An Optimistic Fair Exchange E-commerce Protocol with Automated Dispute Resolution*

Indrakshi Ray and Indrajit Ray

Department of Computer and Information Science
University of Michigan-Dearborn
Email: {iray, indrajit}@umich.edu

Abstract. In this paper we propose an e-commerce protocol with the following features: (1) ensures true fair exchange, (2) does not require manual dispute resolution in case of unfair behavior by any party, (3) does not require the active involvement of a trusted third party, (4) allows the customer to verify that the product he is about to receive is the one he is paying for, and (5) can be used for the fair exchange of any two digital items.

1 Introduction

Researchers have identified a number of desirable properties of e-commerce protocols: (i) should ensure fair exchange, (ii) should not require manual dispute resolution in case of unfair behavior by one party, (iii) each party should have the assurance that the item he is about to receive is the correct one, and (iv) should not require the active involvement of a trusted third party. In this paper we propose an e-commerce protocol that satisfies all these properties.

Fair exchange protocols have been proposed in the context of electronic mails [9, 19] and electronic transactions [8, 14]. Most of these works [8, 9, 19] focus on storing evidence that is to be used in case one party misbehaves. If a dispute occurs, a judge looks at the evidence and delivers his judgment. The dispute resolution is done after the protocol execution, that is, after the customer has obtained his product or the merchant his money. However, such "after-the-fact" protection [14, 15] may be inadequate in an e-commerce environment where the customer and the merchant may simply disappear.

The need for such "after-the-fact" dispute resolution does not arise if protocols provide *true fair exchange* – under all circumstances, either the two parties get each other's items or none do. Protocols providing true fair exchange [14] typically use an online trusted third party. The third party receives the items from each party, verifies the items, and then forwards them to the other party. As a result if any party misbehaves or prematurely quits, no harm is caused to the other party. Moreover, each party has the assurance that the item he is about to receive is indeed the correct one. 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.

* This work has been partially supported by the NSF grant EIA 9977548, and by the University of Michigan-Dearborn Faculty Research Grant and Summer Research Grant.

Several protocols have been proposed [1–3, 5] that do not use the third party unless a problem, such as, a party misbehaving or prematurely aborting, occurs. Such protocols are termed optimistic [1–3]. Some of these protocols provide "after-the-fact" protection [1] and others [5] are restricted to the exchange of digital signatures.

This motivates us to propose an e-commerce protocol satisfying all the above mentioned desirable properties. The protocol is based on the theory of cross validation using which each party can verify that the item he is about to receive is indeed the correct one. The protocol does use a trusted third party to ensure fair exchange. The merchant escrows the encrypted product and a pair of keys with the trusted third party. If the merchant disappears after receiving the payment, the trusted third party can always give the customer the keys for decrypting the product. But the third party is not involved unless a problem, such as, a party misbehaving or prematurely aborting occurs. Thus, the use of the third party is kept to a minimum level.

The rest of the paper is organized as follows. Section 2 describes some related work. Section 3 briefly presents the theory of cross validation and describes how the product validation takes place. Section 4 describes the optimistic protocol. Section 5 discusses how the protocol is extended to handle misbehaving parties. Finally, section 6 concludes the paper.

2 Related Work

Previous work on fair exchange schemes can be classified under two categories: (i) gradual exchange protocols and (ii) third party protocols. Gradual exchange protocols [4, 6, 10] gradually increase the probability of fair exchange over several rounds of message exchanges; these protocols have extensive communication requirements and assume that both the parties have equal computational power.

The third party protocols [8, 9, 11, 19] make use of 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 a mail message was proposed by Deng et al. [9] and Zhou and Gollmann [19]. Dispute resolution is outside the scope of these protocols; however, the protocols do specify what evidence must be stored for the dispute to be resolved in a fair manner.

Using a trusted third party to ensure fair exchange in the context of sale of low-priced network goods has been proposed in the NetBill system [8]. The trusted third party is the NetBill server that maintains financial accounts for the customers and the merchants. The customer, if interested, requests the merchant for a good. The merchant, in response, sends the encrypted good. On receiving the encrypted good, the customer sends the merchant payment information. The merchant forwards this information and the decrypting key to the NetBill server. The NetBill server then debits customer's account, credits merchant's account and sends a receipt to the merchant. Finally, the merchant forwards this receipt containing the decrypting key to the customer. Note that, the NetBill protocol is not optimistic and a merchant who has provided a worthless good is detected only after the exchange occurs; this is in contrast to our work.

A fair exchange protocol ensuring the consistency of the document but requiring the active participation of a trusted third party has been proposed by Ketchpel [14]. The merchant and the customer after agreeing upon the product and the price sign a contract

which is forwarded to the third party. Each party then sends his item to the third party. The third party verifies that the items satisfy the contract, and then forwards them to the respective parties.

Franklin and Reiter [11] also propose a set of fair exchange protocols that verify the consistency of a document before the exchange takes place. The protocols use a one-way function f which has the 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. 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 to Y, and Y sending 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 $f(y_1)$; 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 items. One contribution of this paper is that the third party is semi-trusted (one that can misbehave on its own but will not collude with the participating parties) and will not be revealed the information that X and Y are trying to exchange. However, this protocol requires the active participation of the third party. We invoke the third party when a problem occurs; however, when invoked it is provided with all information pertaining to the items exchanged.

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. [5]. 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 digital items other than signatures. Asokan et al. [3] also provide an optimistic protocol for the fair exchange of digital signatures.

A more general optimistic protocol that allows exchange of any two digital items has been proposed by Asokan et al. [1]. The protocol begins by the two parties promising each other an exchange of items. If they agree on the terms of the exchange, the exchange takes place. In case of any failure or any party misbehaving, the recovery phase which involves the third party is initiated. One difference with our work is that our protocol provides true fair exchange whereas that by Asokan et al. does not. For this reason, the dispute resolution is outside the scope of Asokan's protocol. Another difference is that in Asokan's protocol a party can verify whether he has received the correct item after receiving the item; in our work, we can do this before receiving the item.

3 Theory for Cross Validation

Before presenting our protocol, we provide some background about the theory of cross-validation on which the protocol is based. For details the reader is referred to [18].

Definition 1. *The set of* messages M *is the set of non negative integers m that are less than an upper bound N, i.e.* $M = \{m | 0 \leq m < N\}$.

Definition 2. *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 3. *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. Two integers a, b are said to be* relatively prime *if their only common divisor is 1, that is,* $\gcd(a,b) = 1$*. The integers* n_1,n_2,...,n_k *are said to be* pairwise relatively prime, *if* $\gcd(n_i,n_j) = 1$ *for* $i \neq j$.

Definition 4. *The Euler's totient function* $\phi(N)$ *is defined as the number of integers that are less than N and relatively prime to N.*

Definition 5. *A* key *K is defined to be the ordered pair* $< e,N >$*, where N is a product of distinct primes and e is relatively prime to* $\phi(N)$*; e is the* exponent *and N is the* base *of the key K.*

Definition 6. *The* encryption *of a message* m *with the key* $K = < e,N >$*, denoted as* $[m,K]$*, is defined as* $[m,< e,N >] = m^e \mod N$.

Definition 7. *The* inverse *of a key* $K =< e,N >$*, denoted by* K^{-1}*, is an ordered pair* $< d,N >$*, satisfying* $ed \equiv 1 \mod \phi(N)$.

Theorem 1. *For any message m, where* $K =< e,N >$ *and* $K^{-1} =< d,N >$*,*

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

Corollary 1. *An encryption,* $[m,K]$*, is* one-to-one *if it satisfies the relation*

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

Definition 8. *Two keys* $K_1 =< e_1,N_1 >$ *and* $K_2 =< e_2,N_2 >$ *are said to be* compatible *if* $e_1 = e_2$ *and* N_1 *and* N_2 *are relatively prime. If two keys* $K_1 =< e,N_1 >$ *and* $K_2 =< e,N_2 >$ *are compatible, then the* product *key,* $K_1 \times K_2$*, is defined as* $< e,N_1N_2 >$.

Lemma 1. *For positive integers a,* N_1 *and* N_2*,* $(a \mod N_1N_2) \equiv a \mod N_1$.

Theorem 2. *For any two messages m and* $\hat{m}$*, such that* $m,\hat{m} < N_1,N_2$*,* K_1 *is the key* $< e,N_1 >$*,* K_2 *is the key* $< e,N_2 >$*,* K_1 *and* K_2 *are compatible, and* $K_1 \times K_2$ *is the product key* $< e,N_1N_2 >$*, the following holds:*

$$[m,K_1 \times K_2] \equiv [\hat{m},K_1] \mod N_1 \text{ if and only if } m = \hat{m} \tag{2}$$

$$[m,K_1 \times K_2] \equiv [\hat{m},K_2] \mod N_2 \text{ if and only if } m = \hat{m} \tag{3}$$

3.1 Product Validation

Our protocol performs product and payment token validation using the results of theorem 2. The product validation takes place as follows. Let *m* be the product to be delivered. The third party generates keys K_{M_1} and $K_{M_1}^{-1}$ and provides K_{M_1} to the merchant. The merchant provides the product *m* to the third party to be encrypted with K_{M_1} (where

$K_{M_1} =< e, N_1 >$) and placed at a public place, which we call the catalog, as an advertisement for m. When the customer decides to purchase m from the merchant, the customer acquires $T = [m, K_{M_1}]$ 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_2}, $K_{M_2}^{-1}$) such that K_{M_2} is compatible with K_{M_1} according to definition 8. The merchant provides the customer with $T' = [m, K_{M_1} \times K_{M_2}]$. The customer verifies that $[m, K_{M_1}]$ and $[m, K_{M_1} \times K_{M_2}]$ are encryption of the same message m by verifying: $T \equiv T' \bmod N_1$ as per equation (2). When satisfied, the customer sends the payment token. The merchant, in return, sends the decrypting key $K_{M_2}^{-1}$. The customer obtains m using $m = \left[T', K_{M_2}^{-1}\right]$. The proof of correctness follows from theorem 2. Using the same principle the bank can validate the payment token before signing it.

3.2 Security

In the theory presented in Section 3 if e is chosen small and a customer can guess e correctly, we have a security problem.[1] 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 $K_{M_2} =< e, N_2 >$, $K_{M_3} =< e, N_3 >$ and $K_{M_4} =< e, N_4 >$.

Let $N = N_1 \times N_2 \times N_3 \times N_4$. Knowing $m^e \bmod N_i$, for i $= 1 \ldots 4$, the attacker can, using the Chinese remainder Theorem [16], compute 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 [13]. However, since the customer does not know the value of e, this problem will not arise. Below we provide an additional mechanism using which the security will not be compromised even if 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 $[m, K_{M_1}]$ from the third party. Rather than sending $[m, K_{M_1} \times K_{M_2}]$ to the customer, the merchant sends the following: $[m.r, K_{M_1} \times K_{M_2}]$, $[r, K_{M_1}]$, where $m.r$ is the product of m with r. To validate the product, the customer multiplies $[m, K_{M_1}]$ with $[r, K_{M_1}]$ and the resulting product is compared with $[m.r, K_{M_1} \times K_{M_2}]$. 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 $K_{M_2}^{-1}$, the merchant now sends $K_{M_2}^{-1}$ and r^{-1} where r^{-1} is the multiplicative inverse of r modulo N_2. Using the decrypting key $K_{M_2}^{-1}$, the customer obtains $m.r \bmod N_2$. Multiplying this by r^{-1} the customer can retrieve m.

4 The Basic Protocol

We make the following assumptions in the protocol:

1. Encrypted messages cannot be decrypted without proper keys. Digital signatures cannot be forged. Cryptographic checksums ensure the integrity of messages.

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

2. All parties use the same algorithm for encryption and for generating cryptographic checksums.
3. Payment for product is in the form of a token, PT, that is accepted by the merchant.
4. A constant time out period known to all parties is used when a party waits for a message from another party.

The following table lists the notations used in the description of the protocol.

Table 1: Symbols used in protocol description

SYMBOL	INTERPRETATION
C, M, B and TP	Customer, Merchant, Bank and Trusted Third Party
A_{prv}, A_{pub}	A's private and public keys
$X \Longrightarrow Y : P$	X sends P to Y
$[X,K]$	Encryption of X with key K
$CC(X)$	Cryptographic Checksum of X
K_{C_1}	Key given by the customer's bank B
$K_{C_1}^{-1}$	Decrypting key corresponding to K_{C_1} kept with B
K_{C_2}	Key generated by C that is compatible with K_{C_1}
$K_{C_2}^{-1}$	Decrypting key corresponding to K_{C_2}
K_{M_1}	Key given by TP to M
$K_{M_1}^{-1}$	Decrypting key corresponding to K_{M_1} held with TP
K_{M_2}	Key generated by M that is compatible with K_{M_1}
$K_{M_2}^{-1}$	Decrypting key corresponding to K_{M_2}
r	Random number chosen by M for current transaction
r^{-1}	Multiplicative inverse of r modulo N_2, where N_2 is the base for key K_{M_2}
m	Product the customer purchases
PO	Purchase order used by C to order product m
PT	Payment token used to pay for the product

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

1. C **opens an account with** B**.** B generates a key pair K_{C_1}, $K_{C_1}^{-1}$, provides C with K_{C_1} and escrows $K_{C_1}^{-1}$ with itself. For any transaction that involves payment via B, C 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_{C_j}, $j \neq 1$, is compatible with K_{C_1}. In this paper, we assume C uses the *product key* $K_{C_1} \times K_{C_2}$.

2. M **registers with** TP**.** TP generates the key pair K_{M_1}, $K_{M_1}^{-1}$, provides M with K_{M_1} and escrows $K_{M_1}^{-1}$ with itself. For every product, m, that M wants to advertise in the catalog at this TP, M sends m and its description. TP performs the encryption before uploading $[m,K_{M_1}]$ on the catalog. In this manner TP is able to certify that the product meets its claim.

3. *C* **selects a product to purchase.** *C* downloads $[m, K_{M_1}]$ from the TP. Note that *C* does not have the product *m*, because he does not have the decrypting key $K_{M_1}^{-1}$. This $[m, K_{M_1}]$ will be used later by *C* to validate the product received from *M*.

4. *C* **and** *M* **agree upon a price for the product.** This step may require several messages exchanged between *C* and *M* at the end of which they agree upon a price.

Protocol Description

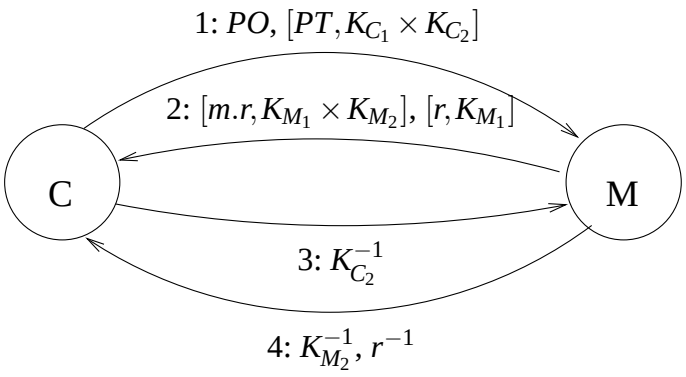

Fig. 1. The Basic Protocol

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

Message 1 $C \Longrightarrow M : PO, [CC(PO), C_{prv}], [[PT, K_{C_1} \times K_{C_2}], B_{prv}]$.

C initiates the e-commerce transaction by sending *M* the following items:

(a) The purchase order, *PO*, containing information about the product identifier, price of the product, identities of *C* and *M*, and a nonce to prevent replay attacks.

(b) A digitally signed cryptographic checksum of *PO*. (In this and the following messages, the purpose of a signed cryptographic checksum is twofold: (i) it can be used as evidence of what the sender has actually sent and (ii) it also ensures the integrity of the message while in transit.)

(c) The payment token, *PT*, containing information about the identities of *B*, *C*, customer's account information, price of the product and a nonce. The payment token is first encrypted with the *product key* $K_{C_1} \times K_{C_2}$ and then digitally signed by *B*. *B*'s signature ensures that *C* has sufficient funds to pay for the product. Note that, since *B* has the key $K_{C_1}^{-1}$, it is able to verify the contents of PT before signing.

Message 2 $M \Longrightarrow C : [Abort, M_{prv}]$

OR

$M \Longrightarrow C : [CC(PO), M_{prv}], [m.r, K_{M_1} \times K_{M_2}], [CC([m.r, K_{M_1} \times K_{M_2}]), M_{prv}]$, $[r, K_{M_1}]$, $[CC([r, K_{M_1}]), M_{prv}]$

M checks to see if PO is to its satisfaction – that is, M agrees to all its contents and is able to verify B's signature on $[PT, K_{C_1} \times K_{C_2}]$. If not, M aborts the transaction and sends C a signed abort message. A signed abort message provides a proof that the sender of the abort message is indeed terminating the transaction.

Otherwise, M sends the following to C: (a) a signed cryptographic checksum of the purchase order, (b) encrypted product, (c) signed cryptographic checksum of the encrypted product, (d) encrypted random number, and (e) signed cryptographic checksum of the encrypted random number.

Message 3 $C \Longrightarrow M : [Abort, C_{prv}]$

OR

$C \Longrightarrow M : [K_{C_2}^{-1}, M_{pub}], [CC([K_{C_2}^{-1}, M_{pub}]), C_{prv}]$

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

Otherwise C validates the product as outlined in Section 3.2. If the product is not validated, C aborts the transaction and sends M a signed abort message.

If the two compare, C sends the payment token decryption key encrypted by M's public key and a signed cryptographic checksum of the encrypted product decryption key.

Finally, C starts a timer and waits for **Message 4**. If the timer expires C executes the extended protocol given in Section 5.

Message 4 $M \Longrightarrow C : [K_{M_2}^{-1}, C_{pub}], [CC([K_{M_2}^{-1}, C_{pub}]), M_{prv}], [r^{-1}, C_{pub}],$
$[CC([r^{-1}, C_{pub}]), M_{prv}]$

If M receives an abort message from C in **Message 3**, it terminates the transaction. Otherwise, if M receives the decrypting key $K_{C_2}^{-1}$ from C, he decrypts the payment token PT and sends the following to C: (a) the product decryption key encrypted with the C's public key, (b) signed cryptographic checksum of the encrypted product decryption key, (c) the multiplicative inverse of the random number r also encrypted with C's public key, and (d) signed cryptographic checksum of the encrypted multiplicative inverse of the random number. Encrypting with C's public key in (a) and (c) ensures that no one else can get these items and decrypt the product.

Using the product decryption key and the multiplicative inverse of the random number, C can get the product as outlined in Section 3.2. The transaction terminates after C gets the product.

5 Extension for handling misbehaving parties and disputes

Note that if any party misbehaves or quits before C has send out the key, $K_{C_2}^{-1}$, for decrypting the payment token, fair exchange is not compromised. If any party misbehaves or quits after C has send out $K_{C_2}^{-1}$, then the extended protocol must be executed to ensure fair exchange. Note that since M sends the decryption key only after it has received payment in a satisfactory matter, it will always be the case that C initiates the extended protocol after the timer used in **Message 3** has expired. In the extended protocol C sends TP all the messages it received from M and the key $K_{C_2}^{-1}$ required to decrypt the payment token.

M **behaves improperly** This includes the following scenarios: (i) *M* receives **Message 3** but does not send the correct product decryption key $K_{M_2}^{-1}$ in **Message 4**, (ii) *M* receives **Message 3** but disappears without sending the product decryption key, and (iii) *M* claims that it did not send the correct decryption key because it has not received payment. In each case, TP asks *M* to send the product decryption key and starts a timer. If *M* does not respond within the timeout period, TP sends the key $K_{M_1}^{-1}$ to *C* and takes appropriate action against *M*. If *M* responds within the timeout by sending $K_{M_2}^{-1}$, TP forwards it to *C*. *M* 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, *M* still has to provide the TP with $K_{M_2}^{-1}$. TP sends $K_{C_2}^{-1}$ to *M* and $K_{M_2}^{-1}$ to *C*.

C behaves improperly This scenario occurs only when TP has sent $K_{C_2}^{-1}$ but that key still does not decrypt the payment token $[PT, K_{C_1} \times K_{C_2}]$. In this case, TP gets in touch with *B* to obtain $K_{C_1}^{-1}$. It then forwards this key to *M*.

Note that the following two disputes are not entertained: (i) *M* claims that it has received inadequate payment. The reason why it is not entertained is that *M* always sends the product decryption key after it has an opportunity to ensure that proper payment has been received. (ii) *C* claims after decrypting the product that the correct product was not provided by *M*. 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, *C* always has the option of aborting the transaction without paying.

6 Conclusion

In this work we have proposed an e-commerce protocol for performing business over the Internet. Our protocol has the following features. First, it provides true fair exchange 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 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 uses 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.

In future we plan to address how the many different forms of electronic payment schemes [17], can be incorporated in our protocol. Another important future work is evaluating the correctness of the protocol using formal methods of software verification like model checking [12] and theorem proving [7].

References

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

2. N. Asokan, V. Shoup, and M. Waidner. Asynchronous Protocols for Optimistic Fair Exchange. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 86–99, Oakland, California, May 1998.
3. N. Asokan, V. Shoup, and M. Waidner. Optimistic Fair Exchange of Digital Signatures. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Eurocrypt '98*, pages 591–606, Helsinki, Finland, June 1998.
4. A. Bahreman and J. D. Tygar. Certified Electronic Mail. In *Proceedings of the Internet Society Symposium on Network and Distributed Systems Security*, pages 3–19, February 1994.
5. F. Bao, R. H. Deng, and W. Mao. Efficient and Practical Fair Exchange Protocols with Off-line TTP. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, May 1998.
6. M. Blum. How to Exchange (Secret) Keys. *ACM Transactions on Computer Systems*, 1:175–193, 1983.
7. D. Bolignano. Towards the Formal Verification of Electronic Commerce Protocols. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop*, June 1997.
8. B. Cox, J. D. Tygar, and M. Sirbu. NetBill Security and Transaction Protocol. In *Proceedings of the 1st USENIX Workshop in Electronic Commerce*, pages 77–88, July 1995.
9. R. H. Deng, L. Gong, A. A. Lazar, and W. Wang. Practical Protocols for Certified Electronic Mail. *Journal of Network and System Management*, 4(3), 1996.
10. S. Even, O. Goldreich, and A. Lempel. A Randomized Protocol for Signing Contracts. *Communications of the ACM*, 28(6):637–647, June 1985.
11. M. K. Franklin and M. K. Reiter. Fair Exchange with a semi-trusted Third Party. In T. Matsumoto, editor, *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 1–6, Zurich, Switzerland, April 1997.
12. N. Heintze, J. Tygar, J. Wing, and H. Wong. Model Checking Electronic Commerce Protocols. In *Proceedings of the 2nd USENIX Workshop in Electronic Commerce*, pages 146–164, November 1996.
13. B. Kaliski and M. Robshaw. The Secure Use of RSA. *CryptoBytes*, 1(3):7–13, 1995.
14. S. Ketchpel. Transaction Protection for Information Buyers and Sellers. In *Proceedings of the Dartmouth Institute for Advanced Graduate Studies '95: Electronic Publishing and the Information Superhighway*, 1995.
15. S. Ketchpel and H. Garcia-Molina. Making Trust Explicit in Distributed Commerce Transactions. In *Proceedings of the 16th International Conference on Distributed Computing Systems*, pages 270–281, 1996.
16. I. Niven and H. S. Zuckerman. *An Introduction to the Theory of Numbers*. John Wiley and Sons, 4th edition, 1980.
17. D. O'Mahoney, M. Peirce, and H. Tewari. *Electronic Payment Systems*. Artech House, 1997.
18. I. Ray, I. Ray, and N. Narasimhamurthi. A Fair-Exchange Protocol with Automated Dispute Resolution. In *Proceedings of the 14th Annual IFIP WG 11.3 Working Conference on Database Security*, Schoorl, The Netherlands, August 2000.
19. J. Zhou and D. Gollmann. A Fair Non-repudiation Protocol. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 55–61, Oakland, California, May 1996.