Anonymous Internet Mercantile Protocol *

David M. Kristol
Steven H. Low
Nicholas F. Maxemchuk

AT&T Bell Laboratories, Murray Hill, NJ 07974

DRAFT: March 17, 1994

Abstract

Security and privacy protection become increasingly important as we enter the era of electronic commerce and information superhighways. We have applied the same principle of information separation in [7, 6] to design an anonymous mercantile protocol that can be used over the Internet for anonymous funds transfer from a customer to a seller and anonymous information delivery in the reversed direction. The idea is to separate all information needed for a transaction into different components and use cryptographic techniques to hide from a party those components that are not necessary for the party to perform its function. The protocol proposed here guarantees customers' anonymity while protects all parties including the seller from being cheated. A version of the protocol, based on the current FTP implementation, is being implemented.

*The authors can be reached at dmk,slow,nfm@allegra.att.com.

Contents

1 Introduction

Accelerated by the government initiatives on the National Information Infrastructure, networking is proliferating at an unprecedented rate. The same enabling computing, storage and communication technologies, however, also make it much easier to assemble more information on individuals and to invade their privacy. Security and privacy protection become more urgent in the era of electronic commerce when electronic market place replaces the conventional stores and electronic cash and credit replace the paper money. For instance, by monitoring the network traffic, it is easy to compile a profile of individuals' spending habits by associating purchases with a customer, unless techniques such as those proposed here are adopted. At some point, the ability to collect information compromises an individual's right and willingness to deviate from the norm. This paper proposes an anonymous mercantile protocol that can be used on the Internet to perform anonymous funds transfer from a customer to a seller and anonymous information delivery in the reversed direction.

The protocol achieves three objectives. First, the seller's interest is protected by transferring funds to the seller's account before information delivery. Furthermore, the information is encrypted so that it is useless to all but the paid customer. Second, the customer's anonymity is guaranteed by the use of information separation and cryptographic techniques. Third, the customer's interest is protected by the creation of a complete (but encrypted) audit trail that can be unravelled when necessary to settle any dispute. The anonymity of the customer the seller is dealing with may perhaps also deter the seller from cheating.

Here, anonymity roughly means that no party involved in the transaction can associate the customer with the purchase. The central idea is to separate all information needed for a transaction into different components and use cryptographic techniques to hide from a party those components that are not necessary for the party to perform its function. We believe anonymity is desirable for the following reasons. First, it protects individuals' privacy and their right to be different; this is especially important in a politically repressive environment. Second, the Internet is likely to be heterogeneous in the foreseeable future, with the underlying transport facilities provided by different vendors. A transaction between a customer and a company A may be carried on the facility of a competing company B. It is then to A's interest to hide the identity of the customer from B.

The protocol is an extension of the anonymous credit card protocol of [7, 6], tailored to its application on the Internet. While the basic idea of information separation is used in both protocols, there are major differences between the two. These are pointed out in §5 after the protocol description. An alternative to our approach that also provides payment anonymity is the use of digital cash first proposed in [3]; see also [2, 8, 1] and references therein. Since the electronic cash is given to a customer, a means is needed to prevent the individual from duplicating and spending it over and over again, and to prevent possible forgery. Our approach avoids these problems since funds are transferred only among trusted entities. Reference [5] designs a protocol to perform payment transactions between mutually distrustful parties over an insecure network. It does not, as we do, focus on guaranteeing customers' anonymity.

The set up and overview of the protocol is given in §2. The cryptographic tools, together with the notations used in the paper, are explained in §3. The protocol is described in §4. We omit many details in our description in order to expose the core idea; e.g. we only describe the messages that would have been exchanged during a successful transaction, but omit considerations on exception conditions. In §6, we describe several extensions to the protocol, such as depositing additional funds into the seller's bank in the middle of a data delivery phase, and using the protocol for small payments or for physical goods not delivered over the Internet, etc. Finally, we provide in the Appendix a formal protocol specification using communicating finite state machines.

2 Setup and overview

As in the anonymous credit card protocol in [7, 6], we consider four types of entities on the Internet: 1) sellers, 2) customers, 3) banks, and 4) electronic federal reserve and communication exchange cx. Every customer C (every seller S) maintains an account, denoted by (B_c, C) (respectively (B_s, S)), at bank B_c (respectively B_s). The forth entity cx should perhaps be considered two separate entities; we combine them into one only to simplify exposition. As a communication exchange, it allows two parties, e.g. a customer and a seller, to communicate without knowing each other's identity through the use of a simple cryptographic technique. As an elctronic federal reserve, it provides a means to transfer funds between banks (this function can be replaced by other electronic funds transfer means). Every bank has an account with cx. A funds transfer between two banks is achieved by updating their accounts at cx. From time to time, the accounts are settled up. When an imbalance occurs in the funds of a bank, cx requests assets from the bank or sends assets to the bank. Since banks do not have to transfer assets directly among themselves, they need not know each other's identity. All transfers are signed and logged to provide an audit trail in case a dispute arises.

A protocol session proceeds in three phases.[1] In phase 1, a customer C obtains anonymously the seller's account (B_s, S) and transfers funds anonymously to the account. C obtains a receipt from B_s certifying the deposit, which C presents to the seller S to open a session account at S. In phase 2, C may make multiple requests of files from the seller S. S deducts the purchase prices from the session account, encrypts the files with a key specified by the customer, and delivers them without knowing either the identity or the address of C. Phase 3 is entered when C finishes all desired purchases. Any balance in the session account is refunded anonymously to C's account (B_c, C).

After a protocol session, The seller only knows that it is paid for the purchase of certain files, but not the identity or the address of the customer. The customer's bank B_c only knows that money has been withdrawn from (and deposited to) the customer's account, but knows neither the purpose nor the source (and destination) of the transfer(s);

[1] By a session, we loosely mean the time between the starting (step 1 below) and termination of the protocol (step 12 below for a successful transaction). The underlying transport may be either connectionless or connection-oriented; 'a session' may or may not be associated with 'a connection' in the latter case.

similarly, the seller's bank B_s only knows that money has been transferred into and out of the seller's account; cx knows that funds transfers have taken place between B_c and B_s but does not know the source and destination accounts. Without collusion among them, no party can associate C with the purchase. Applying the collusion analysis method in [6] to the protocol, we find that steps 6 and 7 below are the most vulnerable steps: if B_s colludes with cx on the message in step 6, B_s can associate C with the seller S; if S colludes with cx on the message in step 7, S can associate C with the files purchased.

3 Cryptographic tools and notations

Each party A in the system has a pair of public key E_A and private key E_A^{-1}; see [4, 9] or [10] for public-key cryptography. We assume that there is a reliable way for any party that wishes to communicate with A to obtain A's public key E_A. $E_A(msg)$ (respectively $E_A^{-1}(msg)$) denotes the message msg encrypted with A's public key (respectively, private key). To ensure secrecy of communication one sends $E_A(msg)$ to A, so that only A can decrypt it with its private key.

Implicit in the notation $E_A(msg)$ is that msg is always appended with a random number before *every* encryption;[2] e.g. $E_A(B, E_B(msg))$ contains two random numbers, one for E_A and the other for E_B. This prevents one type of collusion by disallowing a player other than A that has msg from correlating $E_A(msg)$ with msg; see [6].

A message msg signed by A is denoted $Sgn_A(msg)$; precisely, $Sgn_A(msg) = [\ msg, t(A,B), E_A^{-1}(msg, t(A,B))\ ]$ is the message msg, followed by a time-stamp $t(A,B)$, followed by their encryption with A's private key E_A^{-1}. Anyone with A's public key can verify the signature but only A could have produced it. The time-stamp $t(A,B)$ increases for each successive signed message between the source A and destination B and prevents replay attacks.

The message passing in the protocol is specified in the form:

$$src \rightarrow dst \quad : \quad msg$$

denoting 'src sends the message msg to dst', src and dst being one of C, S, B_c, B_s, and cx. Though not explicit in the notation, it is understood that msg is always encrypted with the public key of dst so that an eavesdropper cannot understand an intercepted message.

A message can be in one of four formats:

1. msg, an unsigned message;
2. $Sgn_A(msg)$, a message msg signed by A;
3. $[D : msg]$, which is always a message sent to cx to be forwarded to D. The final destination D may either be in the clear or in encryption with cx's public key; msg is always in encryption with D's public key.

[2] A more precise notation is $E_A(msg, R)$ instead of $E_A(msg)$, where R is a random number A chooses.

4. $Sgn_A(D: \ msg)$, which is a funds transfer request signed by A and sent to cx. Here, A is the source bank, D is (or contains) the destination bank, and msg contains the amount of transfer. Upon receipt, cx will update A and D's accounts and forward the transfer to D; see below. D is always in encryption with cx's public key.

4 Protocol

As noted above, our description is brief in order to expose the core ideas. For example, we only describe the messages that would have been exchanged during a successful transaction, and omit considerations on exception conditions (see also the Appendix). We also only describe the protocol as it is used to purchase multiple files over the Internet from the same seller. It can be easily modified to pay for physical products whose delivery is not via the Internet, as commented in §6.

A protocol session proceeds in three phases: a funds-transfer phase in which C anonymously transfers funds to seller's account (B_s, S) and creates a session account at the seller; a information-delivery phase in which zero or more files are purchased and delivered; and a refund phase in which the balance in the session account is refunded to customer's account (B_c, C).

4.1 Phase 1: funds transfer

To transfer funds from his account (B_c, C) to a seller S's account (B_s, S), C first asks S anonymously for its account (B_s, S) in steps 1–2. These steps can be omitted if C already knows (B_s, S) through other means, e.g., directory service.

Obtaining seller's account

1. C asks for seller's account (B_s, S) by sending to cx the request $E_s(req, E_{cx}(C), E_c)$ encrypted with S's public key:

$$C \rightarrow cx \quad : \quad [\ S: \ E_s(req, E_{cx}(C), E_c)\ ]$$

and cx forwards $E_s(req, E_{cx}(C), E_c)$ to S. Here, req may contain the type of files C is interested in, $E_{cx}(C)$ is C's encrypted address and E_c is C's public key for the seller S to encrypt its reply to C.

2. Seller S encrypts its account with key E_c and replies to C's encrypted address:

$$S \rightarrow cx \quad : \quad [\ E_{cx}(C): \ E_c(resp, B_s, S)\ ]$$

which cx forwards to C. Here, $resp$ may be a menu of files and prices. Note that the customer's identity C is hidden from S.

Enquiries about the files and negotiation of prices can proceed in like manner at this point.

Transferring funds

From (B_s, S), C computes the following encryption $D(B_s, S, E_c, E_{cx}(C)) = E_{cx}(B_s, E_{B_s}(S, E_c, E_{cx}(C)))$ and initiates a funds transfer from (B_c, C) to (B_s, S) (steps 3–6). B_s will be used by cx to transfer funds between banks B_c and B_s; S will be used by B_s to deposit funds to the seller's account; the key E_c and encrypted address $E_{cx}(C)$ will be used by B_s to encrypt and send a receipt to C without knowing C. These steps are illustrated in Figure 1.

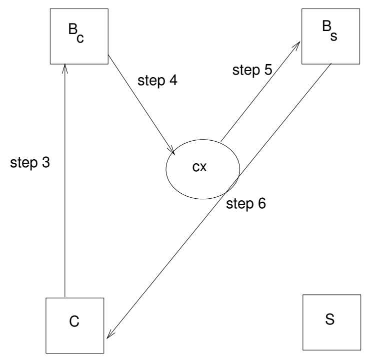

Figure 1: Fund transfer

3. C initiates a funds transfer by sending to B_c the signed request

$$C \rightarrow B_c \quad : \quad Sgn_c(PIN, \$, D(B_s, S, E_c, E_{cx}(C))).$$

Here, *PIN* is the personal identification number to authenticate C to bank B_c, and \$ is the amount of transfer that C requests, and $D(\cdots)$ contains the destination account (B_s, S) in encryption. Hiding store's bank B_s from B_c prevents them from collusion to associate C with S.

4. B_c updates C's account and sends to cx the signed funds transfer

$$B_c \rightarrow cx \quad : \quad Sgn_{B_c}(D(B_s, S, E_c, E_{cx}(C)) : \; \$)$$

5. First cx decrypts $D(\cdots)$ to obtain the destination bank B_s. It verifies the banks B_c and B_s and the signature, and then transfers the amount \$ from B_c's account to B_s's account at cx. It sends to B_s the message

$$cx \rightarrow B_s \quad : \quad Sgn_{cx}(E_{B_s}(S, E_c, E_{cx}(C)), \$)$$

signed and guaranteed by cx. Note that both C and S are hidden from cx.

6. B_s updates the seller's account, generates a receipt consisting of a new session account number SA (see below) and the balance \$, signs the receipt, and encrypts it with E_c. It then sends the message

$$B_s \rightarrow cx \quad : \quad [\; E_{cx}(C) : \; E_c(Sgn_{B_s}(SA, \$)) \,]$$

to cx, which forwards the encrypted receipt $E_c(Sgn_{B_s}(SA, \$))$ to C.

B_s at this point may also send to S the same receipt for S to cross-check the receipt presented by C when requesting a file. The session account number $SA = (B_s, S, i)$ where i denotes the ith session account ever created for account (B_s, S).[3] The strictly increasing last field i of SA ensures that all active session accounts in the *entire* system have distinct numbers at all times.

4.2 Phase 2: information delivery

S keeps a list of active session accounts and a list of terminated session accounts. Each session account is a pair (account number SA, current balance).

7. To use the funds to purchase a file, C sends to cx the message

$$C \rightarrow cx \quad : \quad [\ S:\ E_s(file\_name, Sgn_{B_s}(SA, \$), E_c, E_{cx}(C))\ ]$$

and cx forwards $E_s(\cdots)$ to S. The message contains the name of the file C wants to purchase, the receipt from B_s, the key E_c to encrypt the file and the address $E_{cx}(C)$ to which to send the file.

Encrypting the purchased file not only provides secrecy for the customer, but also protects the seller's interest since otherwise cx can obtain the information for free.

8. After verifying B_s's signature on the receipt, S checks the session account number SA against its lists of active and terminated session accounts. If it is on the list of terminated accounts, B_s rejects the request because the receipt is a replay for a terminated session; otherwise, B_s adds $(SA, \$)$ to the list of active accounts if it is not already on that list. S then deducts the purchase price of the file from the balance in the account[4] and sends the requested file to C via cx:

$$S \rightarrow cx \quad : \quad [\ E_{cx}(C):\ E_c(file)\ ].$$

Steps 7–8 may be repeated many times until C has purchased all desired files. See §6 for steps to deposit additional funds in the middle of a session.

4.3 Phase 3: refund

To close the session, C sends to S, via cx as in step 7, a 'quit' signal, the receipt $Sgn_{B_s}(SA, \$)$ from B_s, and C's bank account in encryption $D(B_c, C) = E_{cx}(B_c, E_{B_c}(C))$. Then S closes the session after returning the balance in the session account to (B_c, C). It then moves the session account from the active account list to the terminated account list. The steps for refund are illustrated in Figure 2.

[3] This field can also be replaced by time-stamp.

[4] Note that the receipt $Sgn_{B_s}(SA, \$)$ contains the opening deposit rather than the current balance in the session account.

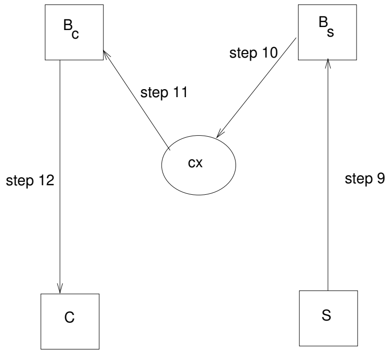

Figure 2: Refund

9. S initiates the refund by sending to B_s the signed message

$$S \to B_s \quad : \quad Sgn_s(PIN, balance, D(B_c, C)).$$

Here, *PIN* authenticates S to B_s, *balance* is the remaining balance in the session account SA, and $D(B_c, C)$ is the destination account for the refund.

10. B_s returns the refund by sending to cx the signed funds transfer

$$B_s \to cx \quad : \quad Sgn_{B_s}(D(B_c, C) : \ balance).$$

11. After verifying the signature, cx updates the accounts of B_s and B_c and sends the signed message to B_c:

$$cx \to B_c \quad : \quad Sgn_{cx}(E_{B_c}(C), balance).$$

12. B_c increments C's account by the amount *balance* and sends a signed acknowledgement to C:

$$B_c \to C \quad : \quad Sgn_{B_c}(ack).$$

5 Comparison with ACC

In this section, we compare the Internet mercantile protocol (IMP) in §4 with the anonymous credit card (ACC) protocol in [7, 6].

The central idea in both protocols is to separate all information needed for a transaction into different components and use cryptographic techniques to hide from a party those components that are not necessary for the party to perform its function. Several features of ACC can also be incorporated here. For instance, before making funds transfer in step 4 of the IMP, B_c may either accept *PIN* or make one or more random and personalized challenges, depending on the amount of transfer, using the techniques in the ACC protocol

[7, 6]. The use of a Swiss bank in ACC that knows the customer only by a pseudonym can also be adopted here for higher security.

Despite many similarities, there are some major differences between the two protocols. First, unlike the ACC, there is an information transfer phase in the mercantile protocol, in which the files are transported over the Internet from the seller to the customer. For this purpose, customer C has to provide its public key E_c and an encrypted address $E_{cx}(C)$ for the seller to encrypt and deliver the files without knowing C (steps 7 and 8). The encryption also protects the seller's interest since otherwise eavesdroppers can get the files for free. These steps are not necessary for the ACC as the customer is physically in the store.

Second, there is also a refund phase in the IMP. This is absent in the ACC where an exact funds transfer is done during a purchase. The approach here is more convenient since a customer can make a funds transfer at the beginning of a session without knowing the exact files to purchase, and then browse through an electronic catalog to decide which files to purchase. It also reduces communication overhead by consolidating the purchases of several files; this may prove to be very convenient for multiple small payments, see §6.

Finally, the customer has direct access to the network in the IMP, whereas the customer has access only through the seller's equipment in ACC. Hence, the funds-transfer request is sent directly to B_c (step 3) in IMP, instead of via cx as in ACC.

6 Extensions

6.1 Variable degree of anonymity

In the case where C logs on the system through a machine not bound to C's identity (e.g. through a proxy or public terminal), or if C is willing to allow S to associate C's network address with the purchase, the protocol can be simplified by allowing C and S to communicate directly rather than through cx in steps 1–2 and 7–8; also in this case B_s may send the receipt to C directly rather than through cx in step 6.

6.2 Deposit additional funds

If the session account runs out of funds in the middle of a session, C can deposit additional funds into the seller's bank account for more purchases, as follows. C repeats steps 3–6 to transfer funds to S's bank account, and receives from B_s a new session account number and its receipt. The same steps 7–8 can be used to purchase more files using the new session account. To close the session, a 'quit' message is sent to S, via cx, containing receipts for all session accounts belong to C, which triggers S to initiate refunds for these accounts, as in steps 9–12.

Note that if the customer have transferred funds from different banks to open different session accounts, the customer can specify to which bank the balance of each sessoin account should be refunded.

6.3 Small payments

If a customer will be buying from the same seller over a long period of time, the customer can choose to leave the session account active over the entire period in order to further reduce the communication and processing overhead with the banks. This may prove very convenient if each purchase only costs a small amount. Another useful option is to have a default idle period (or alternatively, lifetime), say a month, for each session account. If a customer does not initiate the termination of a session account, the seller will automatically refund and close the account (steps 9–12) if it has been idle for a month (or if its lifetime expires). In this case, the seller stores the customer's account $D(B_c, C)$ in encryption in the session account when it is first opened.

6.4 Purchase of physical products

If the protocol is used to purchase physical products, e.g. books, CDs, not to be delivered over the Internet, phase 3 is typically omitted as it is likely that exact amount is paid in phase 1. Obviously the key E_c and address $E_{cx}(C)$ in steps 7–8 may have to be modified depending on the delivery method. Note that regardless of whether the goods is delivered over the Internet, the protocol can be extended using the same information separation technique to allow the goods to be returned and money refunded anonymously.

6.5 Replay attack

Though a replay of the file transfer request in step 7 only transfers encrypted files to C, it may be desirable to guard against such replays by eavesdroppers or by accident as follows. S associates with each active session account an access number which is the number of times the account has been charged since creation. C updates and includes this access number in its file transfer request, and S honors a request only if the incoming access number is larger than the one stored in the session account.

7 Conclusion

Security and privacy protection become increasingly important as we enter the era of electronic commerce and information superhighways. We have applied the same principle of information separation in the ACC protocol in [7, 6] to design an anonymous Internet mercantile protocol. The idea is to separate all information needed for a transaction into different components and use cryptographic techniques to hide from a party those components that are not necessary for the party to perform its function. The protocol proposed here guarantees customers' anonymity while protects all parties including the seller from being cheated. We are currently implementing a version of the protocol on the Internet to augment the current FTP (file transfer protocol) implementation into a mercantile FTP.

8 Appendix

In this appendix we sketch the protocols the communication exchange cx, a bank B, and a seller S runs at their individual sites. The protocols are specified in Figures 3-6 as communicating finite state machines.

As an abbreviation, a transition labelled by $a * b$ denotes a transition labelled by a followed by a transition labelled by b; a transition labelled by $a + b$ denotes two transitions between the same two states one labelled by a and the other by b. Several terms in the specification, such as $src(\_bank)$, $dst(\_bank)$, msg, and msg' in cx's protocol, and ac_owner (account owner) in bank's protocol, etc., are generic and assume different entity at different times.

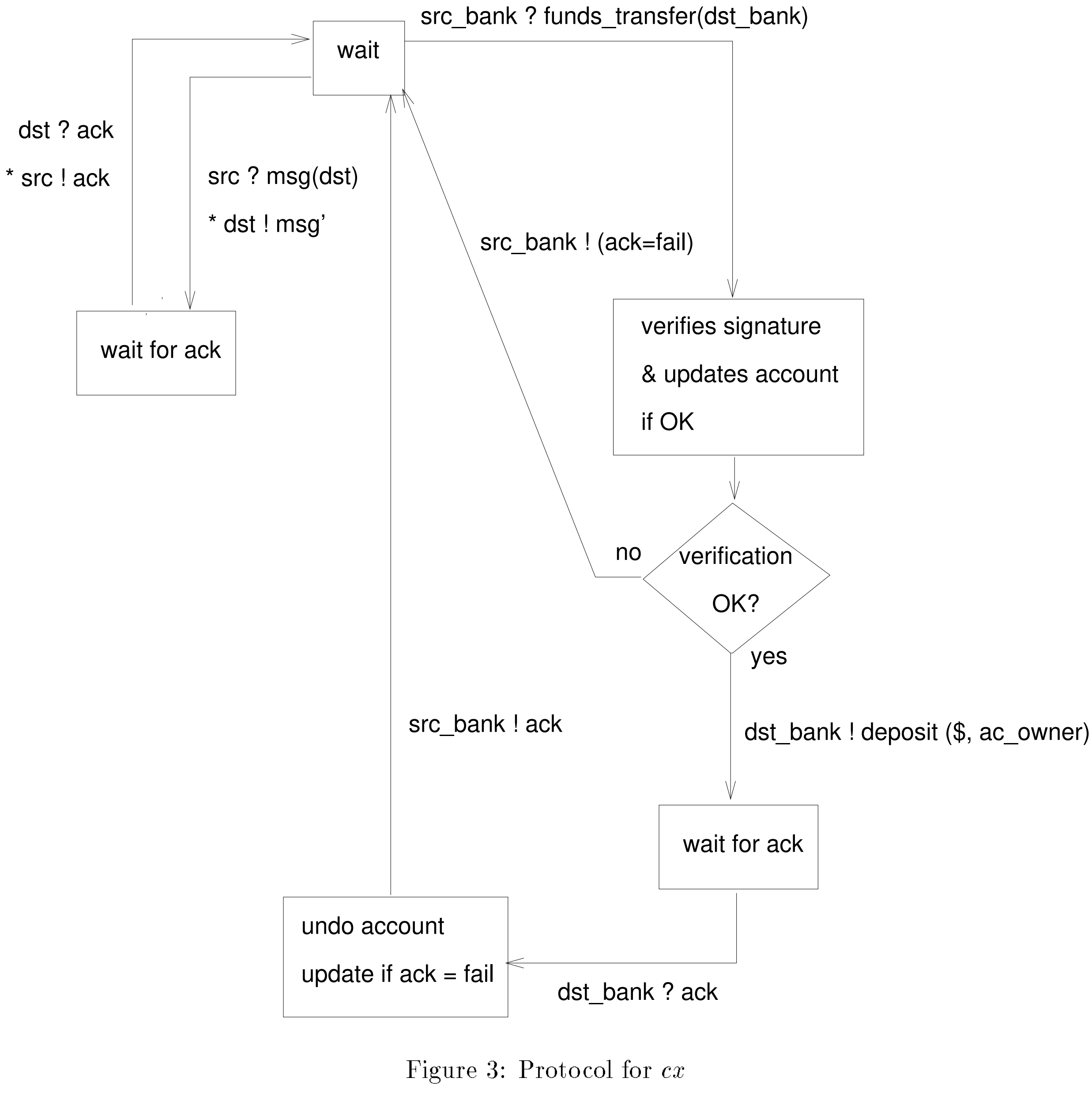

Figure 3: Protocol for *cx*

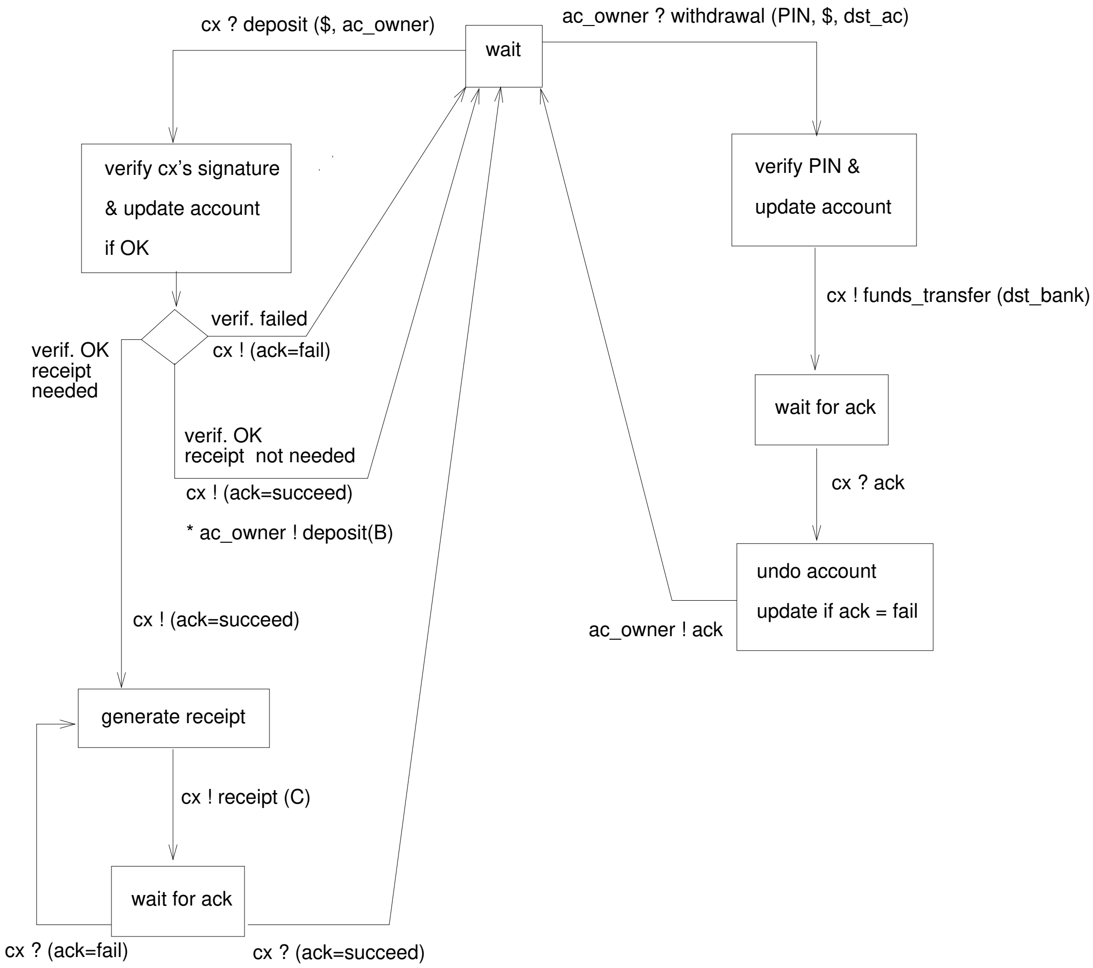

Figure 4: Protocol for a bank B

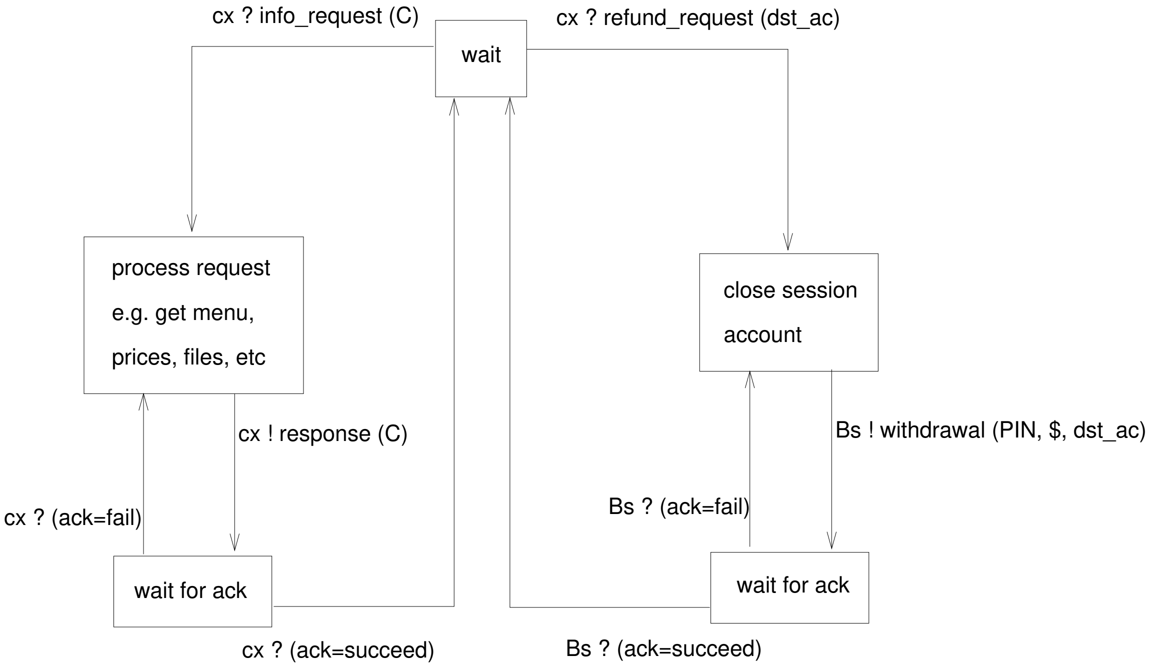

Figure 5: Protocol for a seller S

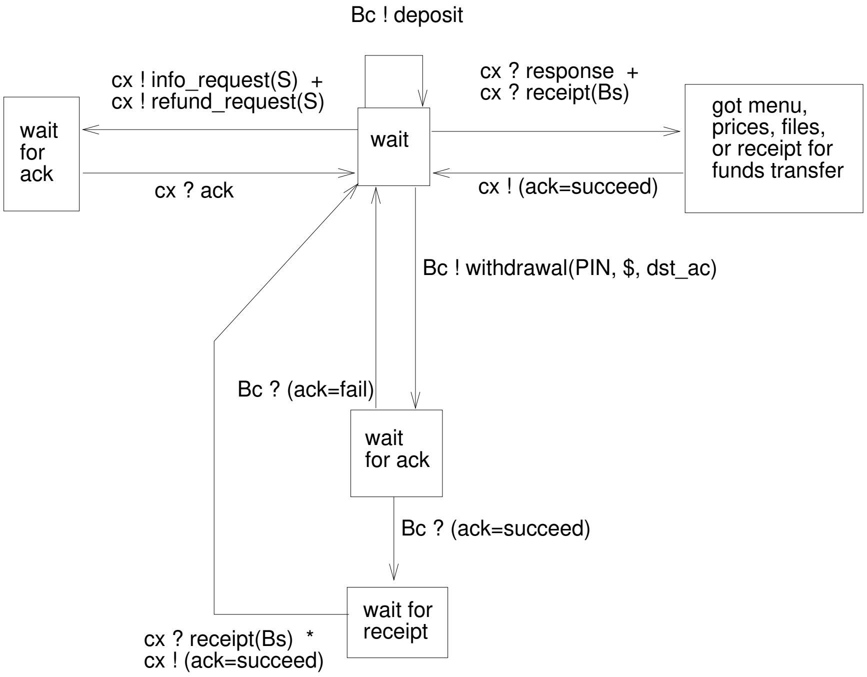

Figure 6: Protocol for a customer C

References

[1] Stefan Brands. An efficient off-line electronic cash system based on the representation problem. Technical report CS-R9323, CWI, Amsterdam, The Netherlands, March 1993.

[2] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology - CRYPTO '88*. Springer-Verlag, 1988. LNCS 403.

[3] David Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of ACM*, 28(10), October 1985.

[4] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22, November 1976.

[5] Semyon Dukach. SNPP: A Simple Network Payment Protocol. In *Proceedings of the Computer Security Applications Conference*, San Antonio, TX, November 1992.

[6] S. H. Low, N. F. Maxemchuk, and S. Paul. Collusion in a multiparty communication protocol for anonymous credit cards. Submitted to IEEE/ACM Transactions on Networking.

[7] S. H. Low, N. F. Maxemchuk, and S. Paul. Anonymous credit cards. Submitted to Globecom'94, 1994.

[8] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In *Advances in Cryptology - CRYPTO '91*. Springer-Verlag, 1992. LNCS 576.

[9] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of ACM*, 21, February 1978.

[10] Gustavus J. Simmons, editor. *Contemporary Cryptology: The Science of Information Integrity*. IEEE Press, 1992.