verify this and refund the payment towards the quotes.

Assuming investors receive their quotes on time and place orders to transact shares with a broker. It is important for these orders to be acted upon immediately. The orders must not only reach the floor brokers (agents at the stock exchange) in time, the floor brokers must also act on it with alacrity. If either the order does not reach the floor broker or the floor broker does not act quickly, the investor may miss the market. Consider an investor who through technical analysis (a tool to predict future trend from past price fluctuations) finds that the price of a stock has reached its upper threshold and is soon going to dip. This investor instructs a broker to sell shares at the market value (best immediate price available). If the broker does not act upon this instruction with alacrity, the share value may start dipping and the investor will lose some of his profits. Since the investor pays the broker for the broker's ability to immediately transact, the investor should at least be refunded the broker's fee, if the broker is not able to deliver on his promise. Under certain circumstances, the broker may also be liable for the losses incurred by the customer. We need a mechanism for the investor to verify if the transaction took place within a stipulated period of time and receive a refund/compensation if this does not happen.

The main difference between the two examples lies in the fact that in the second example, it is not essential for the investor to receive the trade confirmation immediately, only for the sale to take place. While many protocols exist for electronic commerce [Cyb, Dig, SSBR94, Man95, BGH$^+$95, Hic95, B$^+$94], none addresses the issue of real-time transactions.

In this paper, we model the general problem of detecting real-time transaction executions and identify one special case where the real-time phase of a transaction can complete without delivering a product to the customer. We propose a methodology to enhance existing protocols with a general mechanism to detect real-time execution. We then propose a modified methodology that optimizes this mechanism for the special case where there is no product to be delivered to the customer. Furthermore, we show that the general methodology does not violate any of the useful properties of existing electronic commerce protocols and hence is applicable to them. We then illustrate the methodology by applying it to NetBill, a well known existing electronic commerce protocol.

The rest of the paper is organized as follows. Section 2 models the general problem of detecting violations of real-time constraints. Section 3 describes the methodology to enhance any electronic commerce with a real-time violation detector. Section 4 illustrates how the methodology works when applied to an existing protocol, NetBill. Finally, in Section 5 we summarize the work and point out directions for future research.

2 The Real-time Aware Transaction Model

In the previous section, we saw the need for a mechanism to detect violations of real-time constraints. In this section, we will look closely at the assumptions we make about the system, the description of the problem, and the properties required of such a violation detection mechanism.

2.1 System Model

A general real-time transaction execution involves two parties: the merchant and the customer. The merchant is willing to supply a product (P) within a specified period of time

(T) for a fee (F). The customer is willing to pay the fee F in return for receiving the product P within time T. The time T is measured from the time the customer sends the order for P. This is illustrated in Figure 1.

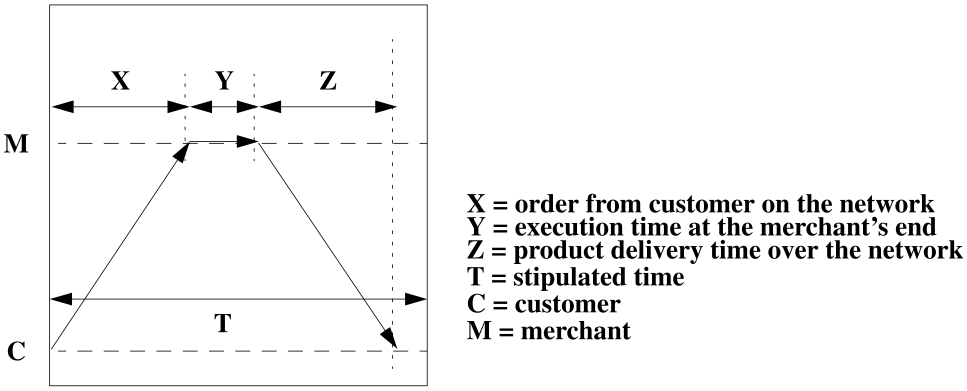

Figure 1: Timing diagram showing a real-time transaction

A special case of this is when the customer is satisfied if the product P is dispatched within a stipulated time from the *receipt* of the order by the merchant. In Figure 1 the constraint would not be on X + Y + Z, but only on Y. Here, the customer is willing to incur losses or delays that occur over the network (X, Z). This may be the case when the product being supplied is the confirmation of a service, say, the sale of some shares. Here, it is important for the sale to be completed on time, but it is not important for the confirmation of the sale to reach the customer in real-time.

The merchant and the customer are connected through an electronic network. They exchange messages containing information, goods, payment, etc. over this network. The communication channels are FIFO. The customer is in close proximity (measured in terms of message delay) from a trusted clock. This trusted clock may be a secure co-processor [Yee94] attached to the customer's workstation. Secure co-processors are secure physical devices that erase their RAM and CPU registers if they are accessed in any method apart from the defined interfaces. These devices have limited capabilities, are inexpensive and are available commercially [WW^{+}91]. Many customers may share the same clock. In order to avoid problems involving clock synchronization, we use a single trusted clock for all measurements in a single transaction. The local clocks of the customer and the merchant may not be synchronized with this clock, but we expect the clock periods to be the same. This is a reasonable assumption given the accuracy of modern day clocks. The secure clock also has a pair of public and private keys.

2.2 Requirements

To be used in conjunction with other electronic commerce protocols, our methodology must not violate any of the useful security related properties ensured by these protocols. Several properties have been identified as either required or desirable in electronic commerce protocols [NM95, And94]. These properties are listed below.

- *Privacy*: No third party should be able to know details about the transaction (payment amount, product details, identity of parties that wish to remain anonymous, etc.)
- *Anonymity*: Nobody (including the merchant involved in the transaction) should know the true identity of the customer involved. At the least, nobody should be able to trace all transactions made by a customer and create a profile of the customer.
- *Security*: Tampering or replay of messages by a third party, non-adherence to the protocol by involved parties, or corruption or loss of message in the network should not cost more than some additional computation to either party.
- *Atomicity*: Under all circumstances, the transaction must either go to completion (the right goods are exchanged for the right amount of money) or be aborted (there is no exchange of goods or money).
- *Low Overhead*: The cost for each transaction must be low enough to justify transactions involving a small amount of money.

3 The Methodology

In this section, we describe the general methodology to enhance an electronic commerce protocol to make it real-time aware. We then show a modified methodology for the special case where the network delivery of the customer order or of the product is not included in the real-time constraint. The general methodology is broken into two parts. The first part specifies some modifications to be made to the original protocol. This mainly involves adding certain timestamps to some messages. The second part is the actual detection mechanism that uses the timestamps added in the first part to detect if the real-time constraints were violated.

Subsection 3.1, Subsection 3.2, and Subsection 3.3 all pertain to the general methodology. These subsections describe the modification to the original protocol, the detection mechanism and the conformity with the requirements specified in Subsection 2.2, respectively. Subsection 3.4 describes the modified methodology for the special case with no network delivery in the real-time constraints.

Merchant Id	*Customer Id*	*# of messages received*
1	1	72
2	1	4
3	1	3
4	1	1
5	1	2
6	1	4
3	11	1
5	11	17

Figure 2: Vector timestamp

3.1 Modification to the Transaction Protocol

The original protocol is modified by first identifying the following messages in the transaction protocol.

- **Order**: A message from the customer to the merchant where the customer makes a binding agreement to buy the goods. This may be in the form of a payment, a binding promise to pay, etc.
- **Product**: A message from the merchant to the customer that finally makes the goods available to the customer. The message may contain the goods themselves, decryption key to previously supplied encrypted goods, etc.

The detection mechanism detects if the time between the dispatching the first message (**Order**) and receiving the second message (**Product**) is greater than the stipulated time.

The trusted clock (possibly, an inexpensive secure co-processor) maintains a vector timestamp (Figure 2) . The timestamp has a field associated with each merchant, customer pair. The field corresponds to the number of messages received from the merchant by that customer.

Order and **Product** are both routed through the trusted clock. The customer attaches a nonce to the **Order** before sending it. The merchant attaches the same nonce to the **Product**. This serves to match the two messages as part of the same transaction. The merchant further encrypts the **Product** with the trusted clock's public key. This prevents the customer from rerouting the **Product** and by-passing the secure clock. The clock attaches to these messages a timestamp consisting of two fields : the current time as per the trusted clock and the number of messages received from the merchant. The significance of these will be clear in Subsection 3.2 where we describe the detection mechanism. The clock further signs the timestamped message with its private key. This prevents anyone from tampering with or forging a timestamped message. This is illustrated in Figure 3 where T refers to the current time, N to the number of messages from the vector timestamp, R to the nonce, S_K to encryption with public key K, and $S_{\frac{1}{K}}$ is the signature function applied using the private key K. The clock also sends a copy of the **Order** after timestamping and signing it to the customer for the customer's records.

Old Msg	*New message sent to clock*	*New Timestamped Msg*
Order	Order, R	Order, $S_{\frac{1}{K}}$(Order, T_1, R, N_1)
Product	Product, S_K(Product, R)	Product, $S_{\frac{1}{K}}$(Product, T_2, R, N_2)

Figure 3: Modification to the Transaction Protocol

3.2 Detection Mechanism

We further augment the electronic commerce protocol with a detection mechanism, that is to be executed when the customer *believes* that the product hasn't been delivered on time. Note that no real-time violation will be detected unless the customer executes the detection mechanism. Thus, the extra cost of the detection mechanism is not associated with transactions that completed to the satisfaction of the customer.

The detection mechanism consists of a demand for refund (**Demand**) sent by the customer via the trusted clock. The **Demand** is simply a plain text message asking for refund that is timestamped and signed in the same manner as an **Order**. Copies of some old messages are also attached to the **Demand**. This **Demand** is then evaluated by the merchant. Refund is given if and only if there was a violation of the real-time constraints.

If the customer has received the **Product**, the customer attaches the **Order** and the **Product** to the demand. If the customer has not yet received the **Product**, the customer attaches all messages the customer received from the merchant after sending the **Order**. Figure 4 shows the attachments to be sent for each situation.

Situation	*Attachments*
Customer has received Product	Order, Product
Customer has not received Product	Order, all timestamped messages received from the merchant after Order ($M_1, M_2, ..., M_X$)

Figure 4: Attachments to be sent with the **Demand**

We now see how the merchant evaluates the demand for refund. We use the following notations throughout the rest of this paper.

For the time stamped on messages:
$T_S \equiv$ stipulated time

$T_P \equiv$ time stamped on the **Product**
$T_O \equiv$ time stamped on the **Order**
$T_D \equiv$ time stamped on the **Demand**
$T_1 \equiv$ time stamped on message M_1
$T_2 \equiv$ time stamped on message M_2
$\vdots$
$T_X \equiv$ time stamped on message M_X

For the nonce attached to messages:
$R_P \equiv$ nonce attached to the **Product**
$R_O \equiv$ nonce attached to the **Order**
$R_D \equiv$ nonce attached to the **Demand**
$R_1 \equiv$ nonce attached to message M_1
$R_2 \equiv$ nonce attached to message M_2
$\vdots$
$R_X \equiv$ nonce attached to message M_X

For the number of messages received from the merchant:
$N_P \equiv$ number of messages received before the **Product**
$N_O \equiv$ number of messages received before the **Order**
$N_D \equiv$ number of messages received before the **Demand**
$N_1 \equiv$ number of messages received before message M_1
$N_2 \equiv$ number of messages received before message M_2
$\vdots$
$N_X \equiv$ number of messages received before message M_X

The customer should be given the refund if and only if the **Demand** was sent after the stipulated time elapsed and the **Product** was not received within the stipulated time. The merchant must verify both of these conditions before giving the refund.

Condition 1: The merchant receives the **Demand** and the **Order** from the customer. The merchant verifies if $T_D > T_O + T_S$. If not, it implies that the stipulated time had not elapsed when the merchant sent the **Demand** and customer's **Demand** is denied.

Condition 2: Here, we assume that Condition 1 has already been verified. It is more difficult to verify if the **Product** was not received within the stipulated time. This is because the **Product** may not have been received by the customer at all and thus not attached to the **Demand**, in which case, the time stamped on the **Product** is not available to the merchant. If the **Product** is attached to the **Demand**, the merchant verifies if $T_P > T_O + T_S$. This verifies that the **Product** was received after the stipulated time. If the **Product** was not attached to the **Demand**, the merchant verifies if $N_D - N_O = X$ and $T_1,\ T_2, \cdots,\ T_X > T_O$ and $R_1,\ R_2, \cdots,\ R_X \neq R_O$. This implies that none of the messages received by the customer

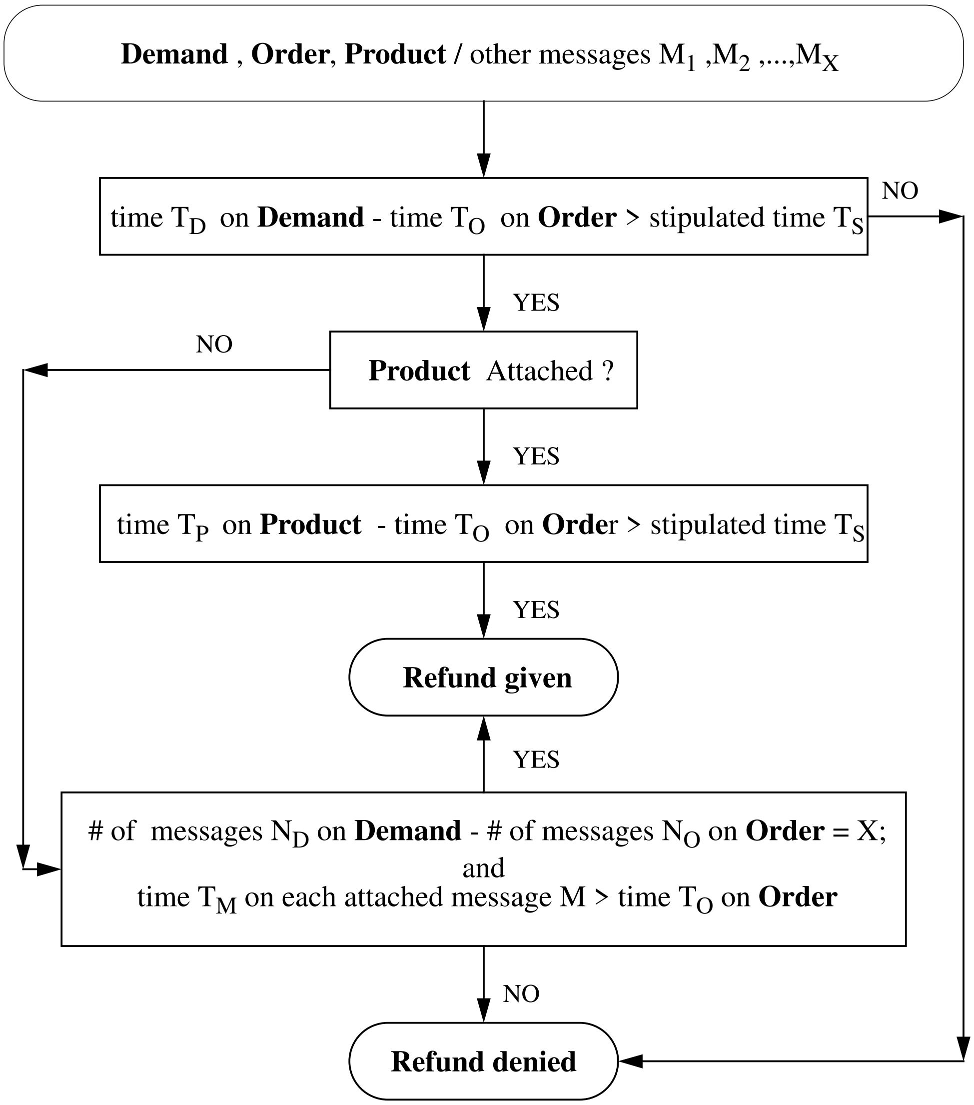

Figure 5: Algorithm to evaluate the customer's **Demand**

after sending the **Order** is the **Product** corresponding to that **Order**, i.e., the **Product** was not received by the customer within the stipulated time.

The above evaluation is further illustrated in Figure 5.

In this subsection we will examine each of the useful properties specified in Subsectio 2.2 and how the methodology affects these properties.

3.3 Conformity with the Requirements

Atomicity: The general methodology described above does not violate the atomicity of the original protocol. Note that the methodology does not remove or change the contents of any of the messages of the original protocol. It only pads the message with some nonces, timstamps, and signatures. Thus the logical flow of the transaction remains the same. The only addition is the detection mechanism, which again does not affect the atomicity of the transaction.

Anonymity: If each customer must reveal the public key of its trusted clock, the merchants can find out the buying pattern of the customer from the transactions that use that key. However, the actual identity of the customer is not revealed. We can incorporate a few different public and private key pairs in the trusted clock and the customer can specify which public key is to be used for each transactions. Thus, a customer can maintain different pseudo-identities that can not be connected to one another. This assures minimal anonymity as required in Subsection 2.2.

Security: Some messages in the real-time aware protocol are unchanged from the original protocol. Let us examine the messages that are modified and see if they become more vulnerable to attacks or if the attached timestamps are now vulnerable to attacks.

The **Order** and the **Product** in the original protocol are secured or signed in the real-time aware protocol and are themselves not more vulnerable to attacks than their counterparts in the original protocol. We examine various attacks on the attached timestamp below.

A customer, merchant, or intruder may try to forge the timestamp on a message. Since, the clock signs every timestamped message with its private key, and private key signatures cannot be forged, this is not a threat to our protocol. Signatures also preserve the integrity of a signed message. Therefore, a malignant party cannot modify the timestamp in a signed message. Nor can anyone tamper with the trusted clock itself. Secure co-processors completely erase their RAM and CPU registers if they are accessed in any method apart from the defined interfaces. Furthermore, replay attacks are easily detectable from the timestamp on the message. The receiver can simply store an array with the timestamp on the latest message received from each sender; if the timestamp in the new message received is not greater than the timestamp stored in the array, the new message is a replay attack. Thus, messages may not be forged, modified or replayed to anyone's advantage.

The customer can, however, have **Product** rerouted through a secret channel and prevent it from reaching the trusted clock. That way, the customer gets the **Product** and can also demand the refund. To prevent this, the merchant secures **Product** with the public key of the receiving trusted processor. Now, the customer cannot receive the product without first having it deciphered by the trusted clock.

The merchant may also deny that he received a **Demand** or this demand may genuinely be lost in the network. Such a denial will be futile since the timestamp records on the trusted clock will show that either the product has still not been received or that it was received after the stipulated time elapsed.

Privacy: The privacy of the messages are not violated by making the protocol real-time aware. This is because none of the messages is transmitted in a less secure form than before. The timestamp on a message may be readable - but it reveals no useful information. The only information revealed is the current time and the number of message received by an unidentified customer from a certain merchant. Even this is readable only in the **Order** or the **Demand**. If the customer wishes for this information to be private, it can be achieved with the merchant's public key or included in the privacy mechanism provided by the original protocol for other fields of the **Order**.

Low Overhead Cost: The real-time aware protocol incurs the extra cost in terms ofq the secure co-processor, the encryptions, the signatures and the **Demand** message. These costs are generally nominal and are justified in most protocols since without these the customer may incur large financial losses. There are some protocols that are specifically designed for low cost transactions, where the payment is made for many transactions together. If (and how) these protocols can be made real-time aware is still an open question.

3.4 The Special Case (no Product delivery)

We now consider the type of transactions where real-time execution is only critical when the transaction is being executed at the merchant's end (Y in Figure 1). This may occur when the merchant is providing a real-time service (such as sale of some shares) that need not be confirmed in real-time or when the user is satisfied to know that the merchant dispatched the goods on time and is ready to incur the losses associated with network delays. Furthermore, in such transactions, if we include the network communications delays (for sending the **Order** or receiving the **Product**) in the stipulated time, the window for the actual execution of the transaction at the merchant's end is reduced.

Such transactions can be handled using a different methodology and architecture that reduces the costs involved at the customer's side. In the new architecture, the trusted clocks are placed in close proximity to the merchant. This not only reduces the cost associated with installing a secure co-processor near every customer, it provides added anonymity as the public key of the clock is now associated with the merchant and not the customers.

The same two messages are identified and modified here. The modification to the original protocol is shown in Figure 6.

Old Msg	*New message sent to clock*	*New Timestamped Msg*
Order	Order, S_K(Order, R)	Order, $S_{\frac{1}{K}}$(Order, T_1, R, N_1)
Product	Product, R	Product, $S_{\frac{1}{K}}$(Product, T_2, R, N_2)

Figure 6: Modification to the Transaction Protocol

In the detection mechanism the customer only sends a **Demand** with the **Order** attached to it. The **Demand** is sent when the customer believes that the confirmation of service was not dispatched on time. The customer may decide to sent a demand after receiving the **Product** and examining the timestamp, or when the customer estimates that the **Product** (confirmation of service) should have arrived but it has not.

The merchant must show that the **Demand** was sent too early ($T_D < T_O + T_S$) or that the **Product** was dispatched on time ($T_P < T_O + T_S$). The merchant can look up T_P from the copy of the **Product** sent to the merchant by the secure clock for the merchant's records. If the merchant is unable to establish either of the above two conditions, the merchant has to refund/compensate the customer for the losses.

The arguments for preserving the useful security properties of the original protocol remains the same as in Subsection 3.3.

4 Application of the General Methodology to the NetBill Protocol

In this section we use an existing well known electronic commerce protocol, namely, NetBill [ST95] to illustrate the general methodology. We will see how our methodology works when applied to an existing operational electronic commerce protocol and how it preserves the useful properties of this protocol.

4.1 The NetBill Protocol

The NetBill protocol was designed at the Carnegie Mellon University. The emphasis in this protocol is on the atomicity of transactions, but the protocol also ensures security, anonymity, privacy and incurs only nominal overhead charges. We describe the protocol briefly below without going into details of how these properties are ensured. Let us, for the purpose of this discussion, just assume that the above mentioned properties are ensured.

We use the notation "X $\Rightarrow$ Y Msg" to indicate that X sends the specified message Msg to Y. The parties involved in the protocol are the customer C, the merchant M and the NetBill server N. The NetBill protocol consists of the following steps [Tyg96]:

1.	C $\Rightarrow$ M	Price request
2.	M $\Rightarrow$ C	Price quote
3.	C $\Rightarrow$ M	Goods request
4.	M $\Rightarrow$ C	Goods, encrypted with key K
5.	C $\Rightarrow$ M	Signed Electronic Payment Order (EPO)

6.	M $\Rightarrow$ N	Endorsed EPO (eEPO), including K
7.	N $\Rightarrow$ M	Signed eEPO, including K
8.	M $\Rightarrow$ C	Signed eEPO, including K

4.2 The General Methodology Applied to the NetBill Protocol

Modification to NetBill

We first identify **Order** and **Product** in the above protocol. **Order** corresponds to Step 5, in the NetBill protocol. Before Step 5, the customer can simply abort the transaction. However, Step 5 is where the customer makes a binding agreement to buy the goods. **Product** corresponds to Step 8 since the customer can use the product only after receiving the key K in Step 8.

When modified using the general methodology, the NetBill protocol looks as shown below. Here we add one extra party - the secure clock, X. Note that we also modify Step 1 and Step 2 to establish the time period within which the goods must be delivered. Note that Step 5 and Step 6 below are two phases of Step 5 in the original protocol; and Step 9 and Step 10 are also two phases of Step 8 in the original protocol. The first phase corresponds to the hop from the sender to the secure clock and the second phase to the hop from the secure clock to the receiver.

1.	C $\Rightarrow$ M	Price and time period request
2.	M $\Rightarrow$ C	Price and time period quote
3.	C $\Rightarrow$ M	Goods request
4.	M $\Rightarrow$ C	Goods, encrypted with key K
5.	C $\Rightarrow$ X	Signed Electronic Payment Order (EPO), R
6.	X $\Rightarrow$ M	EPO, T_1, R, N, $S_{\frac{1}{K_X}}$(EPO, T_1, R, N)
7.	M $\Rightarrow$ N	Endorsed EPO (eEPO), including K
8.	N $\Rightarrow$ M	Signed eEPO, including K (SeEPOK)
9.	M $\Rightarrow$ X	SeEPOK, R, S_{K_X}(SeEPOK, R)
10.	M $\Rightarrow$ X	SeEPOK, T_2, R, N+1, $S_{\frac{1}{K_X}}$(SeEPOK, T_2, R, N+1)

Detection Mechanism

If the customer has received the message in Step 10 and $T_2 - T_1$ is greater than the time quoted in Step 2, the customer sends a demand for refund with copies of the messages in Step 6 (the clock X sends a copy of Step 6 to the customer for the customer's records) and Step 10.

If the customer has not received the message in Step 10, and the time elapsed (as per the customer's local clock) from Step 5 is greater than the time quoted in Step 2, the customer sends a demand for refund with all the timestamped messages received from the merchant with timestamp greater than T_1.

From this information the merchant can evaluate (see Figure 5) if the customer is entitled to a refund. The merchant sends a refund if the customer is entitled to one.

4.3 Properties

Let us examine the properties of the enhanced protocol (the NetBill protocol with a real-time violation detector) in comparison to the properties of the original NetBill protocol.

Atomicity: The NetBill protocol ensures transaction atomicity. In the NetBill protocol, when the merchant cashes the EPO (Electronic Payment Order) sent by the customer, the NetBill server records the product key and the product information. If the customer claims that the key was not delivered, the NetBill server can supply the key from its records. Also, if the customer claims that the wrong key was sent, the NetBill server can get the encrypted product from the customer, decrypt it and verify if the product matches the product description recorded by the server. If not the NetBill server reverses the payment made to the merchants account.

In the real-time aware protocol, atomicity is preserved. The customer has a copy of the EPO sent (Step 6) and the proof (as shown in the demand for refund) that the goods have not been delivered on time. With this information, the customer can directly ask the NetBill server to reverse the payment.

Security: As discussed earlier in Subsection 3.3, the security of the original protocol is preserved. The extra information sent (the timestamps on some messages and the demand for refund) are secure from any tampering.

Privacy: All messages are kept intact. The timestamps attached are never transmitted over the network in plain text. They are in a readable signature, but the intruder has no obvious means of knowing the public key of the secure clock. If the customer wishes to guard against the odds that the timestamp may be read by an intruder, the customer can use the methods suggested in Subsection 3.3 to do so.

Anonymity: The NetBill protocol only provides minimal anonymity. The customer is allowed to operate using different pseudonyms. The same anonymity is ensured by the enhanced protocol by using different pairs of key at the trusted clock. This has been previously discussed in Subsection 3.3.

Low overhead cost: The justification for the extra cost incurred is given in Subsection 3.3.

5 Conclusion

In this paper, we have developed a methodology for making existing secure electronic commerce transaction protocols real-time aware. The methodology consists of a modification to the original protocol by adding some timestamps and augmenting the original protocol with a detection mechanism. We have also proposed a methodology for a special case of real-time applications where the network communications between the merchant and the customer are not included in the real-time constraints. We illustrated the general methodology using an existing electronic commerce protocol, namely, the NetBill protocol. We showed that our methodology preserves useful security properties of existing electronic commerce protocol, namely, security, atomicity, anonymity, privacy, nominal overhead cost. As a result, the methodology is applicable to most of these existing protocols. Our methodology has significant potential to bring important real-time commerce applications to the electronic front and to improve the real-time characteristics of those applications that have already become electronic.

In this paper, we focussed on a general methodology and have explored only one special case where network delays are excluded from the real-time constraints. Other special cases of the general methodology may be explored - especially the case where the customer's order (**Order**) over the network is included in the real-time constraints, while the **Product** is not. We are also working on extending the methodology to protocols that have a single payment phase for multiple transactions (these protocols emphasize largely on low overhead cost and give lesser importance to other security aspects).

References

[And94] R. Anderson. Why Cryptosystems Fail. In *Communications of ACM*, volume 37, 1994.

[B+94] J. P. Boly et al. The ESPRIT Project CAFE - High Security Digital Payment Systems. In *ESORICS '94*, 1994.

[BGH+95] M. Bellare, J. A. Garay, R. Hauser, et al. iKP- A Family of Secure Electronic Payment Protocols. In *Proceedings of the First USENIX Workshop on Electronic Commerce*, 1995.

[Cyb] CyberCash. The CyberCash System - How it works. Available at http://www.cybercash.com/cybercash/cyber2.html.

[Dig] DigiCash. About Ecash. Available at http://www.digicash.com/ecash/ecash-home.html.

[Hic95] K. E. B. Hickman. Secure Socket Library. 1995. Available at http://www.mcom.com/info/SSL.html.

[Man95] M. Manasse. The Millicent Protocols for Electronic Commerce. In *Proceedings of the First USENIX Workshop on Electronic Commerce*, 1995.

[NM95] B. C. Neuman and G. Medvinsky. Requirements for Network Payment: The NetCheque Perspective. In *Proceedings of IEEE COMPCON'95*, 1995.

[SSBR94] L. H. Stein, E. A. Stefferud, N. S. Borenstein, and M. T. Rose. The Green Commerce Model. 1994. Available at http://www.fv.com/tech/green-model.html.

[ST95] M. Sirbu and J. D. Tygar. NetBill: An Internet Commerce System. In *IEEE COMPCON*, 1995. Available at http://www.ini.cmu.edu/netbill/CompCon.html.

[Tyg96] J. D. Tygar. Atomicity in Electronic Commerce. In *PODC*, 1996.

[WW+91] S. R. White, S. H. Weingart, et al. Introduction to the Citadel Architechture: Security in Physically Exposed Environments. In *Technical Report, Distributed Security Systems Group, IBM T. J. Watson Research Center*, 1991.

[Yee94] B. S. Yee. Using Secure Coprocessors. In *Ph.D. thesis, Computer Science Technical Report CMU-CS-94-149, Carnegie Mellon University*, 1994.