Available online at www.sciencedirect.com
SCIENCE DIRECT®
Decision Support Systems 39 (2005) 267–292
Decision Support Systems
www.elsevier.com/locate/dsw

An anonymous and failure resilient fair-exchange e-commerce protocol

Indrajit Ray[a,*], Indrakshi Ray[a], Narasimhamurthi Natarajan[b]

[a] *Computer Science Department, Colorado State University, 217 USC, 601 S Howes Street, Fort Collins, CO 80523-1873, USA*
[b] *Electrical and Computer Engineering Department, University of Michigan-Dearborn, USA*

Received 10 December 2002; received in revised form 15 October 2003; accepted 16 October 2003
Available online 16 December 2003

Abstract

In an electronic commerce environment, the merchant and the customer are unlikely to trust each other. This problem has motivated researchers to propose fair-exchange protocols based on using an on-line trusted third party; the third party receives the items being exchanged from the customer and the merchant and then forwards it to the other party in a fair manner. However, 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. In this paper, we propose an optimistic protocol in which the trusted third party is invoked only if any party misbehaves or prematurely aborts. The protocol achieves fairness and dispute resolution is performed automatically within the scope of the protocol. We show how we can distribute the function of the trusted third party across several third parties; this increases the robustness of the protocol. Additionally, we show how by adopting a payment mechanism based on electronic cash, we provide anonymity to the customer's transactions.

Keywords: Anonymity; Electronic commerce; Fair-exchange; Protocol; Security

1. Introduction

E-commerce transactions, specially those that involve the exchange of digital products between the transacting parties, have stronger requirements as compared to classical brick-and-mortar transactions. In the classical business environment, a transaction involves fulfillment of a contract between two parties; the contract describes the penalties if either party fails to meet its obligation. Since each transacting party has an identifiable place of doing business, any party that behaves unfairly in the transaction can be physically approached and held accountable for its unfair behavior. On the other hand, in an e-commerce environment that deals with only digital products, a party does not always have a physically identifiable place of doing business. After behaving unfairly in the e-commerce transaction, a party can simply vanish without trace. In such a case, it may be next to impossible to enforce the penalties of the contract. Consequently, in an e-commerce environment, the two parties are reluctant to trust each other.

Owing to this lack of trust, e-commerce protocols need to be carefully designed to prevent unfair busi-

* Corresponding author. Tel.: +1-970-491-7097; fax: +1-970-491-2466.
E-mail address: indrajit@cs.colostate.edu (I. Ray).

0167-9236/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2003.10.011

ness dealings by any party involved. However, it is not a simple proposition. Consider the following transaction. A customer, C, contacts an on-line merchant, M, for a product, *m*. The product is an electronic database. Now the customer is not willing to pay for the product without being sure it is the right database sent by the merchant. A merchant is not willing to give the database unless he is sure that he will receive proper payment for it. If the merchant delivers the product prior to receiving the payment, the fraudulent customer may simply disappear after getting the product, causing loss for the merchant. If, on the other hand, the customer pays before receiving the product, the merchant may not deliver or may deliver some wrong product. Thus, unless one of the parties takes the risk and exchanges its product first, the protocol will never continue to completion.

Researchers have proposed secure protocols that address this problem in various degrees. These protocols ensure that no player in an e-commerce transaction can gain an advantage over the other player by misbehaving, misrepresenting or by prematurely aborting the protocol—that is, the protocols ensure *fair exchange*. Fair-exchange protocols have been variously studied in the context of exchange of electronic mails, exchange of digital signatures, exchange of documents (where the consistency of the documents need to be verified before the exchange) and in the context of electronic payment for services. In electronic payment systems, fair exchange is often referred to as ''goods atomicity''—a merchant receives payment if and only if the customer receives the product [9].

Most fair-exchange protocols rely on gathering evidence during the protocol execution, that can be used later for dispute resolution in a court of law. The dispute resolution phase is not a part of the protocol. After the protocol is completed, a human judge looks at the evidence and delivers his judgment. Researchers call such protocols ''weak fair-exchange'' protocols [1]. These protocols try to emulate conventional business transactions. However, in an e-commerce environment, where any of the players can disappear quickly, without any trace, such after-the-fact protection may be inadequate.

To solve this problem, researchers have proposed ''strong fair-exchange'' protocols [1] that rely on on-line trusted third parties to ensure that that either both parties receive each other's item or none do. The trusted third party receives the information from each party 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. However, the third party is a source of bottleneck for these protocols and a single point of failure. Not only is the performance of the third party an issue, but also its vulnerability to hacking activities and other denial of service attacks.

The above factors motivate us to propose an e-commerce protocol that is suitable for business transactions involving digital products and that satisfies the following goals. First, the protocol must provide fairness under all circumstances. That is, fairness should not be compromised if any party misbehaves or prematurely aborts. Bao et al. [3] term this as the *loss-preventing* property. Second, the protocol should not require any manual dispute resolution in case any party behaves unfairly. Third the protocol should not rely on the availability of a single trusted third party. Rather it should use a number of such parties, a quorum of which should always be able to provide the necessary service. Fourth, the interactions with the third parties must be kept to a minimum level. Ideally, the third parties should be invoked only in case of a problem. Fifth, the protocol should allow the exchange of any digital goods and not be restricted to specific applications, such as, digital signatures. In particular, it should allow the exchange of some value over the network—for example, digital money.

E-commerce protocols additionally raise several privacy issues for the customer. First and foremost, payment for such services often requires the customer to provide considerable personal information to the merchant which may be of a sensitive nature; for example, if the payment is by credit card, the credit card number has to be disclosed to the merchant; if the payment is by electronic fund transfer, then information about the customer's financial institution, account number, etc. are disclosed to the financial institution and the merchant. The customer may choose not to disclose such information to the merchant (although the customer probably does not have much option but to disclose the same to the financial institution.). Moreover, it is astonishingly easy for the merchant (or the financial institution) to monitor the on-line activities of the customer that can be mined to

create a customer profile. Such a profile can be used later for profit by the merchant or the financial institution in conjunction with the merchant, and the customer may not really want this to occur. Finally, the customer may just want his/her purchasing habits be not available to others (for example, the customer may be purchasing adult content over the Internet). These factors motivate us to propose an anonymous protocol that allows the customer to use a pseudo identity for a transaction and pay for the product in such a manner that the merchant is prevented from linking a transaction to its source and the financial institution is prevented from linking a customer's transaction to the corresponding product and/or the merchant.

The protocol we propose has application in e-commerce environments where the products exchanged are digital in nature. The merchant sells a digital product that can be delivered to the customer in the form of a message over a network. Some examples of such products are daily news reports, stock quotes and trading information, digital music and movies (Apple Computer recently announced the formation of such a service for selling music over the Internet—see http://www.apple.com/music/). The customer pays for the product also by sending a message; the message contains some kind of a fund transfer information from one financial institution to another—for example, electronic checks, credit cards information, etc. As mentioned earlier, fairness can be compromised very easily in such environments. This is because none of the participants need to have a physically identifiable place to conduct the business from. Our protocol is perfectly suitable in such scenarios. No other protocol that we know of satisfies all these goals.

The proposed protocol is based on some of our previous work reported elsewhere [21]. The current protocol significantly extends the earlier work. It addresses the problem of anonymity of the customer and optionally that of the merchant by incorporating a new payment protocol. Also, it addresses the problem of failure and compromise of the trusted third party by distributing the functionality of the third party across several entities. We organize the rest of the paper as follows. In Section 2, we discuss some of the previous works in fair-exchange protocols and compare them with our approach. Section 3 lays the theoretical foundation on which our protocol is built. We present the initial protocol in Section 4. The handling of misbehaving players is addressed in Section 5. Section 6 provides an analysis of how the protocol satisfies the fundamental properties of e-commerce protocols. Next, in Section 7, we describe how we can increase the robustness of the protocol by using a set of trusted third parties rather than one single trusted third party. Section 8 addresses the issue of anonymity of the customer. To achieve anonymity, we borrow the notion of untraceable electronic cash from the Digicash protocol [8] and integrate the payment-by-digital-coin feature of Digicash into our protocol. We demonstrate in Section 10 that the new protocol satisfies all the desirable properties of e-commerce protocols and additionally provides anonymity to the customer. Finally, we conclude the paper in Section 10 by addressing some of the limitations of the protocol and describing the work in progress to address those issue.

2. Related work

Previous work on fair-exchange schemes can be classified under two categories: (i) gradual exchange protocols and (ii) third party protocols. In this section, we briefly describe some of the important works and compare them with our work.

2.1. Gradual exchange protocols

Gradual exchange protocols like the ones proposed by various researchers [4,5,11,22], gradually increase the probability of fair exchange over several rounds of message exchanges. These protocols have extensive communication requirements and majority of the protocols assume that both the players have equal computational power.

The protocol presented by Blum [5] provides a mechanism by which two players can exchange secrets. The secrets are such that they are prime factors of the players' publicly announced composite numbers. The two players exchange their respective secrets bit by bit, alternately. For each bit provided to the adversary, a player has to prove that the bit is good, that is, it is part of the secret. The author shows how the protocol can be used in conjunction with

digital signatures to sign contracts and send certified emails.

Even et al. [11] propose the notion of a *1-out-of-2 oblivious transfer protocol*. The authors define a message to be a "recognizable secret message" if, although the receiver cannot compute the message, he/she can authenticate it once received. An *oblivious transfer* of a recognizable secret message is a protocol by which a sender transfers a message to the receiver so that the latter gets the message with a probability of 0.5, while for the sender, the a posteriori probability that the receiver got the message is 0.5. A special case of the oblivious transfer protocol is the 1-out-of-2 oblivious transfer protocol by which the sender is able to transfer exactly one secret out of two recognizable secrets. Using the 1-out-of-2 protocol as the basis, the authors propose protocols for contract signing, certified mail and coin flipping. In this protocol, like the one by Blum [5], the two the players exchange the items one bit at a time.

Ben-Or et al. [4] provide an approach in which each party gradually releases information that incrementally increases the probability that a fair exchange is valid. This probability approaches one after several rounds of message exchanges. This protocol, unlike that proposed by Blum [5], does not require both players to have equal computational power.

One of the major shortcomings of the protocols described by researchers [4,5,11] is that they lack in simultaneity of the exchange, and consequently, they are not suitable for electronic commerce systems that exchange some value over the network—for example, digital money. If midway through the execution of any of these protocols, one of the players decide to stop the exchange, then it is possible that that player will hold an unfair advantage over the other player. Such midway, unilateral termination of an exchange may be quite possible in real life. For example, the transaction may seem profitable to a player when viewed ex ante. However, during the course of the transaction, some event occurs that modifies the perception of the player about the transaction. Our protocol, on the other hand, ensures fair exchange even if a player aborts midway through the transaction. Although we propose the protocol in the context of exchange of value, the protocol can equally well be used for fair exchange of any other digital commodity.

Sandholm and Lesser [22] choose a game theoretic approach in the context of automated negotiation systems, to motivate the players to behave fairly in the transaction. The authors propose a leveled commitment contracting protocol that allows any player to pay a penalty and withdraw from a contract due to some unexpected event happening in the course of the transaction. This ensures that no player has unfair advantage over the other player at any point in the protocol. However, the problem with this approach is that the protocol assumes that both players behave rationally during the protocol execution. For e-commerce transactions over the Internet, this is, we believe, too strong a requirement. Our protocol imposes no such requirement.

2.2. Third party protocols

The third party protocols, as proposed by the researchers [9,10,13,26], each uses a trusted on-line third party. The idea of using a trusted on-line third party to obtain non-repudiation of origin and delivery of an email message was proposed by Deng et al. [10] and Zhou and Gollmann [26]. These protocols are essentially similar. They differ in what information is exchanged and how the information gets transferred from one party to the other. The basic idea is as follows. When A wants to send a message to B, A encrypts the message with a key, and sends B the encrypted message and a trusted third party the key. B after submitting his proof of delivery can get the key and read the message. In these protocols, the dispute resolution is outside the scope of the protocol. However, the protocols do specify what evidence must be stored and how they must be collected for the dispute to be resolved in a fair manner. Our work, which addresses the fair-exchange problem in the context of electronic transactions, automatically does the dispute resolution within the scope of the protocol itself and without requiring any human intervention.

The NetBill system [9] is one of the earliest protocols to provide a complete solution to the problem of selling and delivering electronic goods. Our approach is quite similar to the NetBill system and so we discuss it in some details here. The NetBill system uses a trusted third party called the NetBill server. The NetBill server maintains accounts for both the customer and the merchants, and is linked with conventional financial

institutions. The basic NetBill protocol is as follows. The customer requests the merchant for the price of an item. The merchant sends the price quote. The customer then requests the merchant for the goods. The merchant sends the goods encrypted with a key. Upon receipt of this encrypted product, the customer supplies the merchant with a signed electronic purchase order. The electronic purchase order contains a segment that has payment information. This portion is readable only by the NetBill server. The merchant endorses the electronic purchase order, and forwards it to the NetBill server together with the decrypting key. The NetBill server debits the customer's account and credits the merchant's account and then sends a signed message to the merchant that includes the result of the transaction and an encrypted receipt intended for the customer. The encrypted receipt contains the decrypting key, and the status of the customer's account after the transaction. The receipt can be read only by the customer. The merchant forwards the encrypted message to the customer to complete the transaction. If, for some reason, the merchant does not deliver the receipt, the customer gets it from the NetBill server.

The goal of our protocol is similar to that of NetBill. In NetBill, a merchant who has provided a worthless good is detected only after the exchange occurs. The protocol gathers evidence during protocol execution; any dispute resolution is handled manually and is outside the scope of the protocol. With our protocol, on the other hand, the customer can verify whether the merchant is delivering the product promised, before the customer actually makes the payment. The customer is guaranteed to receive the product for which he is paying; the merchant is guaranteed to receive payment for the product he has sold. Thus, fair exchange is ensured. Further, any dispute resolution can be handled automatically by the trusted third party without resorting to manual intervention. The dispute resolution involves both player receiving each other's product.

A fair-exchange protocol that ensures the consistency of the document has been proposed by Ketchpel [17]. The basic protocol is as follows. After agreeing upon the product and the price, the customer and the merchant sign a contract which is forwarded to the third party. The customer sends the payment to the third party and the merchant sends the required product to the third party. The third party verifies that the product and payment satisfy the terms of the contract and then forwards the product to the customer and the payment to the merchant. Thus, the third party plays an active role in this protocol. Our protocol, in contrast, reduces the involvement of the third party. Further, the third party in our protocol is not a source of bottleneck. We use multiple third parties, a coterie of which can resolve disputes and ensure fairness.

Franklin and Reiter [13] also propose a set of fair-exchange protocols that verify the consistency of a document before the exchange takes place. These protocols require a semi-trusted third party. A semi-trusted third party is one that can misbehave on its own but will not collude with any of the participating parties. The protocols use a one-way function f which has the additional property that there exists another efficiently computable function F such that $F(x, (f(y)) = f(xy)$. The function, f, is known by both the parties, and F is known by the third party.

The basic protocol is as follows. Suppose X and Y wish to exchange some secret information K_X and K_Y. Before the protocol is initiated, it is assumed that X and Y know $f(K_Y)$ and $f(K_X)$, respectively. The first step involves X sending a random number x_1 (which is in the domain of f) to Y, and Y sending a similar random number y_1 to X. In the second step, X sends the following to the third party: $f(K_X)$, $f(K_Y)$, $K_X x_1^{-1}$ and $f(y_1)$ and Y also sends the corresponding components to the third party. The third party makes some comparisons to ascertain that each is sending the correct components, and then forwards $K_X x_1^{-1}$ to Y and $K_Y y_1^{-1}$ to X. Y and X can multiply these by x_1 and y_1 respectively to get the respective keys.

One contribution of this paper is that the third party is semi-trusted and the information that X and Y are trying to exchange is never revealed to the third party. Our protocol, in contrast, requires the third party to be trusted. Franklin's protocol requires an active involvement of the third party for all scenarios. We resort to the third party only when no one misbehaves or aborts.

Three fair-exchange protocols that do not require the involvement of the third party unless there is a problem have been proposed by Bao et al. [3]. The first one exchanges digital signatures on some document, the second one exchange signatures on two documents, and the third one exchanges a document and a signature on the document. The important contribution of this paper is that the authors provide a theory based on

which each party is able to verify that the signature he is about to receive is indeed the correct signature, before actually receiving the signature. However, the protocol is not general, and does not apply when both the parties want to exchange items of some value over the network other than digital signatures. Asokan et al. [2] also provides an optimistic protocol that deals with the fair-exchange of digital signatures.

A more general protocol that allows exchange of any two digital items has been proposed by Asokan et al. [1]. This protocol does not involve the third party unless one of the parties behaves unfairly or aborts. The protocol begins by the two parties promising each other an exchange of items. If they do not agree on the terms of the exchange, the protocol is aborted. Otherwise, the exchange takes place. The items as well as non-repudiation tokens are exchanged. In case of any failure or any party misbehaving, the recovery phase is initiated. The authors assume there is a reliable communication channel between each party and the third party. Hence, all the messages exchanged in the recovery phase uses these reliable channels via the third party. When any party misbehaves, the third party can issue an affidavit which can be used in a court of law in case of a dispute. Non-repudiation of origin and non-repudiation of receipt is guaranteed by these protocols. The protocol always guarantees weak fairness, that is, an honest party can prove his case in case of a dispute.

The main similarity with our work is that the third party is not invoked unless there is a problem. However, there are a number of significant differences with our work. The most significant difference is that our protocol always ensures strong fairness, whereas the protocol proposed by Asokan et al. ensures only weak fairness. The other significant difference is that dispute resolution is outside the scope of Asokan et al.'s protocol. The honest party can prove his case, but how he gets compensated or the dishonest party punished is not elaborated. In our protocol, the honest party is always compensated by receiving the correct item. This is taken care of by the protocol itself—no manual intervention is necessary. A third difference is that in Asokan's protocol both the parties exchange descriptions of the item before exchanging the items. Once an item is received, the receiving party checks whether the item matches the description or not. In our work, each party is able to verify that the item he is about to receive is the one he expects, before exchanging the items. Another significant difference is that anonymity of the customer is not preserved in Asokan's protocol. Our protocol does so. Last but not the least, in Asokan's protocol, the third party is a source of bottleneck; in ours, it is not.

3. Theory of cross validation

Our protocol uses a novel approach based on "RSA like" cryptography to provide fair exchange. We term this approach "cross validation". In this section, we establish the theory.

We begin by assuming that the customer buys a digital product and pays the merchant using a digital payment token. (Later on, in Section 8, we show how we modify the protocol so as to replace the payment token with electronic cash.) For confidentiality as well as integrity purposes, the digital product is encrypted while in transit. The customer needs to get a decryption key in order to decipher the encrypted product. The encryption is assumed to be sufficiently strong that, without the decryption key, the encrypted product is useless to the customer. The customer receives the decryption key if and only if the merchant receives payment for the product. Thus, in order to validate the product, the customer should be able to recognize the contents of the encrypted product, without requiring to decrypt it. Note that using cryptographic checksums does not solve the problem of validation. There can be two ways checksums can be added to the delivered product.

1. The merchant sends a cryptographic checksum of the actual product (i.e. the cryptographic checksum computed before encryption). In this case, the customer has no way of ascertaining that the checksum corresponds to the actual product without first obtaining the actual product.
2. The merchant sends a cryptographic checksum of the encrypted product. This can at most guarantee that the customer has received the encrypted product correctly, without any error. This does not provide any protection if the merchant sends the wrong product.

With the above background information, we now present the theory of cross validation.

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.

$$\mathcal{M}\{m \mid 0 \leq m < N\} \quad (1)$$

Definition 2. Given an integer a and a positive integer N, the following relationship holds,

$$a = qN + r \text{ where } 0 \leq r < N \text{ and } q = \lfloor a/N \rfloor \quad (2)$$

where $\lfloor x \rfloor$ denotes the largest integer less than equal to x. The value q is referred to as the *quotient* and r is referred to as the *remainder*. The remainder r, denoted $a \bmod N$, is also referred to as the *least positive residue* of $a \bmod N$.

Definition 3. For positive integers a, b and N, we say a is *equivalent* to b, modulo n, denoted by $a \equiv b \bmod n$, if $a \bmod n = b \bmod n$.

Definition 4. For positive integers a, x, n and $n>1$, if $\gcd(a, n) = 1$ and $a.x \equiv 1 \bmod n$, then x is referred to as the *multiplicative inverse of a modulo n*.

Definition 5. Two integers a, b are said to be *relatively prime* if their only common divisor is 1, that is, $\gcd(a, b) = 1$.

Definition 6. The integers $n_1, n_2, \ldots, n_k$ are said to be *pairwise relatively prime*, if $\gcd(n_i, n_j) = 1$ for $i \neq j$.

Definition 7. The Euler's totient function $\phi(N)$ is defined as the number of integers that are less than N and relatively prime to N. Below we give some properties of totient functions that we need in this paper.

1. $\phi(N) = N - 1$ if N is prime.
2. $\phi(N) = \phi(N_1)\phi(N_2)\ldots\phi(N_k)$ if $N = N_1N_2\ldots N_k$ and $N_1, N_2, \ldots, N_k$ are pairwise relatively prime.

Theorem 1. *Euler's theorem states that for every a and N that are relatively prime,*

$$a^{\phi(N)} \equiv 1 \bmod N$$

Proof of Theorem 1. We omit the proof of Euler's theorem and refer the interested reader to any book on number theory (see, for example, Ref. [19]) or on cryptography (for example, Ref. [24]). □

Corollary 1. *If $0 < m < N$ and $N = N_1N_2 \ldots N_k$ and $N_1, N_2, \ldots, N_k$ are primes, then $m^{x\phi(N)+1} \equiv m \bmod N$.*

Definition 8. A *key* K is defined to be the ordered pair $\langle e, N \rangle$, where N is a product of distinct primes, $N \geq M$ and e is relatively prime to $\phi(N)$; e is the *exponent* and N is the *base* of the key K.

Definition 9. The *encryption* of a message m with the key $K = \langle e, N \rangle$, denoted as $[m, K]$, is defined as

$$[m, \langle e, N \rangle] = m^e \bmod N \quad (3)$$

Definition 10. The *inverse* of a key $K = \langle e, N \rangle$, denoted by K^{-1}, is an ordered pair $\langle d, N \rangle$, satisfying $ed \equiv 1 \bmod \phi(N)$.

Theorem 2. *For any message m,*

$$[[m, K], K^{-1}] = [[m, K^{-1}], K] = m \quad (4)$$

where $K = \langle e, N \rangle$ and $K^{-1} = \langle d, N \rangle$.

Proof of Theorem 2. We first show that

$$[[m, K], K^{-1}] = m$$

$$\begin{aligned}
\text{L.H.S.} &= [[m, K], K^{-1}] \\
&= [m^e \bmod N, K^{-1}] \quad \text{(from Definition 9)} \\
&= (m^e \bmod N)^d \bmod N \quad \text{(from Definitions 9 and 10)} \\
&= m^{ed} \bmod N \quad \text{(from laws of modular arithmetic)} \\
&= m^{(x\phi(N)+1)} \bmod N \quad \text{(from Definitions 2 and 10, } ed = x\phi(N) + 1\text{)} \\
&= m \bmod N \quad \text{(from Corollary 1)} \\
&= m \quad \text{(since we assume } m < N\text{; see Definition 1)} \\
&= \text{R.H.S.}
\end{aligned}$$

By symmetry $[[m, K^{-1}], K] = m$. □

Corollary 2. *An encryption, [m, K], is one-to-one if it satisfies the relation*

$$[[m,K],K^{-1}] = [[m,K^{-1}],K] = m$$

Definition 11. Two keys $K_1 = \langle e_1, N_1\rangle$ and $K_2 = \langle e_2, N_2\rangle$ are said to be *compatible* if $e_1 = e_2$ and N_1 and N_2 are relatively prime.

Definition 12. If two keys $K_1 = \langle e, N_1\rangle$ and $K_2 = \langle e, N_2\rangle$ are compatible, then the *product* key, $K_1 \times K_2$, is defined as $\langle e, N_1\ N_2\rangle$.

Lemma 1. *For positive integers a, N_1 and N_2,*

$$(a \bmod N_1N_2) \equiv a \bmod N_1$$

Proof of Lemma 1. Let $a = N_1N_2x + N_1y + z$, where x, y and z are integers.

$$\text{L.H.S.} = (a \bmod N_1N_2) \bmod N_1$$
$$= \left(N_1N_2x + N_1y + z - \left\lfloor \frac{N_1N_2x + N_1y + z}{N_1N_2} \right\rfloor \times N_1N_2\right) \bmod N_1 = (N_1y + z) \bmod N_1 = z$$

$$\text{R.H.S.} = (a \bmod N_1)$$
$$= (N_1N_2x + N_1y + z) \bmod N_1 = z$$

Hence the proof. □

Theorem 3. *For any two messages m and $\hat{m}$, such that m, $\hat{m} < N_1, N_2$,*

$$[m, K_1 \times K_2] \equiv [\hat{m}, K_1] \bmod N_1 \text{ if and only if } m = \hat{m} \quad (5)$$

$$[m, K_1 \times K_2] \equiv [\hat{m}, K_2] \bmod N_2 \text{ if and only if } m = \hat{m} \quad (6)$$

where K_1 is the key $\langle e,N_1\rangle$, K_2 is the key $\langle e,N_2\rangle$ and $K_1 \times K_2$ is the product key $\langle e, N_1\ N_2\rangle$.

Proof of Theorem 3. The proof for Eq. (6) is the same as that for Eq. (5). We just consider the proof for Eq. (5).

[**If part**] Given $m = \hat{m}$, we have to prove that $[m, K_1 \times K_2] \equiv [\hat{m}, K_1] \bmod N_1$, that is,

$$[m, K_1 \times K_2] \bmod N_1 = [\hat{m},\ K_1] \bmod N_1$$

$$\text{L.H.S.} = [m,\ K_1 \times K_2] \bmod N_1$$
$$= (m^e \bmod N_1N_2) \bmod N_1 \text{ (from Definitions 9 and 10)}$$
$$= m^e \bmod N_1 \text{ (substituting } m^e \text{ for } a \text{ in Lemma 1)}$$

[**Only If part**] Given $[m, K_1 \times K_2] \equiv [\hat{m}, K_1] \bmod N_1$, we have to prove $m = \hat{m}$

$$[m, K_1 \times K_2] \equiv [\hat{m}, K_1] \bmod N_1$$

$$\text{or } [m, K_1 \times K_2] \bmod N_1 = [\hat{m}, K_1] \bmod N_1 \text{ (from Definition 3)}$$

$$\text{or } (m^e \bmod N_1N_2) \bmod N_1 = (\hat{m}^e \bmod N_1) \bmod N_1 \text{ (from Definitions 9 and 12)}$$

$$\text{or } (m^e \bmod N_1N_2) = (\hat{m}^e \bmod N_1)$$

$$\text{or } m^e \bmod N_1 = (\hat{m}^e \bmod N_1) \bmod N_1 \text{ (from Lemma 1)}$$

$$\text{or } [m, \langle e, N_1\rangle] = [\hat{m}, \langle e, N_1\rangle] \quad \text{(from Definition 9)}$$

$$\text{or } m = \hat{m} \quad \text{(since the encryption is one to} - \text{one)}$$

□

3.1. Validated receipt property

The validated receipt property is stated as follows:

Definition 13. An electronic commerce protocol is said to satisfy *validated receipt* if a customer is able to ensure within the scope of the protocol and before the customer pays for a product, that the product the customer is about to receive from a merchant, is the same as the product the customer intended to purchase.

The *validated receipt* property is desirable for electronic commerce protocols. E-commerce protocols operate in an environment where the customer and the merchant do not trust each other. The presence of

validated receipt will give some confidence to the customer that he will receive the correct product if he makes the payment.

Our protocol achieves the validated receipt property using the results of Theorem 1. Let m be the product to be delivered to the customer. The trusted third party generates keys $K_M = \langle e,N \rangle$ and K_M^{-1} and provides K_M to the merchant. The merchant provides the product m to the trusted third party to be encrypted with K_M and placed at a public place, henceforth called the catalog, as an advertisement for m. When the customer decides to purchase m from the merchant, the customer acquires $T=[m,K_M]$ from the catalog and keeps it for future validation of the product received.

To sell m to the customer, the merchant selects a second set of keys $(K_{M_1}, K_{M_1}^{-1})$ such that $K_{M_1} = \langle e,N_1 \rangle$ is compatible with K_M according to Definition 11. The merchant provides the customer with $T' = [m, K_M \times K_{M_1}]$.

The customer verifies that $[m, K_M]$ and $[m, K_M \times K_{M_1}]$ are encryption of the same message m by verifying: $T \equiv T' \bmod N_1$, as per Eq. (5).

When satisfied, the customer sends the payment token. The merchant, in return, sends the decrypting key $K_{M_1}^{-1}$. The customer obtains m using $m=[T',K_{M_1}^{-1}]$. The proof of correctness follows from Theorem 3:

$$[m, K_M \times K_{M_1}] \equiv [\hat{m}, K_M] \bmod N_1$$
$$\text{if and only if } m = \hat{m}$$

3.2. Security

In the theory presented in Eq. 3, if e is chosen small and a customer can guess e[1], we can have a security problem. Assume that the exponent e is small, say $e=3$. A customer starts as if he is buying the same product m three times, but always stops after having received $[m, K_{M_1} \times K_{M_2}]$, $[m, K_{M_1} \times K_{M_3}]$, $[m, K_{M_1} \times K_{M_4}]$, where $K_{M_2} = \langle e,N_2 \rangle$, $K_{M_3} = \langle e,N_3 \rangle$ and $K_{M4} = \langle e,N_4 \rangle$.

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 $z = m^e \bmod N$. Since $0 < m < N_i$, $m^e < N$ and hence $z = m^e$. By extracting the eth root of z, the customer can get m. Thus, a customer can get the product, without paying for it. Note that this attack is similar to the low exponent attack on the RSA cryptosystem [16]. This mode of attack requires the attacker to try all possible primes less than e. Also, the size of z increases as e increases, thus, making this an infeasible mode of attack when e is sufficiently large.

We provide an additional mechanism using which the security will not be compromised even if the customer can guess e correctly. For every transaction that the merchant performs, the merchant chooses a random number r such that r is relatively prime to N_2. The customer downloads $[m, K_M]$ from the third party. Rather than sending $[m, K_M \times K_{M_1}]$ to the customer, the merchant sends the following: $[mr,K_M \times K_{M_1}]$, $[r,K_M]$, where mr is the product of m with r. To validate the product, the customer multiplies $[m,K_M]$ with $[r,K_M]$ and the resulting product is compared with $[mr,K_M \times K_{M_1}]$. If both match, the customer is confident that the product he is about to receive is the one he is going to pay for. Finally, instead of sending just $K_{M_1}^{-1}$, the merchant now sends $K_{M_1}^{-1}$ and r^{-1} where r^{-1} is the multiplicative inverse of r modulo N_1. Using the decrypting key $K_{M_1}^{-1}$, the customer obtains $mr \bmod N_1$. Multiplying this by r^{-1} the customer can retrieve m.

4. The fair-exchange protocol

To present our protocol, we begin by defining the terms we use in the protocol description.

4.1. Terminologies

Definition 14. A *third party* is a participant in an e-commerce protocol who is neither a customer nor a merchant but whose involvement in the protocol is important for its proper functioning.

Definition 15. A *trusted third party* is a third party which is relied on not to misbehave or collude with any one of the transacting parties to the detriment of the other.

Definition 16. An electronic commerce protocol satisfies the money atomicity property [25] if money is neither created nor destroyed during the execution of the protocol.

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

Definition 17. A *fair-exchange* electronic commerce protocol is one in which two players exchange items of value in such a manner that no player can gain an advantage over the other by misbehaving, misrepresenting or by prematurely aborting the protocol. In other words, in a fair-exchange protocol, either the exchange takes place completely or does not take place at all. Fair-exchange is a more general term for the goods atomicity property [25]. Goods atomicity ensures that a merchant receives payment if and only if the customer receives the goods. The term goods atomicity is used in the context of e-commerce, whereas fair-exchange is used in a more general sense.

Definition 18. An *optimistic fair-exchange* protocol is a fair-exchange protocol that relies on a trusted third party but does not require the active involvement of the third party.

An optimistic protocol assumes that most of the time the players will not misbehave—that is why optimistic. Only when something wrong happens, the third party is contacted to resolve the dispute.

The protocol that we describe here is an optimistic fair-exchange protocol that satisfies the *validated-receipt* property.

4.2. Assumptions

We make the following assumptions in the protocol:

1. We assume that all encryptions are strong enough that the receiver of an encrypted message is unable to decrypt the message without the appropriate key. Similarly, cryptographic checksums are strong enough to ensure integrity of messages and digital signatures are strong enough to provide non-repudiable evidence about the identity of the signatory.
2. All parties use the same algorithm for encryption as well as for generating cryptographic checksums.
3. Payment for product is in the form of a token, PT, that is accepted by the merchant.
4. Each party involved in the transaction keeps a copy of the information that it sends to the other party in its stable storage till such time as the information is no longer needed. Writes to the stable storage are atomic and durable until intentionally purged.
5. A system wide constant time out period known to all parties.

Table 1 lists the notations used in the description of the protocol.

4.3. Protocol description

Before the protocol begins, we assume that the following steps have already executed that sets up the environment in which the protocol operates.

1. *The customer opens an account with a financial institution*. The latter generates a key pair K_{C_1}, $K_{C_1}^{-1}$, provides the customer K_{C_1} and escrows $K_{C_1}^{-1}$

Table 1
Symbols used in protocol description

Symbol	Interpretation
C, M, B and TP	Ids for customer, merchant, customer's financial institution and trusted third party
A_{prv}, A_{pub}	A's private and public keys
A⇒B: X	A sends X to B
$[X,K]$	encryption of X with key K
CC(X)	A cryptographic checksum of X, using an algorithm such as the Secure Hash [18]
ψ_A	A nonce for entity A. Each entity's nonces is unique
K_{C_1}	Key given by the customer's financial institution
$K_{C_1}^{-1}$	The decrypting key corresponding to K_{C1} kept by the financial institution
K_{C_2}	Key generated by the customer that is compatible with K_{C_1}
$K_{C_2}^{-1}$	Decrypting key corresponding to K_{C_2}
K_M	Key given by the trusted third party to the merchant
K_M^{-1}	The decrypting key corresponding to K_{M_1}, held by the trusted third party
K_{M_1}	Key generated by the merchant that is compatible with K_{M_1}
$K_{M_1}^{-1}$	Decrypting key corresponding to K_{M_2}
r	Random number chosen by merchant for current transaction
r^{-1}	Multiplicative inverse of r modulo N_1, where N_1 is the base for key K_{M_1}
m	Product the customer purchases
PID	The id for product m
PO	Purchase order used by the customer to order product m
C_{Act}	Customer's account information with customer's financial institution
PR	Price of the product
PT	Payment token used for paying for the product

with itself. For any financial transaction that the customer performs and that involves payment via this financial institution, the customer is obligated to use product keys $K_{C_1} \times K_{C_2}$, $K_{C_1} \times K_{C_3} \ldots$, and so on, where $K_{Cj}, j \neq 1$, is compatible with K_{C_1}. For this discussion, we will assume that the customer uses $K_{C_1} \times K_{C_2}$ as the product key.

2. *Merchant registers with trusted third party.* The latter generates the key pair K_M, K_M^{-1}, provides the merchant with K_M and escrows K_M^{-1} with itself. For every product, m, that the merchant wants to advertise in the catalog at this trusted third party, the merchant sends m and the description of m containing the product identifier, PID, to the third party. The third party performs the encryption before uploading $[m, K_M]$ on the catalog. In this manner the third party is able to certify that the product meets its claim. To sell a product, m, the merchant sends the customer $[m, K_M \times K_{Mj}]$, where K_{M_j} is compatible with K_M. For this discussion, we assume that the merchant uses K_{M_1} as the compatible key.

The protocol executes in the following five steps when no party misbehaves or prematurely quits. The messages exchanged in the protocol are shown in Fig. 1. Note that only the major contents of each message is shown in the figure.

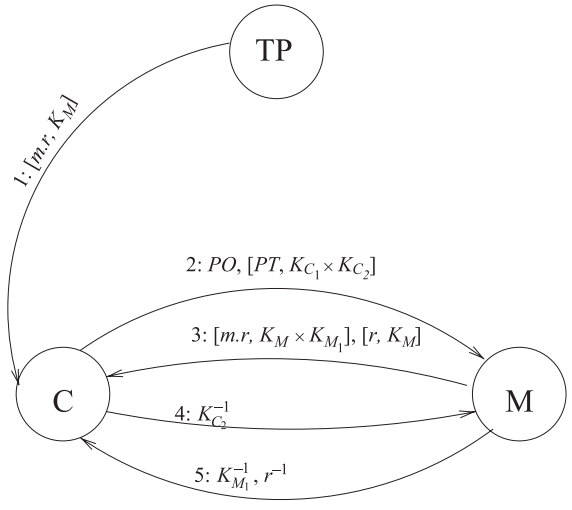

Fig. 1. The basic protocol.

Message 1

$$\mathrm{TP} \Rightarrow \mathrm{C}: \ [m, K_M].$$

The customer downloads $[m, K_M]$ from the trusted third party server together with the product identifier, PID. Note that the customer does not actually have the product m, because he does not have the decrypting key K_M^{-1}. This $[m, K_M]$ will be used later by the customer to validate the product received from the merchant.

Message 2

$$\mathrm{C} \Rightarrow \mathrm{M}: \ \mathrm{PO}, [\mathrm{CC(PO)}, \mathrm{C_{prv}}], [[\mathrm{PT}, K_{C_1} \times K_{C_2}], \mathrm{B_{prv}}], \mathrm{B}.$$

The customer initiates the e-commerce transaction by associating a unique system-wide identifier T_i for the transaction. The identifier is a tuple of the form ⟨PID, C, M⟩. The customer stores a log record of the form ⟨T_i, *INITIATE*⟩ to its stable storage and then sends **Message 2** to the merchant. The purchase order, PO, contains the following information:

(i) the product identifier, PID;
(ii) the customer's identity, C;
(iii) the merchant's identity, M;
(iv) the price of the product, PR; and
(v) a nonce, ψ_C, for the customer.

The customer generates a cryptographic checksum of PO and digitally signs it. The cryptographic checksum forestalls debate over the details of the order, or whether the order was received completely and correctly. The customer's signature forestalls debate over whether the customer expressed intention to purchase the product. The nonce, ψ_C, in the purchase order forestalls a replay of the purchase order with the merchant.

The payment token, PT, contains the following information:

(i) the identity of the customer's financial institution, B;
(ii) the identity of the customer, C;
(iii) the customer's account information, C_{Act} at B;
(iv) the price the customer will pay for the product, PR;
(v) a nonce, ψ_C, for the customer.

The payment token is first encrypted with the *product key* $K_{C_1} \times K_{C_2}$ and then digitally signed by the customer's financial institution. The financial institution's signature ensures that the customer has sufficient fund for payment for the product. Note that, since the financial institution has the key $K_{C_1}^{-1}$, it is able to verify the contents of PT before signing.

In addition, the customer also send the identity of its financial institution, B.

Message 3

$$\mathrm{M} \Rightarrow \mathrm{C}: [[\mathrm{CC}(\mathrm{PO}), \mathrm{C_{prv}}], \mathrm{M_{prv}}], [mr, K_{\mathrm{M}} \times K_{\mathrm{M_1}}], [r, K_{\mathrm{M_1}}], [\mathrm{CC}([r, K_{\mathrm{M_1}}]), \mathrm{M_{prv}}], [\mathrm{CC}([mr, K_{\mathrm{M}} \times K_{\mathrm{M_1}}]), \mathrm{M_{prv}}]$$

OR

$$\mathrm{M} \Rightarrow \mathrm{C}: \text{ Abort}$$

When the merchant receives **Message 2** from the customer, it writes a log record $\langle T_i, \mathit{INITIATE} \rangle$ to its stable storage; then the merchant checks to see if the purchase order is to its satisfaction—that is, the merchant agrees to all its contents and is able to verify the signature of the customer's financial institution on [PT, $K_{C_1} \times K_{C_2}$]. If not, the merchant writes an abort record in its stable storage—$\langle T_i, \mathit{ABORT} \rangle$—and aborts the transaction. It informs the customer of this decision. Otherwise, the merchant endorses the purchase order by signing [CC(PO), $\mathrm{C_{prv}}$]. The merchant sends this endorsed purchase order together with the encrypted product [mr, $K_{\mathrm{M}} \times K_{\mathrm{M_1}}$], its signed cryptographic checksum [CC([mr, $K_{\mathrm{M}} \times K_{\mathrm{M_1}}$]), $\mathrm{M_{prv}}$], the random number, r, encrypted with K_{M1} and its cryptographic checksum. The merchant's endorsement on the purchase order forestalls debate over whether the purchase order was received correctly or not and whether the merchant agreed to the terms of the current transaction. The signed cryptographic checksum [CC([mr, $K_{\mathrm{M}} \times K_{\mathrm{M_1}}$]), $\mathrm{M_{prv}}$] proves the origin of the encrypted product [mr, $K_{\mathrm{M}} K_{\mathrm{M}} \times K_{\mathrm{M_1}}$] as well as ensures the integrity of [mr, $K_{\mathrm{M}} \times K_{\mathrm{M_2}}$] in transit.

Message 4

$$\mathrm{C} \Rightarrow \mathrm{M}: [K_{\mathrm{C_2}}^{-1}, \mathrm{M_{pub}}], [\mathrm{CC}([m, K_{\mathrm{M}} \times K_{\mathrm{M_1}}]), \mathrm{C_{prv}}]$$

OR

$$\mathrm{C} \Rightarrow \mathrm{M}: \text{ Abort}, [\mathrm{CC}([m, K_{\mathrm{M}} \times K_{\mathrm{M_1}}]), \mathrm{C_{prv}}]$$

After receiving **Message 3** from the merchant, the customer checks to see if it is an abort message or the encrypted product. If it is an abort, the customer aborts the transaction and writes a log record of the form $\langle T_i, \mathit{ABORT} \rangle$. Otherwise the customer validates the product as outlined in Section 3.2. If the two compare, the customer sends the payment token decryption key, $K_{\mathrm{C_2}}^{-1}$ to the merchant, encrypted by the merchant's public key and a signed cryptographic checksum of the encrypted product received. The customer then writes a log record to its stable storage. The log record is of the form $\langle T_i, \mathit{PAYMENT\text{-}SENT} \rangle$. Finally the customer starts a timer, waiting for the product decryption key to arrive from the merchant. If the timer expires before the product decryption key arrives from the merchant, the customer executes the extended protocol. If, on the other hand, the product is not validated the customer can request the product from the merchant once more, or abort the transaction. For the latter the customer enters $\langle T_i, \mathit{ABORT} \rangle$ in its log and sends an abort message to the merchant together with a signed cryptographic checksum of the product received from the merchant. The signed cryptographic checksum of the encrypted product forestalls debate over whether the encrypted product received by the customer is the same as the encrypted product sent by the merchant.

Message 5

$$\mathrm{M} \Rightarrow \mathrm{C}: [K_{\mathrm{M_1}}^{-1}, \mathrm{C_{pub}}], [r^{-1}, \mathrm{C_{pub}}]$$

Once the merchant receives the decrypting key $K_{\mathrm{C_2}}^{-1}$ from the customer, the merchant obtains the payment token PT, that is the payment for the product sold. The merchant then sends the product decryption key, $K_{\mathrm{M_1}}^{-1}$, to the customer encrypted with the customer's public key and also the multiplicative inverse of r modulo N_1, namely, r^{-1}. Finally it writes a log record $\langle T_i, \mathit{FINISH} \rangle$. When the customer receives the product decryption key and obtains the product m, the customer writes a log record $\langle T_i, \mathit{FINISH} \rangle$, and the protocol terminates. If instead of the decryption key, the merchant receives an abort message from the customer (in **Message 4**), it terminates the transaction after writing the log record $\langle T_i, \mathit{FINISH} \rangle$.

5. Extension for handling misbehaving parties and communication problems

Certain changes need to be made to the basic protocol to ensure fair exchange if any player misbehaves or if there is a communication failure. Our aim is to handle disputes automatically within the protocol without resorting to a human arbitrator. By "handling disputes automatically within the protocol", we mean that the suffering party does not have to take recourse to a separate legal system to be compensated for.

The extended protocol described below is executed if the timer used by the customer (see **Message 4** in the basic protocol) expires and the basic protocol does not execute to completion or if there is a dispute after the protocol execution. Note that since the merchant sends the decryption key only after it has received payment in a satisfactory matter, it will always be the case that the customer initiates the extended protocol. The extended protocol involves interaction with the trusted third party and is initiated by the customer by sending M, [[CC(PO), C_{prv}], M_{prv}], [CC([mr, $K_M \times K_{M_1}$]), M_{prv}], [CC([r, K_{M_1}]), M_{prv}]—evidences of the merchant misbehaving—and the key [$K_{C_2}^{-1}$] and the payment token [[PT,$K_{C_1} \times K_{C_2}$], B_{prv}] and B.

M behaves improperly. This includes the following scenarios.

1. Merchant receives **Message 4** but does not send the correct product decryption key $K_{M_1}^{-1}$ in **Message 5**.
2. Merchant receives **Message 4** but disappears without sending the product decryption key.
3. Merchant claims that it did not send correct decryption key because it has not received payment.

The trusted third party asks the merchant to send the product decryption key and starts a timer. If the merchant does not respond within the timeout period, the trusted third party sends the key K_M^{-1} to the customer and takes appropriate action against the merchant. If the merchant responds within the timeout by sending $K_{M_1}^{-1}$, the third party forwards it to the customer. The merchant can also respond by saying that the reason it did not send the product decryption key in the first instance, is because it did not receive proper payment, that is $K_{C_2}^{-1}$. In this case, the merchant still has to provide the trusted third party with $K_{M_1}^{-1}$ and r^{-1}. The trusted third party sends $K_{C_2}^{-1}$ to the merchant and $K_{M_1}^{-1}$ and r^{-1} to the customer.

C behaves improperly. In such an event, the third party looks at all the messages and finds out that the customer has sent an incorrect key. In such a case, the third party does not forward the product key to the customer.

Note that the following two disputes are not entertained.

1. Merchant claims that it has received inadequate payment. The reason why it is not entertained is that the merchant always sends the product decryption key after it has an opportunity to ensure that proper payment has been received.
2. Customer claims after decrypting the product that the correct product was not provided by the merchant. The reason why this is not entertained is that the validated receipt property allows the product to be validated. In other words, if the product does not validate, the customer always has the option of aborting the transaction without paying.

6. Analysis of the properties of the optimistic protocol

Assuming that neither the customer nor the merchant behaves unfairly in the protocol, we show that the protocol satisfies the properties of money atomicity, fair exchange and validated receipt.

Theorem 4. *The optimistic protocol satisfies the money atomicity property.*

Proof of Theorem 4. Assume to the contrary.

Money can be created in the system in two different ways.

1. The customer uses the same payment token to purchase multiple products.
2. The merchant uses the same token to get its account credit multiple number of times.

Both of these are examples of replaying the payment token. The nonce value within the payment

token prevents such replays. Thus money cannot be created in the system.

Money can be destroyed in two different manners.

1. The merchant does not use the payment token to get its account credited.
2. The token is lost after the merchant receives it but before the merchant could get its account credited.

The merchant is assumed to behave rationally and thus it will always use the payment token properly. To protect against the second scenario the merchant can always keep a second copy of the token or approach the third party in good faith. Thus, money is neither created nor destroyed in the system and hence the protocol ensures money atomicity. □

Theorem 5. *The optimistic protocol ensures fair exchange.*

Proof of Theorem 5. Note that fair-exchange will be violated if one of two things happen at the end of the protocol.

Case 1. The customer receives the product, m, but the merchant does not receive correct payment.

Case 2. The merchant receives correct payment but the customer receives the wrong product.

We show that none of these two cases can happen at the end of the protocol. For Case 1 to happen, the customer must receive the product before it pays. Observe that the product is delivered to the customer in an encrypted form in **Message 3** (Fig. 1). To receive the product the customer must have the corresponding product decryption key. The customer does not receive the product decryption key (in **Message 5**) until after the customer has effectively made a payment for the product (sending the payment token decryption key, $K_{C_2}^{-1}$, in **Message 4**). If the payment token decryption key is a correct decryption key, then Case 1 does not arise. If on the other hand the payment token decryption key is not a correct decryption key, then the merchant does not send the product decryption key in **Message 5**. Thus, either way, Case 1 cannot arise at the end of the protocol. For Case 2 to happen, the merchant either does not send the product decryption key or sends the wrong decryption key in **Message 5** of the basic protocol (Section 4.3). Under both circumstances, the extended protocol is executed (Section 5) and the protocol terminates with the customer receiving the product it has paid for.

Hence, fair-exchange is ensured in the protocol. □

Theorem 6. *The optimistic protocol satisfies the validated receipt property.*

Proof of Theorem 6. The customer receives a copy of the encrypted product before sending out the payment for the product (in the form of the decryption key for the payment token PT). The customer is able to compare this copy of the encrypted product with the one he has downloaded earlier from the trusted third party's catalog and make sure that both are encryptions of the same product. In other words, he has received the encrypted version of the product that he is trying to purchase. Theorem 3 ensures this. Thus, the protocol satisfies the validated receipt property. □

7. Increasing the robustness of the protocol

We have seen that the optimistic protocol relies on the trusted third party to resolve disputes and ensure fair exchange in the event of the merchant playing unfairly. Although we do not require the third party to be on-line for the entire duration of the transaction—the third party is invoked only when there is a problem, not otherwise—it is still a source of bottleneck and susceptible to denial of service attacks. For example, the customer tries to initiate the extended protocol but fails to contact the third party. Or, suppose the customer has initiated the extended protocol for dispute resolution. However, the third party is not being able to contact the merchant because the connection to the merchant has failed. In this case, the third party's timer will expire and the third party will take the extreme measure of providing the customer with K_M^{-1}. Under this scenario, the merchant is not offered an opportunity to state its case and suffers due to no fault of its own. We can mitigate this to some extent by not requiring the third party to disclose the key K_M^{-1} to the customer. Instead the

third party decrypts $[m,K_M]$ with K_M^{-1}, and provides m to the customer. Later the third party may take up the issue with the merchant. However, if the link is down for a very long time, then the third party may interpret this as the merchant having disappeared.

Having a single third party makes the protocol more vulnerable to hacking activities at the third party. Recall that for each merchant that chooses to avail of the services of the third party, the latter escrows a key for the merchant—the key K_M^{-1}. The merchant encrypts every copy of every product that it sells, with the key $K_M \times K_{M_i}$, such that the key K_{M_i} is compatible with K_M. If the trusted third party is compromised, then the merchant suffers financially because the corresponding key K_M^{-1}'s is compromised.

To minimize these problems, we distribute the services of the third party across several third parties. This impacts two aspects of the protocol—(i) distributing the key K_M^{-1} and (ii) executing the extended the protocol. We discuss these next.

7.1. Distributing the key K_M^{-1}

If we replicate the key K_M^{-1} over several servers, then we expose the key to more attacks. Thus, the key becomes more vulnerable than it would otherwise be if there was just one server storing the key. However, if we can partition the key into n parts $K_{M_1}^{-1}$, $K_{M_2}^{-1}$,..., etc., and provide each part to a separate third party such that some or all of these third parties can later get together to reconstruct the key, then such a distribution is more secure.

Definition 19. Given a key, K, a protocol to partition the key into n disjoint portions and distribute these among n different entities, is *p-secure* if less than p parties, $p \le n$, cannot regenerate the key K.

If a key K is partitioned in a p-secure manner, then it is guaranteed that the compromise of any number $q < p$ of parties cannot jeopardize the confidentiality of the key. However, making $p = n$ has some disadvantages; in this case, if any one of the parties holding a share of the key fails, then the key cannot be reconstructed. This leads to the definition of *p-availability.*

Definition 20. Given a key, K, a protocol to partition the key into n disjoint portions and distribute these among n different entities, is p-available if greater than or equal to p, $p \le n$, parties can regenerate the key.

Shamir [23] proposed a scheme based on polynomial interpolation by which a secret can be partitioned into n parts such that a subgroup consisting of m or more portions ($m \le n$) can be used to reconstruct the secret. We employ a similar scheme to distribute the key K_M^{-1} across several third parties.[2]

We assume that there are n third parties T_1, T_2, ..., T_n. The merchant knows the identity of each of these third parties and registers with any one of the third parties. We also assume that some p or more number of these third parties need to get together to regenerate the key.

1. M registers with third party T_1. T_1 generates the merchant key-pair (K_M, K_M^{-1}) and provides K_M to the merchant.
2. The third party T_1 randomly generates a $(p-1)$ degree polynomial.

$$y = a_{p-1}x^{p-1} + a_{p-2}x^{p-2} + \ldots + a_1 x + K_M^{-1}$$

   The coefficients a_{p-1}, a_{p-2}, ..., a_1 are chosen randomly over the finite field GF(N).
3. Next T_1 chooses n random points (x_1,y_1), (x_2,y_2), ..., (x_n,y_n) on the polynomial. Each point (x_i,y_i) corresponds to one of the shares for the key K_M^{-1}.
4. T_1 destroys the key K_M as well as the polynomial $y = a_{p-1}x^{p-1} + a_{p-2}x^{p-2} + \ldots + a_1 x + K_M^{-1}$. T_1 stores in its database a record of the form $<M, (x_1,y_1)>$. The value (x_1,y_1) constitutes T_1's portion of the key for M.
5. T_1 generates $(n-1)$ tuples of the form $\langle M,(x_i,y_i)\rangle$, $1 < i \le n$, with each of the remaining values (x_i,y_i) and distributes them securely to each of the other third parties.

[2] Secret sharing schemes have been variously used by researchers. Of particular interest is the COCA project at Cornell University [27] that uses secret sharing techniques for fault-tolerance purposes, much in the same manner as ours. Also of interest is the PASIS project [14] at Carnegie Melon University that uses similar techniques to design survivable storage systems. The authors would like to thank the anonymous reviewers for drawing our attention to these two projects.

Theorem 7. *The key partitioning scheme is p-secure and p-available.*

Proof of Theorem 7. To prove that the protocol is p-secure, we need to show that less than p portions do not disclose any information about the key, K_M^{-1}.

Note that each portion (x_i, y_i) of the key, K_M^{-1} assigned to a third party is a point on the $(p-1)$ degree polynomial $y = a_{p-1}x^{p-1} + a_{p-2}x^{p-2} + \ldots + a_1 x + K_M^{-1}$. This curve intersects the y-axis at exactly one single point, which is the key. If we have less than p portions of key, then there can be many $(p-1)$ degree polynomials that pass through those points and that intersect with the y-axis—all of which can possibly be the key. Or in other words, given less than p portions of the key, we cannot uniquely identify the key. Hence, the partitioning scheme is p-secure.

To prove that the protocol is p-available, we need to show that greater than or equal to p portions can regenerate the key. This is true because there can at most be one $(p-1)$ degree polynomial that passes through p or more given points. Lagrange's interpolation formula can be used to uniquely determine this polynomial. The intersection of the $(p-1)$ degree polynomial, with the y-axis is the key. □

7.2. Executing the extended protocol

As before the extended protocol is executed if the timer used by the customer (see **Message 4** in the basic protocol) expires and the basic protocol does not execute to completion, or if there is a dispute in the protocol execution. The customer initiates the extended protocol by contacting any one of the third parties that participates in the protocol. The customer sends that third party the message containing M, {M, [[CC(PO), C_{prv}], M_{prv}]}, {[CC([mr, $K_M \times K_{M_1}$]), M_{prv}]}, [CC([r, K_{M_1}]), M_{prv}]—evidences of the merchant misbehaving—and the key $[K_{C_2}^{-1}]$, the identity of the financial institution B, and the payment token [PT, $K_{C_1} \times K_{C_2}$].

The trusted third party tries to resolve the dispute as earlier. However, if everything else fails and the only recourse the third party has is to provide the customer with the key K_M^{-1}, then the third party gets in touch with $p-1$ of the other third parties to gather p portions of the key K_M^{-1}. It then regenerates the key K_M^{-1} as described earlier and provides that to the customer.

8. Providing anonymity for the customer

For reasons stated in Section 1, a customer may want to have its privacy protected. A merchant, on the other hand, will rarely want to hide its true identity. It seldom makes good business sense to do so. Consequently, we are more interested in ensuring the privacy of the customer. We first formally define what it means for a protocol to provide customer anonymity.

Definition 21. An e-commerce protocol provides customer-anonymity if no participant can link an executed transaction to a customer's true identity.

For our protocol, the customer's identity is available to the merchant in two different manners. The first is via the signature on the purchase order and the second is indirectly via the payment token. The only purpose the signature on the purchase order serves is to provide a non-repudiable proof of the customer's intention to purchase a product at an agreed upon price. The identity of the customer is important only to the extent that the identity can be linked to the signature. No other information about the customer is available from the signature.

More information about the customer is however available from the payment token. The payment token contains detailed information about the customer's identity at the customer's financial institution, the customer's account information and the amount of the transaction. Once the merchant gets the payment token, the merchant has detailed information about the customer (particularly so if it colludes with the customer's financial institution) with which it can create a customer profile. The customer may not be willing to divulge or share this information. In short, the customer may want some degree of anonymity.

The approach that we choose to provide anonymity to the customer is to use the digital equivalent of cash in the transaction. Okamoto and Ohta [20] identified six properties that any electronic cash system must have.

1. The cash should be sent securely through computer networks.
2. The cash cannot be copied and reused.
3. The spender of the cash can remain anonymous if needed.
4. The transaction can be done off-line.
5. The cash can be transferred to others.
6. A piece of cash can be divided into smaller amounts.

The commercially available protocol known as "eCash™" provides such functionality. It is based on "Digicash" [8]. However, the problem with Digicash, and consequently with eCash, is that it does not provide either money atomicity or goods atomicity (fair exchange) [15]. Nor does it provide the validated receipt property. In this section, we show how we can modify our basic protocol to accommodate an electronic cash facility similar to eCash, yet preserving the money atomicity, fair-exchange and validated receipt properties. At the same time, the merchant does not obtain enough information about the customer to create a customer profile. Neither does the customer's financial institution have enough information on its own to link a particular transaction to the customer.

To keep the protocol simple, we relax some of the properties of electronic cash systems identified by Okamoto and Ohta [20]. In particular, we disallow the transfer of electronic cash to others (relaxing item 5 above); we do not require that a piece of electronic cash be divisible into smaller amounts (relaxing item 6); we also assume that all the coins issued by the bank have the same denomination; finally we require that the transaction with the bank be done in an on-line manner.

8.1. The optimistic, anonymous protocol

Briefly, the anonymous protocol executes as follows.

1. Customer decides to buy product and downloads an encrypted version from the trusted third party (as in the basic protocol).
2. Customer buys digital coins of appropriate value from the customer's bank.
   (a) Customer sends unsigned blinded coins with unique serial numbers to the bank.
   (b) The bank debits the customer's account by the relevant amount.
   (c) The bank signs the blinded coin and sends them back to the customer.
   (d) Customer un-blinds the coin to get a signed digital coin having the necessary value.
3. Customer sends signed digital coins, properly encrypted, to the merchant together with the purchase order.
4. Merchant sends the encrypted product.
5. Customer validates encrypted product and sends the decryption key for the digital coin.
6. Merchant verifies with customer's bank that the coins are still valid, that is they are not already spent.
7. The bank notifies merchant that coins are still valid, credits the merchant's account and records in a database the serial number of the coin spent.
8. Merchant sends product decryption key and completes transaction.

The anonymous optimistic protocol is shown in Fig. 2.

The generation and use of electronic cash in our protocol requires the use of blind signatures. The idea of blind signatures is as follows. Suppose that a message m has to be signed by an entity, but without the signer knowing the contents of the message. Then the generator of m should be able to

1. randomize m so that the signer cannot determine m,
2. obtain the signature on the randomized message,
3. undo the randomization to obtain the signature on the original message.

Chaum [6,7] has shown how to implement such blind signatures based on integer factorization. The scheme, briefly, is as follows: Suppose a customer, C, has a message m that she wishes to have signed by her financial institution, B. C does not want B to learn anything about m. Let (n,e) be B's public key and (n,d) be its private key. C generates a random value r such that $\gcd(r,n)=1$ and sends $x=(r^e m) \bmod n$ to B. The value x is "blinded" by the random value r (the "blinding factor"); hence, B cannot derive any useful information from it. B returns the signed value $t=x^d \bmod n$ to C. Since $x^d \equiv (r^e m)^d \equiv r m^d \bmod n$, C can obtain the true signature s of m by computing $s=r^{-1}t \bmod n$.

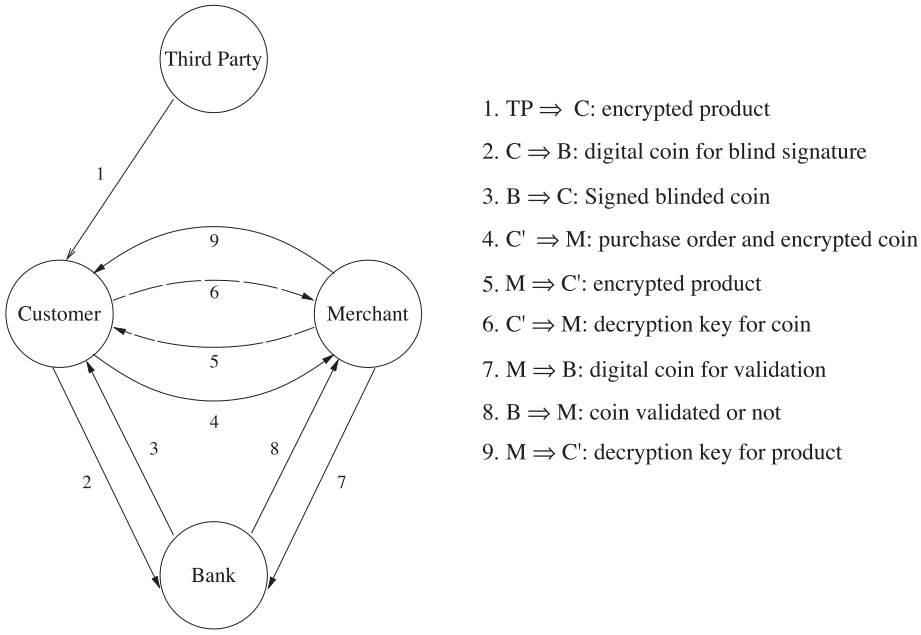

Fig. 2. Message exchanged in the integrated protocol.

We now provide the complete details of the anonymous optimistic protocol. As earlier, we assume that the customer opens an account with a bank, B. The bank knows the true identity of the customer, C, and associates an account number AC_C with this customer. We assume that the merchant, M, has an account, AC_M, at the same bank. Note that this assumption does not affect the properties of the protocol in any manner. If the merchant's bank and the customer's bank are the same entity then any information that is available at one bank is available immediately at the "other bank". Co-locating the banks makes collusion easier. If these two banks are different, then we need to have additional messages exchanged between the two to arrive at the same state. On the other hand, if the banks are different, then providing anonymity is easier[3] than when the banks are the same. Finally, we assume that we have a group of trusted third parties working as outlined in Section 7; however, to keep the discussion simple, we abstract the group of third parties by a single third party.

[3] We can, in theory at least, come up with a way to stop message flow between the two banks thereby preventing information exchange.

For each new transaction with a merchant, the customer uses a different pseudo identifier, C′. The real identifier, C, is known only to the customer's financial institution and is not divulged by the customer in the protocol. The protocol ensures that except the customer himself, no other player in the protocol—namely, the bank, the trusted third party, or the merchant—has enough information, either by itself or through collusion with each other, to link the pseudo identifier, C′, used by the customer in the transaction with the customer's real identity, C. Thus, the anonymity of the customer is preserved. If the customer chooses not to remain anonymous, then the customer simply uses his true identity.

The customer, as before, selects the product to purchase, m, from the catalog at one of the trusted third parties. The protocol starts when the customer tries to acquire digital coin of some value from the bank. The following messages describe the anonymous optimistic protocol.

Message 1

$$\mathrm{TP} \Rightarrow \mathrm{C}: \ [m, K_M].$$

The customer downloads $[m, K_M]$ from the trusted third party server together with the product identifier,

PID. Note that the customer does not actually have the product m, because he does not have the decrypting key K_M^{-1}. This $[m, K_M]$ will be used later by the customer to validate the product received from the merchant.

Message 2

$$C \Rightarrow B: \ [C, AC_C, \psi_C, B_{pub}], nr, [CC(nr), C_{prv}]$$

The customer C requests the bank for digital coins. To do this, C sends an unsigned blinded coin to the bank together with information about his identity and account number and a nonce ψ_C. The nonce is used to prevent replay of this message at the bank.

C also sends a signed cryptographic checksum of the blinded coin. This ensures that the blinded coin gets to the bank properly. The customer uses his true identity to sign the checksum of the blinded coin. This is needed to ensure that the proper accounts get debited.

There is no need to encrypt the blinded coin with the bank's public key. This is because the blinding factor, r, is available only to C, so C is the only one that can get n.

Message 3

$$B \Rightarrow C: \ [nr, B_{prv}]$$

The bank receives the blinded coin and debits the customer's account for an amount equal to the value of the coin. It then issues the digital coin by signing the blinded serial number, nr, and sends the resulting message to the customer.

Note that at this stage the bank knows about the identity of the customer and the value of some digital coin that has been issued to the customer. However, since the coin serial number is blinded, the bank has no way of linking a particular digital coin to this customer, once the blinding factor is removed. The customer receives the blinded coin and the blinded checksum of the coin and removes the blinding factor, r, to get a coin signed by the bank—($[n,B_{prv}]$).

Message 4

$$C' \Rightarrow M: \ [PO, M_{pub}], [CC(PO), C'_{iprv}], [[n, B_{prv}], S]$$

The customer initiates the e-commerce transaction by associating a unique system-wide identifier T_i for the transaction. For this transaction, the customer assumes a pseudo identifier, C′ and a one time private-key/public-key pair—$\{C'_{iprv}, C'_{ipub}\}$. Also, the customer chooses a secret key, S, to encrypt the signed coin. The transaction identifier is a tuple of the form ⟨PID, C′, M⟩. The customer stores a log record of the form $\langle T_i, INITIATE\rangle$ to its stable storage and then sends to the merchant, the purchase order, PO, and the signed digital coin obtained from the bank.

The purchase order, PO, contains the following information:

(i) the product identifier, PID;
(ii) the customer's pseudo identifier, C′;
(iii) the price of the product, PR; and
(iv) a nonce for the customer ψ_C'; this nonce can contain information about the customer's pseudo identity but not the real identity.

The purchase order is encrypted with the merchant's public key. The customer generates a cryptographic checksum of PO and digitally signs it using his one time private key C'_{iprv}. The cryptographic checksum forestalls debate over the details of the order, or whether the order was received completely and correctly. The customer's signature on the PO, although cannot be authenticated by the merchant at this time, can be used later to forestall debate over whether a customer with pseudo identity C′ expressed an intention to purchase the product. The nonce, ψ_C', in the purchase order forestalls a replay of the purchase order with the merchant.

The customer sends the digital coin encrypted with the secret key, S.

Note that, we assume that all coins have the same denominations. To purchase a product, a customer may have to send multiple coins to the merchant. In such scenarios, **Messages 2**, **3** and **4** will be repeated a number of times.

Message 5

$$M \Rightarrow C': \ [[CC(PO), C'_{iprv}], M_{prv}], [mr'', K_M \times K_{M_1}],$$
$$[r'', K_M], [CC([r'', K_M]), M_{prv}],$$
$$[CC([mr'', K_M \times K_{M_1}]), M_{prv}]$$

OR

$$M \Rightarrow C': \ \text{Finish}$$

When the merchant receives **Message 4** from the customer, it writes a log record $\langle T_i, INITIATE\rangle$ to its stable storage; then the merchant checks to see if the purchase order is to its satisfaction—that is, the merchant agrees to all its contents. If not, the merchant writes a finish record in its stable storage—$\langle T_i, FINISH\rangle$—and terminates the transaction. It informs the customer of this decision. Otherwise, the merchant endorses the purchase order by signing [CC(PO), C'_{iprv}]. The merchant sends this endorsed purchase order together with the encrypted product [mr'', $K_M \times K_{M_1}$], its signed cryptographic checksum [CC([mr'', $K_M \times K_{M_1}$]), M_{prv}], the random number, r'', encrypted with K_M and its cryptographic checksum. The merchant's endorsement on the purchase order forestalls debate over whether the purchase order was received correctly or not and whether the merchant agreed to the terms of the current transaction. The signed cryptographic checksum [CC([mr'', $K_M \times K_{M_1}$]), M_{prv}] proves the origin of the encrypted product [mr'', $K_M \times K_{M_1}$] as well as ensures the integrity of [mr'', $K_M \times K_{M_1}$] in transit.

Message 6

$$C' \Rightarrow M: [S, M_{pub}], [CC([m, K_M \times K_{M_1}]), C'_{iprv}]$$

OR

$$C' \Rightarrow M: \text{ Abort}, [CC([m, K_M \times K_{M_1}]), C'_{iprv}]$$

After receiving **Message 5** from the merchant, the customer checks to see if it is an abort message or the encrypted product. If it is an abort, the customer terminates the transaction and writes a log record of the form $\langle T_i, FINISH\rangle$. Now if the customer still wants to purchase the product, the customer initiates a new transaction with a new purchase order. If the message from the merchant is not an abort, the customer knows that the merchant is willing to proceed with the transaction. The customer then validates the product as outlined in Section 3.2. If the two compare, the customer sends the coin decryption key, S to the merchant, encrypted by the merchant's public key and a signed cryptographic checksum of the encrypted product received. The customer then writes a log record to its stable storage. The log record is of the form $\langle T_i, PAYMENT\text{-}SENT\rangle$. Finally the customer starts a timer, waiting for the product decryption key to arrive from the merchant. If the timer expires before the product decryption key arrives from the merchant, the customer executes the extended protocol. If, on the other hand, the product is not validated the customer can request the product from the merchant once more, or abort the transaction. For the latter the customer enters $\langle T_i, ABORT\rangle$ in its log and sends an abort message to the merchant together with a signed cryptographic checksum of the product received from the merchant. The signed cryptographic checksum of the encrypted product forestalls debate over whether the encrypted product received by the customer is the same as the encrypted product sent by the merchant.

Message 7

$$M \Rightarrow B: [\{[[n, B_{prv}], M_{prv}], M\}, B_{pub}]$$

OR

$$M \Rightarrow C': \text{ Abort}$$

After the merchant receives the private key from the customer to open the digital coin, the merchant verifies the signature of the bank on the coin. Depending on the price of the product, the merchant will receive multiple such messages. If the merchant is not satisfied for any reason at this stage, it sends an abort message to the customer and writes an abort record, $\langle T_i, ABORT\rangle$, in its log. Note by keeping an abort record in its log, the merchant indicates that something went wrong with the transaction and that it expects to hear either from the customer or from the third party. The customer, on receiving the abort message, can choose to abort the protocol and return the coin to the bank or can re-send the key. If the customer feels that the merchant is playing unfairly, it can initiate the extended protocol to resolve disputes.

If, on the other hand, the merchant is satisfied, it digitally signs the coin and sends the coin to the bank together with the merchant's identity. The message is encrypted with the bank's public key. At this time the merchant writes a log record, $\langle T_i, PAYMENT\text{-}EXPECTING\rangle$. It then starts a timer waiting for a response to come back. If the timer expires before a response, the merchant resends the coin.

If instead of the private key, the merchant receives an abort message from the customer, it terminates the transaction after writing the log record $\langle T_i, FINISH\rangle$. Writing a finish record indicates that there will not be a dispute resolution phase for this transaction.

Since all the coins are of the same denominations, the merchant may need to send **Message 7** a number of times depending on the value of the product.

Message 8

$B \Rightarrow M$: yes

OR

$B \Rightarrow M$: no

The bank maintains a list of coins spent. Each record in the table is of the form $\langle$*COIN-SERIAL-NO*, *PAID-TO*$\rangle$. When the bank receives the coin from the merchant, it checks to see if the coin has been spent before. If so the bank replies by a "no" message. Otherwise it credits the merchants account for an amount equal to the value of the coin and replies "yes".

Message 9

$M \Rightarrow C'$: $[K_{M_2}^{-1}, C'_{ipub}], [r''^{-1}, C'_{ipub}]$

OR

$M \Rightarrow C'$: Abort

Once the merchant receives a "yes" message from the bank, the merchant sends the product decryption key, $K_{M_2}^{-1}$, to the customer encrypted with the customer's public key and also the multiplicative inverse of r'' modulo N_2, namely, $r^{-1}{}''$. Then it writes a log record $\langle T_i, FINISH\rangle$ and forgets about the transaction. When the customer receives the product decryption key and obtains the product m, the customer writes a log record $\langle T_i, FINISH\rangle$, and the protocol terminates.

If on the other hand the merchant receives a "no" message from the bank, it writes an abort log record $\langle T_i, ABORT\rangle$ in its log and sends an abort message to the customer.

If the customer receives the product decryption key $K_{M_2}^{-1}$ and the multiplicative inverse of r'' modulo N_2, it writes a finish record in its log $\langle T_i, FINISH\rangle$ and terminates the transaction. Else if the customer feels that the merchant has played unfairly, it may choose to execute the extended protocol with the third party for dispute resolution.

The extended protocol that is used for dispute resolution is slightly modified from the one outlined in Section 5. The extended protocol is initiated by the customer if the customer's timer expires after **Message 9** or the customer receives an abort message from the merchant in reply to **Message 9**. The customer provides the same set of evidence to the trusted third party as in the basic protocol in order to initiate a dispute resolution. The third party gets in touch with the bank to validate the coins. If the coins have not been spent, the TP knows that the customer is behaving fairly. Otherwise, the TP does not know who is behaving unfairly. In such circumstances, it requests more information from the bank about who has deposited the coin, requests evidence from the party who has deposited the coin, and takes the appropriate action.

There can be a scenario in which the customer does not care to initiate a dispute resolution phase (because, for example, it realized that it had knowingly or unknowingly played unfairly). However, the merchant has not yet terminated the transaction as it expects a round of dispute resolution. Under this scenario, the merchant will contact the trusted third party after a suitable period of time and inquire about a "certain transaction T_i" with a "certain customer" having the pseudo-id C′. If the third party is not aware of any such transaction, the merchant will terminate the transaction.

9. Analysis of the anonymous, optimistic protocol

In this section, we show that the anonymous, optimistic protocol satisfies the properties of *money-atomicity*, *fair-exchange*, *validated-receipt* and *customer-anonymity.*

Theorem 8. *The anonymous, optimistic protocol satisfies the money atomicity property.*

Proof of theorem 8. The proof is similar to Eq. (4). We assume the contrary. Money can potentially be created if any player attempts to double-spend the digital. The customer can try to double spend the coin by using it for another purchase, either at the same merchant or at a different merchant. However, this will soon be caught because the financial institution keeps the serial number of all coins that has been presented to it for crediting and will not validate a coin (step 7/8 of protocol) if it is presented for validation more than once. The merchant may want to double spend the coin by presenting it more than once

to its financial institution. Here again the financial institution will not validate the coin and prevent double spending.

Money can be destroyed in one of three different ways:

1. The customer destroys the digital coins after obtaining them from the financial institution. Since the customer's account is debited before the coins are issued, this causes destruction of money. (Note that for Theorem 4, the account was not debited when the token was issued, so this case did not arise.)
2. The merchant destroys the digital coins before depositing them to the financial institution.
3. The merchant looses the digital coins before using them.

Cases (1) and (2) reflect irrational behavior on the part of the customer or the merchant hence are disregarded. Case (3) is taken care of in the same manner as in Theorem 4.

Thus, money atomicity is ensured in the protocol. □

Theorem 9. *The anonymous, optimistic protocol ensures fair-exchange.*

Proof of Theorem 9. The proof is the same as in Theorem 5. □

Theorem 10. *The anonymous, optimistic protocol satisfies the validated-receipt property.*

Proof of Theorem 10. The proof is the same as in Eq. (6). □

9.1. Analysis of anonymity

The main objective of the anonymous, optimistic protocol is to prevent disclosure of the true identity of the customer to all possible adversaries while ensuring money atomicity, fair-exchange and validated-receipt. We claim that

Theorem 11. *The protocol provides customer-anonymity.*

Proof of Theorem 11. Recall (from Definition 21) that to provide customer-anonymity, the protocol must ensure that (i) no single party (other than the customer) has enough information to link the pseudo-identifier used by the customer to the customer's real identity and (ii) it will not be possible for all the parties (besides the customer) to collude and get this information. The only way to break the customer's anonymity is to get either the information "C′'s account is AC_C" or the information "C carries on transaction T_i", or the information "C and C′ refer to the same entity".

First we identify the information that each of the protocol participants—the customer (C), the trusted third party (TP), the bank (B) and the merchant (M)—individually knows at the end of the protocol execution. We only consider the information that may potentially be used to link C to C′. This information is tabulated in Table 2. The entry Maybe in a cell is interpreted as Yes if the extended protocol is executed, otherwise it is No. The table is interpreted as follows: Consider first row 1 in the

Table 2
Information possessed by each party

Information	C	TP	B	M
C	Yes	No	Yes	No
B	Yes	Yes	Yes	Yes
M	Yes	Yes	Yes	Yes
TP	Yes	Yes	Yes	Yes
C′	Yes	Maybe	No	Yes
C_{pub}	Yes	No	Yes	No
C_{prv}	Yes	No	No	No
C'_{ipub}	Yes	Maybe	No	Yes
C'_{iprv}	Yes	No	No	No
K_M	No	Yes	No	Yes
K_M^{-1}	No	Yes	No	Yes
K_{M_1}	No	No	No	Yes
$K_{M_1}^{-1}$	Yes	No	No	Yes
PO	Yes	Maybe	No	Yes
C's account is AC_C	Yes	No	Yes	No
C″s account is AC_C	Yes	No	No	No
n	Yes	Maybe	Yes	Yes
r	Yes	Maybe	No	No
r''	Yes	Maybe	Maybe	Yes
$[m, K_M \times K_{M1}]$	Yes	Maybe	Maybe	Yes
C and C′ refer to the same entity	Yes	No	No	No
C carries on transaction T_i	Yes	No	No	No
C′ carries on transaction T_i	Yes	Maybe	No	Yes
C_{pub} and C'_{ipub} belongs to the same entity	Yes	No	No	No

table. No under columns TP and M indicate that the true identity of the customer, that is C, is not known to TP or M. Yes under columns C and B indicate that the true identity of C is known to C (of course) and B. Consider next row 5. A Maybe under column TP indicates that in general TP does not know the identity of C′. Only if the dispute resolution part of the protocol is executed will TP be aware of certain C′. □

From Table 2, we note that the only participant who has the information "C and C′ refer to the same entity" is C itself. So unless C volunteers this information to others, nobody has access to that information. We find that at the end of the protocol, no single participant has enough information to link the pseudo-identifier C′ to the real customer C. Let us see if, by collusion between two or more entities (other than the customer), they can acquire enough information to link C′ with C.

The necessary condition for two or more parties to collude are (a) the parties must know each other's identity and (b) the parties must have some common piece of information pertaining to the transaction that the customer carries on with the merchant; this is the first step before the colluding parties can continue with the process.

We initially assumed that the third party will not misbehave. So under that assumption, TP will not be colluding with any other player. However, for the sake of analysis, let us assume that TP also colludes. We first determine if two given parties are in a position to collude.

TP and M — TP knows about M's identity and vice versa. This is because M has to register at TP. Also they have a common piece of information $[m, K_M \times K_{M_1}]$ pertaining to the transaction that can enable collusion. TP is the only entity to profit by this collusion. It gains knowledge about the following pieces of information—C′, C'_{ipub}, K_{M1}, $K_{M_1}^{-1}$, PO, n, r'' and C′ carries on transaction T_i. M does not learn anything new about the true identity of C.

TP and B — TP and B know each other's identity. However, they do not have any common piece of information about the transaction. Thus, they are not in a position to collude.

M and B — M and B know each other's identity and they have the following piece of information about the transaction—n, the serial number for the coin. Both B and M benefit from this collusion. B gets to have the following additional information—C′, K_{M_1}, $K_{M_1}^{-1}$, r'', and "C′ carries on transaction T_i". M, on the other hand, acquires the following additional knowledge—C, C_{pub}, AC_C, and "C's account is AC_C". However, M cannot link C′ to C.

Thus, it is clear that via two party collusion, no entity acquires sufficient information to link C to C′.
Let us now consider the only possible three party collusion.

TP, B and M — TP, B and M have each other's identity and via two party collusion they have a number of common pieces of information about the transaction—$[m, K_M \times K_{M_1}]$ and n to name two. By such a collusion, TP, B and M all come to share the following pieces information—C, C_{pub}, C'_{ipub}, M_{prv}, B_{prv}, K_M, K_M^{-1}, K_{M_1}, $K_{M_1}^{-1}$, PO, AC_C, AC_M, n, r'', $[m, K_M \times K_{M_1}]$, "C′ carries on transaction T_i" and "C's account is AC_C". It is clear that even by three party collusion, no entity is able to acquire either the information "C carries on transaction T_i" or the information "C′'s account is AC_C". Hence, the protocol provides customer-anonymity.

10. Conclusion and future work

Fair exchange is an important property that needs to be addressed by all electronic commerce protocols. Majority of protocols proposed in the literature rely on collecting evidence during protocol execution that can be used later in a court of law to settle disputes. Unfortunately, such after-the-fact dispute resolution

may often be unacceptable, for example, in situations where it is not possible to bring an offending party to a suitable court of law. Keeping this in mind, we proposed an e-commerce protocol that ensures strong fairness within the scope of the protocol itself. Our protocol falls into the category that uses a trusted third party for assuring fair-exchange.

Most of the times, e-commerce transactions proceed without any party misbehaving. Keeping this in mind, we take an optimistic approach in the protocol—do not contact the trusted third party unless necessary. This is in contrast to some other similar protocols that actively use the third party to ensure fair exchange. Our approach reduces the bottleneck at the trusted third party and also allows us to achieve our objectives using an off-line trusted third party that need not be active for the entire duration of the transaction. We further reduce the bottleneck at the third party by distributing the services of the third party over several servers. Distributing the third party provides an additional advantage. The protocol is now more resilient towards denial of service attacks and hacking activities aimed at the trusted third party.

We extend the protocol to allow anonymous transactions by the customer. This is achieved by allowing the customer to use digital money similar to the one in the Digicash protocol. We ensure that a particular transaction can, in no way, be linked to a particular customer, even though the customer's financial institution that issues the digital money knows the customer's real identity. Such functionality is advantageous for the customer's privacy in many ways.

In summary, our protocol has the following features. First, it provides strong fairness under all circumstances. The customer does not get the product unless he pays for it and the merchant does not get paid unless he delivers the product. Note that fairness is always ensured and is not compromised even if any party misbehaves or prematurely aborts. 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 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 can be generalized and used for the fair exchange of any two digital items, not necessarily electronic goods and electronic payment. Sixth, the protocol is not susceptible to denial-of-service attacks aimed at the trusted third party. Last but not the least, the protocol allows the customer to remain anonymous in the transaction.

To our knowledge, no other e-commerce protocol offers all these features. However, the protocol at this time is in its early stages of development and a lot of work remains to be done. First and foremost, we need to formally analyze the protocol. Formal analysis will help us answer two questions: (i) Is the protocol really secure and safe to use? If so, under what assumptions? If not, why not and can it be improved? (ii) Does the protocol behave correctly when it is operating in conjunction with other protocols and under different operating condition? We plan to use formal software specification and verification tools like FDR [12] for this purpose.

Second, the use of trusted third parties can sometimes be regarded as a limitation of the protocol. Identifying a third party that can be completely trusted is no trivial proposition. We are currently investigating ways by which we can use semi-trusted third parties. Semi-trusted third parties can misbehave but they are assumed not to collude actively with any player in the protocol to the detriment of the other players.

Third, the performance of this protocol has to be formally evaluated. In particular, we plan to study the load at the trusted third party and how the frequency of failure of the third party affects the performance. Such a study will help us identify ways to optimize the protocol.

Last but not the least, we plan to implement the protocol. We plan to use COTS components for this implementation. Implementation will give us a different perspective on the protocol and may require addressing new issues.

Acknowledgements

The authors would like to thank the anonymous reviewers for their useful comments and suggestions. This work was partially supported by the U.S. National Science Foundation under grants EIA 9977548 and IIS 0242258.

References

[1] N. Asokan, M. Schunter, M. Waidner, Optimistic protocols for fair exchange, Proceedings of the 4th ACM Conference on Computer and Communications Security, Zurich, Switzerland, Association for Computing Machinery, New York, 1997 (April), pp. 7–17.

[2] N. Asokan, V. Shoup, M. Waidner, Optimistic fair exchange of digital signatures, in: K. Nybergi (Ed.), Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Eurocrypt '98, Helsinki, Finland, Lecture notes in Computer Science, vol. 1403, Springer-Verlag, Berlin, 1998 (June), pp. 591–606.

[3] F. Bao, R.H. Deng, W. Mao, Efficient and practical fair exchange protocols with off-line TTP, Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, IEEE Computer Society, California, 1998 (May), pp. 77–85.

[4] M. Ben-Or, O. Goldrich, S. Micali, R. Rivest, A fair protocol for signing contracts, IEEE Transactions on Information Theory 36 (1) (1990) 40–46.

[5] M. Blum, How to exchange (secret) keys, ACM Transactions on Computer Systems 1 (1983) 175–193.

[6] D. Chaum, Blind signatures for untraceable payments, in Advances in Cryptology-Proceedings of CRYPTO '82, Plenum Press, New York, 1982, pp. 199–203.

[7] D. Chaum, Security without identification: transaction systems to make big brother obsolete, Communications of the ACM 28 (10) 1985 (October) 1030–1044.

[8] D. Chaum, A. Fiat, M. Naor, Untraceable electronic cash, in Advances in Cryptology—Proceedings of CRYPTO '88, Santa Barbara, CA, Springer-Verlag, Berlin, 1990, pp. 200–212.

[9] B. Cox, J.D. Tygar, M. Sirbu, NetBill security and transaction protocol, in Proceedings of the 1st USENIX Workshop in Electronic Commerce, New York, NY, USENIX Association, California, 1995 (July), pp. 77–88.

[10] R.H. Deng, L. Gong, A.A. Lazar, W. Wang, Practical protocols for certified electronic mail, Journal of Network and Systems Management 4 (3) (1996).

[11] S. Even, O. Goldreich, A. Lempel, A randomized protocol for signing contracts, Communications of the ACM 28 (6) 1985 (June) 637–647.

[12] Formal Systems (Europe) Ltd., Failure Divergence Refinement-FDR2 User Manual, version 2.64 edition, 1999 (August).

[13] M.K. Franklin, M.K. Reiter, Fair exchange with a semi-trusted third party, Proceedings of the 4th ACM Conference on Computer and Communications Security, Zurich, Switzerland, Association for Computing Machinery, New York, 1997 (April), pp. 1–6.

[14] G.R. Ganger, et al, Survivable storage systems, Proceedings of the DARPA Information Survivability Conference and Exposition, Anaheim, CA, vol. 2, IEEE Computer Society, California, 2001 (April), pp. 184–195.

[15] N. Heintze, J. Tygar, J. Wing, H. Wong, Model checking electronic commerce protocols, Proceedings of the 2nd USENIX Workshop in Electronic Commerce, Oakland, CA, USENIX Association, California, 1996 (November), pp. 146–164.

[16] B. Kaliski, M. Robshaw, The secure use of RSA, CryptoBytes 1 (3) (1995) 7–13.

[17] S. Ketchpel, Transaction protection for information buyers and sellers, Proceedings of the Dartmouth Institute for Advanced Graduate Studies: Electronic Publishing and the Information Superhighway, Dartmouth College, New Hampshire, 1995.

[18] National Institute of Standards, FIPS 180: Secure Hash Standard, NIST Information Technology Laboratory, USA, 1993 (April).

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

[20] T. Okamoto, K. Ohta, Universal electronic cash, Advances in Cryptology—Proceeding of CRYPTO '91, Santa Barbara, CA, Lecture Notes in Computer Science, vol. 576, Springer-Verlag, Berlin, 1992, pp. 324–337.

[21] I. Ray, 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, Lecture Notes in Computer Science, vol. 1875, Springer-Verlag, Berlin, 2000 (September), pp. 84–93.

[22] T.W. Sandholm, V.R. Lesser, Advantages of a leveled commitment contracting protocol, Proceedings of the 13th National Conference on Artificial Intelligence, Portland, OR, The MIT Press, Massachusetts, 1996, pp. 126–133.

[23] A. Shamir, How to share a secret, Communications of the ACM 22 (1979) 612–613.

[24] W. Stallings, Cryptography and Network Security: Principles and Practice, 2nd edition, Prentice-Hall, New Jersey, 1999.

[25] J.D. Tygar, Atomicity in electronic commerce, Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing, Philadelphia, PA, Association for Computing Machinery, New York, 1996 (May), pp. 8–26.

[26] J. Zhou, D. Gollmann, A fair non-repudiation protocol, Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, IEEE Computer Society, California, 1996 (May), pp. 55–61.

[27] L. Zhou, F.B. Scheider, R. van Renesse, COCA: a secure distributed on-line certification authority, ACM Transactions on Computer Systems 20 (4) 2002 (November) 329–368.

Dr. Indrajit Ray is an Assistant Professor in the Computer Science Department at Colorado State University. He joined the faculty of Computer Science in August 2001. Between fall 1997 and spring 2001 he worked as a tenure track faculty in the Computer and Information Science department at the University of Michigan - Dearborn after obtaining his PhD in Information Technology from George Mason University, Fairfax VA. He teaches courses in computer networks, database systems and computer security. His main research interests are in the areas of computer and network security and computer forensics. Some of his recent works involve developing secure transaction processing systems, designing survivable networks, and designing secure and reliable fair-exchange protocols, anonymous protocols, and voting protocols.

Dr. Indrakshi Ray is an Assistant Professor in the Computer Science department at Colorado State University. She joined the faculty of Computer Science in August 2001. She graduated from George Mason University in summer 1997. Her dissertation focused on how formal methods can be used to solve problems in the area of multilevel secure databases, heterogeneous databases, and databases having long-duration transactions. Before joining Colorado State University, she was an Assistant Professor at the University of Michigan – Dearborn where she taught courses in Database Security, Software Engineering, and Formal Methods in Software Engineering. Her research during this period was on the development and formal verification of secure internet protocols. At Colorado State University she teaches courses on Database Systems and Database Security. Her research spans two areas: Data and Application Security and Software Specification Languages. Currently she is working on how to design secure software systems. She is also looking into some problems associated with information security policies.

Dr. Narasimhamurthi Natarajan is an Associate Professor in the Electrical and Computer Engineering department at the University of Michigan-Dearborn. He had previously held faculty positions in the Electrical and Computer Engineering department at the University of Michigan - Ann Arbor and in the System Sciences and Mathematics department at the Washington University, St. Louis. He obtained his PhD in Electrical Engineering and Computer Science in 1977 from the University of California - Berkley. His research interests are in the areas of control systems, modeling, electric vehicles and software development and algorithms.