

Anonymous Credit Cards and Their Collusion Analysis

Steven H. Low, *Member, IEEE*, Nicholas F. Maxemchuk, *Fellow, IEEE*,
and Sanjoy Paul, *Member, IEEE*

Abstract— Communications networks are traditionally used to bring information together. They can also be used to keep information apart in order to protect personal privacy. A cryptographic protocol specifies a process by which some information is transferred among some users and hidden from others. We show how to implement anonymous credit cards using simple cryptographic protocols. We pose, and solve, a collusion problem which determines whether it is possible for a subset of users to discover information that is designed to be hidden from them during or after execution of the anonymous credit card protocol.

I. INTRODUCTION

IN THIS paper, we describe a way to implement a credit card system that preserves personal privacy using simple cryptographic building blocks. In our system, the organization that extends credit knows a cardholder's identity and the store knows the purchases. The protocol allows each to perform its function without either knowing both pieces of information. Thus, e.g., spending habits of individuals cannot be compiled, as they routinely are in current systems.

Complex communications protocols employing cryptographic building blocks are being developed not only to communicate, but also to protect privacy, e.g., in broadband networks [17], [18], in mobile networks [6], in electronic commerce [5], [9], [13], and in health insurance systems [12]. We view a cryptographic protocol as a process by which information is transferred among some users and hidden from others. The collusion problem determines whether a subset of users can discover, through collusion, the information that is designed to be hidden from them during or after a protocol's execution. We have introduced a formal model for collusion analysis, and formulated and solved the collusion problem [8], [10], [11]. The second purpose of this paper is to apply the results to analyze the collusion property of the anonymous credit card protocol. For ease of exposition, we only describe and analyze in detail a subpart of the protocol that constitutes a successful transaction. For a more complete protocol description, the reader can see [9].

Manuscript received September 14, 1993; revised October 14, 1994 and January 3, 1996; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. S. Lam.

S. H. Low is with the Department of Electrical and Electronic Engineering, The University of Melbourne, Australia (email: slow@ee.mu.oz.au).

N. F. Maxemchuk is with AT&T Laboratories, Murray Hill, NJ 07974 USA (email: nfm@research.att.com).

S. Paul is with Bell Laboratories, Lucent Technologies, Holmdel, NJ 07733 USA (email: sanjoy@bell-labs.com).

Publisher Item Identifier S 1063-6692(96)08934-0.

Communication networks are traditionally used to bring information together. Here, we use it to keep information apart. In constructing the system, we trust the various parties in the network not to collude with one another to join the information that we are trying to separate. However, our trust is not always warranted in that, either because of human intervention or implementation error, parties may be compromised. The ease with which parties can collude is an indication of how well the system separates the information.

The idea of the anonymous credit card protocol is simple. The credit card company knows the individual and trusts the individual to pay her bills. The credit card company places funds in an anonymous account, in a different bank, and the individual spends funds from this account. When an individual makes a purchase at a store, funds are transferred from the anonymous account to the store's account. Once the store is convinced that it has been paid, it has no need to know the individual's identity. Periodically, the bank that maintains the anonymous account bills the credit card company, which bills the individual.

In order to make the system more difficult to break, a double-locked box protocol is used to transfer funds between accounts. The funds are transferred through an intermediary in such a way that neither bank knows the identity of the other, and only the bank that maintains an account knows the account that the funds are transferred into or out of. The user deposits a box in the source account that can only be opened by the intermediary. Inside this box is the name of the destination bank and a second box that can only be opened by that bank. The second box has the name of the account in that bank.

In Section II, we describe the double-locked box and two building block protocols on which the anonymous credit card is built. These protocols assume the existence of a point-to-point protocol that provides certain standard cryptographic functions.

In Section III, we present the anonymous credit card protocol. For simplicity, we ignore many functions that a credit card system performs beyond extending credit and collecting bills. They are addressed in [9].

In Section IV, we briefly review the collusion model introduced in [10] and [11], and then apply it to the analysis of the anonymous credit card protocol. We draw conclusions in Section V.

We close this section by remarking on the anonymous credit card system. The use of intermediaries to hide information is an application of Chaum's idea in [3], in which an intermediary

forwards electronic mail between two communicating parties, and hides the sender's identity from the receiver. There are two differences between the current application and the earlier application of intermediaries. First, the communication in the present system is always between two specific parties, while the earlier system was designed for general communications. The limited communications connectivity makes it reasonable to precalculate all of the boxes that a source may need, instead of using a public-key cryptosystem. Second, because the current system transfers funds between banks, the intermediary must perform the function of the Federal Reserve. The intermediary determines that the source is a valid bank, guarantees the fund's transfer to the destination, and does the bookkeeping necessary to settle accounts. In the earlier application, the intermediary did not operate on the information between the source and destination.

To date, most work on electronic funds transfer mechanisms that protect a customer's privacy has been on digital cash mechanisms, e.g., [2], [4], and [16], and the references therein. The anonymous credit card protocol protects a cardholder's privacy while providing the convenience and security of conventional credit cards. The ability to link purchases and identity in our system may make it reasonable to increase personal privacy in some applications. For instance, one of the major problems in electronic publishing is the ability to copy and redistribute electronic documents. A mechanism that has been proposed in [1] and [7] to discourage redistribution makes each copy of a document unique and associates the document with the original recipient. If illegitimate copies are found, the original recipient can be determined. A problem with this approach is that publishers assemble reading profiles for individuals. A commercial company may consider such profiles very sensitive since it may not want its competitors to know what its employees are reading. If the publishers receive payment for the document with the anonymous credit card, they do not know who received the document. However, if illegitimate copies are located a subpoena can be issued to require the necessary parties to join their information to determine who received the original. The ability to join the identity and purchase when illegal copies are recovered makes it more reasonable to keep the information separate under normal conditions.

II. BUILDING BLOCKS

Our credit card system consists of the following entities: banks, customers, stores, and an intermediary we call the communication exchange. Customers maintain accounts with banks. Banks maintain accounts with the communication exchange; the communication exchange also serves as the Federal Reserve for funds transferred among banks. In the following subsections, we describe two building block protocols on which the anonymous credit card protocol is based. They are used for bank-to-bank communication and customer-to-bank communication.

The system can be based entirely on a symmetric key cryptosystem (e.g., DES) or on a public key cryptosystem (e.g.,

RSA), or a mix of both. This paper assumes a symmetric key cryptosystem; for a public key implementation, see [9].

A. Notation and Assumptions

The account of a customer or store X at a bank B is denoted (B_x, X) . For parties X and Y in the system, $K_{X,Y}$ denotes an encryption key shared only between X and Y in a symmetric key cryptosystem. $K_{X,Y}(m)$ denotes the encryption of a message m with key $K_{X,Y}$. It can be decrypted only by X or Y . $K_{X,X}$ denotes a key known only to X . Throughout the paper, we use the notation

$$X \rightarrow Y : m \quad (1)$$

to mean that X sends Y a message whose body is m . The header of the message contains X, Y among other things. As will be seen, m is frequently a ciphertext. The concatenation of two messages m_1 and m_2 is denoted m_1, m_2 .

We will describe a protocol informally as a sequence of two-party message exchanges of the form $X \rightarrow Y : m$. We assume that standard cryptographic techniques are used to guarantee that:

- 1) only the intended receiver Y can decrypt the message m (encrypted with key $K_{X,Y}$);
- 2) the receiver Y can verify m 's integrity and whether it originated with X ;
- 3) the sender X knows if Y receives the message correctly;
- 4) all messages are unique so that a replay attack is not possible.

Given that the underlying cryptosystem is secure, there are techniques in the literature that provide these functions (see, e.g., [14]). For simplicity, provisions for these techniques are not explicit in our notation. In effect, we assume a lower protocol layer that provides such functions. For instance, Property 3 requires that Y sends an acknowledgment to X . In order to guarantee that the acknowledgment is generated by the intended receiver, the acknowledgment may be a one-way hash of the message m encrypted with key $K_{X,Y}$. Property 4 can be provided through timestamping or other techniques (see, e.g., [14, pp. 997] and [15]). In order to achieve these characteristics, additional information is implicitly included in messages, and additional exchanges might occur between the source and destination of the message.

B. Double-Locked Box

Central to our system is a device we called a double-locked box. Each bank account has an associated double-locked box. It allows a message or funds to be deposited into that account without the receiving bank and the sender knowing each other, as will be seen.

Suppose a customer X has an account with bank B_x denoted by (B_x, X) . The associated double-locked box is constructed as follows. Customer X asks her bank B_x to encrypt her account number and give her the encryption $K_{B_x, B_x}(X)$. As defined in Section II-A, K_{B_x, B_x} is a key known only to B_x . X then asks the communication exchange cx to encrypt $[B_x, K_{B_x, B_x}(X)]$ with key $K_{cx, cx}$ known only to cx . We call

the resultant encryption a double-locked box of the account and denote it by

$$DLB(B_x, X) := K_{cx, cx}[B_x, K_{B_x, B_x}(X)].$$

Note that X cannot directly validate the content of $DLB(B_x, X)$. In Section II-C, we describe a way for X to perform such a validation.

If a public key system is used, X can compute $DLB(B_x, X)$ without involving her bank nor cx provided that she can obtain the appropriate public keys securely. Then, $K_{X, X}$ in a DLB should be interpreted as X 's public key.

Double-locked box is an application of Chaum's idea in [3] to hide a sender's identity from any receiver whom the sender knows and wishes to communicate with. Here, we use it to allow a sender and a receiver to communicate without knowing each other. This is possible because a sender in our system is always given a precomputed box that contains the destination account and, hence, never needs to know the receiver in order to construct the box.

C. Bank-to-Bank Communication

In our system, a bank does not interact with another bank directly, but only through the communication exchange. The first building block protocol allows a bank B_x to transfer funds and a message from its customer X 's account (B_x, X) to another customer Y 's account (B_y, Y) at another bank B_y . After the protocol is executed, the receiving bank B_y is assured that the transfer originated with a valid bank, but neither bank knows the other.

We assume that every bank B shares a secret key $K_{B, cx}$ with cx . The source account (B_x, X) has available the double-locked box $DLB(B_y, Y)$ of the destination account. In the following, m may be null, in which case only funds transfer occurs.

Protocol 1 Bank-to-Bank $[(B_x, X), (B_y, Y), \$, m]$:

- 1) B_x encrypts $DLB(B_y, Y)$ and m with key $K_{B_x, cx}$ shared between B_x and cx and sends it to cx

$$B_x \rightarrow cx : K_{B_x, cx}(DLB(B_y, Y), \$, m).$$

- 2) cx finds out the source B_x from the message header, decrypts the body with the shared key, and is assured that the transfer originated with B_x . It decrypts the outer encryption in $DLB(B_y, Y)$ to determine the destination bank B_y .

Besides relaying messages, cx also serves as the electronic Federal Reserve to transfer funds between banks. Every bank has an account at cx ; cx decrements source bank B_x 's account and increments destination bank B_y 's account by $\$$.

It encrypts m and the inner box with key $K_{B_y, cx}$ shared with B_y , and sends it to B_y :

$$cx \rightarrow B_y : K_{B_y, cx}(K_{B_y, B_y}(Y), \$, m).$$

Bank B_y decrypts the inner box $K_{B_y, B_y}(Y)$ and can associate m with its intended account (B_y, Y) . Assured by cx that funds has been transferred to its account, B_y increments account (B_y, Y) by $\$$.

Even though B_y does not know the source of the message, it is assured by cx that it originated with a valid bank. In addition, because of the assumptions in Section II-A on the underlying protocol, cx and B_y are assured that the messages they receive have not been altered and are not replays of earlier messages. The senders B_x and cx are assured that the messages that they have transmitted have been correctly received.¹

D. Customer-to-Bank Communication

The second building block protocol allows X to send a message m to her account (B_x, X) from a location S , in a way such that the following hold true: 1) bank B_x is assured that m originated with X , 2) B_x does not know where the message is sent from, and 3) S does not know where the message is sent to. This protocol is used when X sends a funds transfer request to her bank from a store and does not want either the store or her bank to know the other.

We assume that every customer X shares a secret key $K_{B_x, X}$ with her bank B_x , and the location S shares a key with cx .

Protocol 2 Customer-to-Bank $[S, B_x, X, m]$

- 1) X encrypts m with $K_{B_x, X}$ and gives it to (the computer at location) S together with $DLB(B_x, X)$. S encrypts them with a key shared with cx and sends the ciphertext to cx :

$$S \rightarrow cx : K_{S, cx}(DLB(B_x, X), K_{B_x, X}(m)).$$

- 2) The communication exchange decrypts the outer encryption in $DLB(B_x, X)$ and sends the inner box with the encrypted message to B_x

$$cx \rightarrow B_x : K_{B_x, cx}(K_{B_x, B_x}(X), K_{B_x, X}(m)).$$

The secret key $K_{B_x, X}$ assures B_x that no one but X could have originated m . The intermediary cx hides the store and B_x from each other.

In addition to making purchases at a store, the customer can use this protocol to verify that the double-locked box $DLB(B_x, X)$ constructed by B_x and cx is correct. X constructs a message $m = \{n, a\}$ where n is a nonce and a is the address X wants a response sent to. X sends m to account (B_x, X) using the double-locked box and the customer-to-bank $[S, B_x, X, m]$ protocol. Bank B_x is to return n to address a . Since the message m is encrypted with a secret key that is shared by B_x and X the correct response can be returned only if $DLB(B_x, X)$ is correct.

¹As discussed in Section II-A, we assume that cx acknowledges B_x in Step 1) and B_y acknowledges cx in Step 2). If communication in Step 2) is unsuccessful, cx will either retry until it succeeds or inform B_x of the failure using some other mechanism.

III. ANONYMOUS CREDIT CARD PROTOCOL

A customer C maintains accounts at two different banks B_c and B_p . Bank B_c issues the anonymous credit card and knows the identity of C . Bank B_p only manages money that has been deposited in P 's account. Since it does not extend credit, bank B_p knows the customer only by the pseudonym P . When the accounts are set up, C places $DLB(B_p, P)$ of the anonymous account (B_p, P) in her credit card account (B_c, C) ; she also places $DLB(B_c, C)$ in her anonymous account (B_p, P) . In the following, P and C both refer to the same customer.

At the beginning of a billing period, B_c transfers funds to the anonymous account (B_p, P) using $DLB(B_p, P)$ and the bank-to-bank $[(B_c, C), (B_p, P), \$_1]$ protocol. Here, $\$1$ is the amount of credit B_c is willing to extend to C .

To use these credits to make purchases at a store S , P requests bank B_p to transfer funds from account (B_p, P) to the store's account (B_s, S) at possibly another bank B_s . We abuse notation to use S to denote both the store and the unique network address (of the point-of-sale terminal) from which P sends her request. The double-locked box $DLB(B_s, S)$ is available at S . The steps to make a purchase of amount $\$3$ are as follows.

- 1) P uses the protocol customer-to-bank $[S, B_p, P, m]$ to send a funds transfer request to account (B_p, P) . Here, $m = \{\$3, DLB(B_s, S), K_{B_s, S}(n, \$3)\}$ contains the amount $\$3$ and destination account (B_s, S) . $K_{B_s, S}(n, \$3)$ binds the amount to a transaction number which will be returned to the store S by B_s as an acknowledgment.
- 2) B_p deducts $\$3$ from account (B_p, P) and transfers this amount to account (B_s, S) using the protocol bank-to-bank $[(B_p, P), (B_s, S), \$3, K_{B_s, S}(n, \$3)]$.
- 3) The store's bank B_s deposits the funds into account (B_s, S) and sends an acknowledgment to the address S : $B_s \rightarrow S$: $K_{B_s, S}(n)$. The store then releases the merchandise to P .

At the end of a billing period, B_p compiles a bill for the anonymous account (B_p, P) by adding up the amount for all purchases in the period. B_p sends it to the credit card account (B_c, C) using protocol bank-to-bank $[(B_p, P), (B_c, C), \$2]$, where $\$2$ is the bill amount. B_c bills C through the conventional billing procedure. When C pays the bill, B_c places additional credits in (B_p, P) .

Remarks:

- 1) After the protocol is executed, the store S knows no more than before, bank B_c only knows the total amount $\$2$ of purchases in the billing period, bank B_p knows the available credit $\$1$ at the beginning of each period as well as the amounts $\$3$ of individual purchases, the store's bank B_s only knows that the purchase amount has been deposited into the store's account, and the communication exchange cx only knows that funds have been transferred among banks.
- 2) The customer can generate a personalized purchase record at the time of purchase, encrypt it with a secret key known *only* to her, and send it to her anonymous account (B_p, P) along with the funds transfer request.

At the end of the billing period, these records will be forwarded to (B_c, C) and then to C along with the bill. Other features of conventional credit cards, such as cancelling lost cards or abnormal spending alert, can also be implemented using the building blocks of Section II.

- 3) The communication exchange logs all messages to create an audit trail in case a dispute arises or for use when an item is returned.
- 4) If P always deposits cash into the anonymous account (B_p, P) instead of having (B_c, C) transferring funds into it, the customer's identity need not be known to the system. The system then provides complete anonymity.

IV. COLLUSION ANALYSIS

In this section, we analyze the collusion property of the protocol in Sections II and III. For completeness, we first briefly review our model for collusion analysis [8], [10], [11]. Then we apply it to the credit card protocol.

Cryptographic protocols in the literature are typically described in a style similar to that followed in Section II. It spells out the steps involved in a successful transaction but only hints at how errors will be handled. For a more complete description of the anonymous credit card protocol, with minor modifications, see [9]. For ease of exposition, we will only deal with the subpart of the protocol as described in Sections II and III. It is indeed only one of many possible realizations of the complete protocol.

A. Collusion Model

Collusion is carried out in an *environment* described by the five-tuple (U_p, D, K, U_c, L) , where:

- 1) U_p is a finite set of protocol users;
- 2) D is a finite set of data;
- 3) K is a finite set of cryptographic keys, including the identity key ϵ ;
- 4) $U_c \subseteq U_p$ is a set of colluders;
- 5) $L \subseteq U_p \cup D \cup K$ is a set of information that determines whether two users can collude; see Condition (3) below.

Define the *information set* as the set of every possible encryption and clear text combination of every piece of information in the system

$$I := K^*(U_p \cup D \cup K)$$

where K^* denotes K 's Kleene closure. For example, if $d \in U_p \cup D \cup K$ and $k_i \in K$, then $d, k_1(d), k_2k_1(d), k_1k_2k_1(d)$ are all in I . Here, a string $k_n \cdots k_1$ represents successive application of keys k_1, \dots, k_n in order. We assume that encrypting the concatenation of two items is equivalent to two separate encrypted items as far as information is concerned, though they almost always differ in their form. Hence, e.g., $k_2(d_2, k_1(d_1))$ and $\{k_2(d_2), k_2k_1(d_1)\}$ are used interchangeably.

For any key k , k^{-1} denotes its inverse with the cancellation rule $k^{-1}k = k k^{-1} = \epsilon$, the identity key. For example, The keys k and k^{-1} are identical in secret-key cryptosystems, but not in public-key cryptosystems. Decryption is a function $\Delta : 2^I \rightarrow 2^I$ that is defined by the cancellation rule. For

instance, $\Delta(\{k(d)\} \cup \{k^{-1}\}) = \{d, k^{-1}\}$. $\Delta(A)$ represents the decryption of a set $A \subseteq I$ of information by the keys included in A such that if $k_n \cdots k_1(d) \in \Delta(A)$ then $k_n^{-1} \notin \Delta(A)$.

The *knowledge set* is the combination of the messages and information

$$W := 2^N \times 2^I$$

where N is the set of unique message identifiers and I is the information set. An element $w = (w.N, w.I)$ of W represents a user's knowledge. It has two components: the first component $w.N \subseteq N$ represents all the messages the user has seen, and the second component $w.I \subseteq I$ represents all the information the user knows.

As colluders in U_c collude by exchanging messages, their knowledge is modified. This evolution is modeled by a transition system $\Theta = (W^{|U_c|}, \Sigma, \delta)$. Here, a state $w = (w_u, u \in U_c)$ in $W^{|U_c|}$ is the knowledge of all colluders. An event $\sigma = (s, r)$ in $\Sigma := U_c \times U_c$ describes the transfer of the sender's complete knowledge w_s to the receiver r to attempt to extract the hidden information at the receiver. The transition function δ describes the transformation of colluders' knowledge as a result of the message exchange, as elaborated next.

It is not always possible for two users to collude. In order for s to send a message to r , s must know r , as in the protocol. Sometimes, they also must share a common, unique piece of information pertaining to the protocol run in question. To motivate this requirement, consider as an example an intermediary cx that forwards a piece of data to its recipient r in order to hide the identity of its sender s from r

- 1) $s \rightarrow cx: k_{cx}(r, k_r(d))$;
- 2) $cx \rightarrow r: k_r(d)$.

In the above, s encrypts the (encrypted) data $k_r(d)$ and the recipient's identity r with a key k_{cx} that can only be decrypted by the intermediary cx and sends them to cx (message 1). The intermediary cx then forwards the encrypted data to r (message 2), thus hiding the identity of the sender s from r . Variants of this simple protocol have been the building blocks of large cryptographic protocols to provide privacy in broadband networks [17], [18], in credit card transactions [9], and in mobile networks [6], where traffic volumes are high. After the above steps are carried out, cx knows $w_{cx} := (\{\text{message 1, message 2}\}, \{s, cx, r, k, k_{cx}^{-1}, k_r(d)\})$ and r knows $w_r := (\{\text{message 2}\}, \{cx, r, k_r, k_r^{-1}, d\})$. For r to discover s , r must learn the information in w_{cx} . In a large system, however, cx may have forwarded a large number of messages to the same recipient r in a short period of time and they have collected a large number of w_{cx} and w_r , corresponding to different protocol runs. Moreover, the larger protocol of which the above is only a part can be implemented on a datagram network so that messages from different protocol runs may be interleaved at cx . Hence to combine the information in w_{cx} and w_r of the *same* protocol run, cx and r must share a unique piece of information pertaining to that protocol run. The unique message that is exchanged between cx and r can be used to pair up w_{cx} and w_r that belong to the same protocol run.

Alternatively, two users can collude if they share a unique piece of data in L , e.g., two banks may have the unique social security number of a customer and, hence, can combine their knowledge about the customer.

Formally, for each state $w = (w_u, u \in U_c)$ and event $\sigma = (s, r)$, the transition $\delta(w, \sigma)$ is defined if and only if $r \in w_s.I$ and *at least* one of the following conditions is satisfied:

$$w_s.N \cap w_r.N \neq \phi \tag{2}$$

$$w_s.I \cap w_r.I \cap L \neq \phi. \tag{3}$$

Note that if $L = U_c$, then since $u \in w_u$ for all $u, r \in w_s.I$ implies Condition (3). Hence, the collusion prerequisites reduce to the special case in which collusion is allowed as long as the sender s knows the receiver r .

When the current state is $w = (w_u, u \in U_c)$ and a transition $\sigma = (s, r)$ is made, the receiver's knowledge is expanded to include that of the sender. The next state, $w' := \delta(w, \sigma)$, is defined by

$$w'_y = w_y, \text{ if } y \neq r$$

$$w'_r.N = w_r.N \cup w_s.N, \quad w'_r.I = \Delta(w_r.I \cup w_s.I).$$

We call an event $\sigma = (s, r)$ in Σ *enabled in state* w if the transition $\delta(w, \sigma)$ is defined. A path is *valid* if every transition on the path is enabled in the state from which the transition is made.

We summarize our model in the following definition. The transition system Θ describes all the possible sequences of message exchanges among the colluders and how their knowledge evolves as collusion proceeds.

Definition 1: Given an environment (U_p, D, K, U_c, L) , a collusion system is the (unique) transition system $\Theta = (W^{|U_c|}, \Sigma, \delta)$ defined above.

The collusion problem is to determine if a subset of users can combine their information, by passing messages, and extract the hidden information after or during a protocol's execution. Suppose we have a collusion system $\Theta = (W^{|U_c|}, \Sigma, \delta)$.

Collusion Problem: Given an initial state $w(0) \in W^{|U_c|}$ and a target set of unencrypted information $T \subseteq U_p \cup D \cup K$, does there exist a valid path ρ in Θ that starts in $w(0)$ and terminates in a state $w(\rho)$ in which a colluder $c \in U_c$ knows T , i.e., $w_c(\rho).I \supseteq T$?

We call the valid path ρ in the definition of the collusion problem a *collusion path*. Though the collusion problem is simply a reachability analysis on the state machine Θ , given $w(0)$, the reachable set contains up to $2^{|U_c|(|U_c|-1)}$ states. It is, hence, impractical to do an exhaustive search. In [11], we provide an algorithm that, for the special case where $L = \phi$, determines whether a collusion path exists, and constructs one when it does, from just the initial state $w(0)$. In [10], we extend the algorithm to solve the general case where L can be arbitrary. These results, and a negative result on least cost collusion paths, are proved in [8].

For the anonymous credit card protocol, we are interested to see whether various parties can collude to discover both the identity of the customer and her purchase. For this protocol, we can assume $L = \phi$, i.e., two users can collude only if

they share a unique message that was exchanged during the protocol run [Condition (2)].²

The solution for this special case is completely characterized by the following theorem from [11]. Define $F = (U_c, E(w(0)))$ as a undirected graph, depending on the initial state $w(0)$, that describes all the events that are initially enabled by Condition (2). F contains all colluders as its nodes. There is an edge (u, v) in $E(w(0))$ if and only if u and v have exchanged a message in the protocol run.

Theorem 1: As per [11], Theorem 1 follows.

- 1) The collusion problem has a solution if and only if there is a connected component $F' = (V, E)$ of the undirected graph F such that $\Delta(\cup_{u \in V} w_u(0).I) \supseteq T$.
- 2) The collusion problem has a solution if and only if there is a collusion path with the simple structure

$$\rho = (u_0, u_1)(u_1, u_2) \cdots (u_{n-1}, u_n)$$

where u_i are all distinct.

B. Anonymous Credit Card Protocol

We have described the intermediary cx as one logical entity. It can be implemented as one or multiple physical entities. A different program instance will be invoked at the intermediary cx to relay each message. Regardless of whether cx is one centralized entity or implemented as geographically distributed entities, these program instances cannot share information that pertains to a single credit card transaction. This is because, by assumption, the system processes a large number of transactions and cx has no way to tell which program instances correspond to different steps of the same transaction. Hence, we model different invocations of cx separately as cx_1, \dots, cx_4 .

We will model the protocol that consists of a sub-protocol to place credit of amount $\$1$ into (B_p, P) , one to bill the credit card account (B_c, C) by an amount $\$2$, and one to make a purchase of amount $\$3$. Specifically, the following executions are modeled: bank-to-bank $[(B_c, C), (B_p, P), \$1]$ to place credit that involves cx_1 (messages 1 and 2), bank-to-bank $[(B_p, P), (B_c, C), \$2]$ to bill the customer that involves cx_2 (messages 3 and 4), customer-to-bank $[S, B_p, P, m]$ to request funds to be transferred during a purchase that involves cx_3 (messages 5 and 6), bank-to-bank $[(B_p, P), (B_s, S), \$3, K_{B_s, S}(n, \$3)]$ that transfers funds that involves cx_4 (messages 7 and 8), and finally, the acknowledgment from B_s to S (message 9).

The environment consists of (U_p, D, K, U_c, ϕ) where

$$\begin{aligned} U_p &= \{C, P, S, B_c, B_p, B_s, cx_1, cx_2, cx_3, cx_4\} \\ D &= \{g, n, \$1, \$2, \$3\} \\ K &= \{K_{B_c, B_c}, K_{B_p, B_p}, K_{B_s, B_s}, K_{cx, cx}, K_{B_c, cx}, \\ &\quad K_{B_p, cx}, K_{B_s, cx}, K_{S, cx}, K_{B_p, P}, K_{B_s, S}\} \end{aligned}$$

where g are the goods purchased by C . Suppose we want to determine whether it is possible after the protocol is

²Alternatively, we can assume $L = \{n\}$, i.e., two users can collude if both know the transaction number for the purchase. Then the algorithm in [8] and [10] can be applied, but the same conclusion will be reached. We assume $L = \phi$ here for simplicity.

TABLE I
INITIAL STATE OF COLLUSION SYSTEM Θ

User	Msg	Info before protocol starts	Add. info after protocol completes
S	5, 9	S, B_s, cx_3 $g, n, \$3$ $K_{S, cx}, K_{B_s, S}$ $K_{cx, cx}(B_p)$ $K_{cx, cx}K_{B_p, B_p}(P)$ $K_{cx, cx}(B_s)$ $K_{cx, cx}K_{B_s, B_s}(S)$	
B_c	1, 4	C, B_c, cx_1 $\$1$ $K_{B_c, B_c}, K_{B_c, cx}$ $K_{cx, cx}(B_p)$ $K_{cx, cx}K_{B_p, B_p}(P)$	cx_2 $\$2$
B_p	2, 3 6, 7	P, B_p, cx_2, cx_4 $\$2$ $K_{B_p, B_p}, K_{B_p, cx}$ $K_{B_p, P}$ $K_{cx, cx}(B_c)$ $K_{cx, cx}K_{B_c, B_c}(C)$	cx_1, cx_3 $\$1, \3 $K_{cx, cx}(B_s)$ $K_{cx, cx}K_{B_s, B_s}(S)$ $K_{B_s, S}(n, \$3)$
B_s	8, 9	S, B_s $K_{B_s, B_s}, K_{B_s, cx}$ $K_{B_s, S}$	cx_4 $n, \$3$
cx_1	1, 2	cx_1 $K_{cx, cx}, K_{B_c, cx}$ $K_{B_p, cx}, K_{B_s, cx}$ $K_{S, cx}$	B_c, B_p $\$1$ $K_{B_p, B_p}(P)$
cx_2	3, 4	cx_2 $K_{cx, cx}, K_{B_c, cx}$ $K_{B_p, cx}, K_{B_s, cx}$ $K_{S, cx}$	B_c, B_p $\$2$ $K_{B_c, B_c}(C)$
cx_3	5, 6	cx_3 $K_{cx, cx}, K_{B_c, cx}$ $K_{B_p, cx}, K_{B_s, cx}$ $K_{S, cx}$	S, B_p $K_{B_p, B_p}(P)$ $K_{B_p, P}(\$3)$ $K_{B_p, P}K_{cx, cx}(B_s)$ $K_{B_p, P}K_{cx, cx}K_{B_s, B_s}(S)$ $K_{B_p, P}K_{B_s, S}(n, \$3)$
cx_4	7, 8	cx_4 $K_{cx, cx}, K_{B_c, cx}$ $K_{B_p, cx}, K_{B_s, cx}$ $K_{S, cx}$	B_p, B_s $\$3$ $K_{B_s, B_s}(S)$ $K_{B_s, S}(n, \$3)$

executed for any users except C and P to discover both the identity C of the customer and her purchase g . Then $U_c = \{S, B_c, B_p, B_s, cx_1, cx_2, cx_3, cx_4\}$. The target information set is $T = \{C, g\}$.

The environment uniquely defines a collusion system Θ . The initial state of the transition system corresponds to the knowledge of all colluders after the protocol has terminated. It is given in Table I.

We seek a collusion path in Θ that starts from the state given in Table I and leads to a state in which at least one user in U_c knows T .

Note that solving the collusion problem by exhaustive search on the transition system Θ will not be efficient as the search space has $2^{|U_c|^2 - |U_c|} \sim 10^{17}$ states.

According to the algorithm given in [11], we will first construct the undirected graph $F = (U_c, E(w(0)))$ defined in the last section. Then we will run the algorithm that performs a

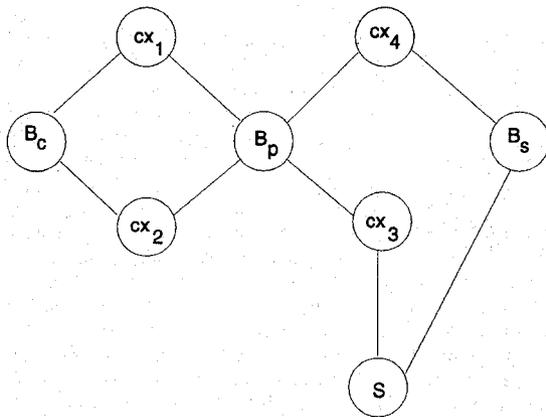


Fig. 1. Graph F .

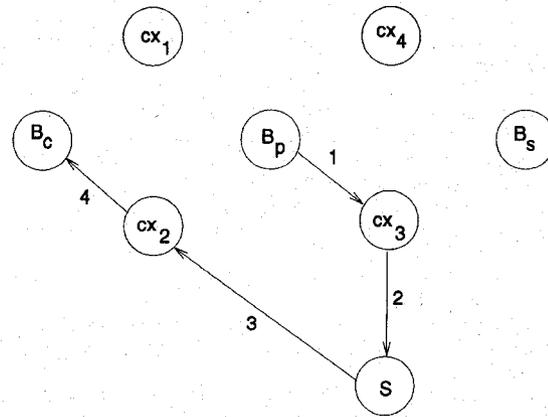


Fig. 3. A minimum collusion path.

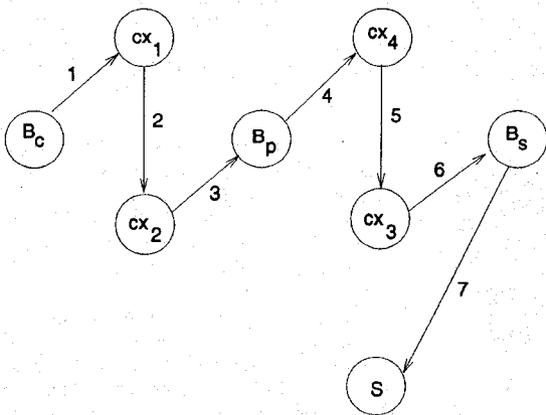


Fig. 2. A collusion path.

breadth-first search on each connected component of F while attempting to construct a collusion path of the form given in Theorem 1. The algorithm will return a collusion path if and only if the collusion problem has a solution; otherwise it returns NIL.

From Table I, F is as shown in Fig. 1. Two nodes are connected if and only if one has received a message from the other in the protocol phase. F is a connected graph; moreover, since $C \in w_{B_c}$ and $g \in w_S$, $\Delta(\cup_{u \in F} w_u(0).I) \supseteq T = \{C, g\}$. Hence, according to Theorem 1, the collusion problem has a solution.

Application of the algorithm in [11] yields a collusion path defined by the graph shown in Fig. 2. The labels on the edges specify the order in which these transitions are carried out. Hence, the collusion path in Θ that corresponds to the given graph is

$$(B_c, cx_1)(cx_1, cx_2)(cx_2, B_p)(B_p, cx_4) \\ (cx_4, cx_3)(cx_3, B_s)(B_s, S).$$

In exploring a connected component of F , the collusion path produced visits every node in the connected component. It is minimal (i.e., if any edge is removed, it ceases to be a collusion path), but not minimum. Fig. 3 shows a collusion graph that involves the minimum number of colluders. It is much shorter than the collusion path produced by the algorithm. It can

be shown however that finding a shortest collusion path is NP-complete [8].

If we restrict the set U_c of colluders to, say, $\{B_c, B_p, B_s, cx_3, cx_4\}$, then no collusion path exists according to the theorem. The same is true if B_p is excluded from the set of possible colluders even though B_p knows neither C nor g . This is because excluding B_p breaks F into two separate connected components, neither knows the entire T .

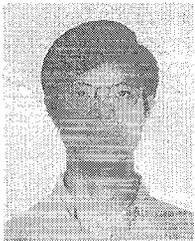
V. CONCLUSION

We have presented a way to implement anonymous credit cards with simple building blocks that use networks to separate information in order to protect personal privacy. We have shown how to formally analyze the collusion property of the protocol. The analysis shows, for example, that in order to associate the purchase with the identity of the customer, the anonymous bank and both intermediaries that are involved in the purchase and in billing must collude even though none of them knows the customer's identity or purchase.

REFERENCES

- [1] J. Brassil, S. Low, N. Maxemchuk, and L. O'Gorman, "Electronic marking and identification techniques to discourage document copying," in *Proc. INFOCOM'94*, 1994, pp. 1278-1287.
- [2] D. L. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *Advances in Cryptology-CRYPTO '88*. New York: Springer-Verlag, 1988, pp. 319-327.
- [3] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84-88, Feb. 1981.
- [4] ———, "Security without identification: transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030-1044, Oct. 1985.
- [5] D. Semyon, "SNPP: A simple network payment protocol," in *Proc. Computer Security Applicat. Conf.*, San Antonio, TX, Nov. 1992.
- [6] F. Hannes, A. Jerichow, and A. Pfitzmann, "Mixes in mobile communication systems: Location management with privacy," in *Proc. Workshop Informat. Hiding*, May 1996.
- [7] S. H. Low, A. M. Lapone, and N. F. Maxemchuk, "Document identification to discourage illicit copying," in *Proc. GLOBECOM'95*, Nov. 1995.
- [8] S. H. Low and N. F. Maxemchuk, "Collusion in cryptographic protocols," Univ. of Melbourne, Tech. Rep., 1996.
- [9] S. H. Low, N. F. Maxemchuk, and S. Paul, "Anonymous credit cards," in *Proc. 2nd. ACM Conf. Computer Commun. Security*, Nov. 2-4, 1994.
- [10] S. H. Low and N. F. Maxemchuk, "Collusion analysis of cryptographic protocols," in *Proc. GLOBECOM'96*, London, U.K., Nov. 1996.

- [11] ———, "Modeling cryptographic protocols and their collusion analysis," in *Proc. 1st Int. Workshop Informat. Hiding*, Univ. of Cambridge, U.K., May 1996.
- [12] ———, "The use of communications networks to increase personal privacy," in *Proc. INFOCOM'95*, 1995, pp. 504-512.
- [13] G. Medvinsky and B. C. Neuman, "Netcash: a design for practical electronic currency on the internet," in *Proc. 1st ACM Conf. Computer Commun. Security*, Nov. 1993.
- [14] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Commun. ACM*, vol. 21, no. 12, pp. 993-999, Dec. 1978.
- [15] ———, "Authentication revisited," *ACM Oper. Syst. Rev.*, vol. 21, no. 1, p. 7, Jan. 1987.
- [16] T. Okamoto and K. Ohta, "Universal electronic cash," in *Advances in Cryptology—CRYPTO '91*. New York: Springer-Verlag, 1992, pp. 324-337.
- [17] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "ISDN-MIXes—Untraceable communications with very small bandwidth overhead," in *Proc. IFIP/Sec'91*, 1991, pp. 245-258.
- [18] A. Pfitzmann and M. Waidner, "Networks without user observability," *Comput. Security*, vol. 6, no. 2, pp. 158-166, 1987.



Steven H. Low (S'88-M'92) received the B.S. degree from Cornell University, Ithaca, NY, in 1987, and the M.S. and Ph.D. degrees from University of California at Berkeley, in 1989 and 1992, respectively, all in electrical engineering.

He was a Consultant to NEC, San Jose, CA, in 1991, and was with AT&T Bell Laboratories, Murray Hill, NJ, from 1992 to 1996. He has held visiting positions at Rutgers University, NJ, and the University of Science and Technology, Hong Kong. He joined the University of Melbourne, Australia,

as a Senior Lecturer, in May 1996. His areas of research are in the design, control, and optimization of communication networks and protocols. He holds one patent.

Dr. Low received the 1996 R&D 100 Award for his work on document marking.

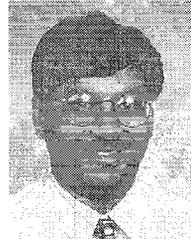


Nicholas F. Maxemchuk (F'89) received the B.S.E.E. degree from the City College of New York, and the M.S.E.E. and Ph.D. degrees from the University of Pennsylvania.

He is currently the Head of the Distributed Systems Research Department at AT&T Bell Laboratories, Murray Hill, NJ, where he has been since 1976. Prior to joining Bell Laboratories, he was at the RCA David Sarnoff Research Center, Princeton, NJ, for eight years. He has been on the adjunct faculties of Columbia University and the

University of Pennsylvania. He has been an Advisor on data networking to the United Nations, the National Science Foundation, the Rome Air Development Center, Canada's Information Technology Research Center, the Telecommunications Research Institute of Ontario, and other organizations. He has served as the Editor for Data Communications for the IEEE TRANSACTIONS ON COMMUNICATIONS, as a Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, and is currently on JSAC's Editorial Board, and the Chairman of the Steering Committee of the IEEE/ACM TRANSACTIONS ON NETWORKING.

Dr. Maxemchuk was awarded the RCA Laboratories Outstanding Achievement Award in 1970, the Bell Laboratories Distinguished Technical Staff Award in 1984, the IEEE's 1985 and 1987 Leonard G. Abraham Prize Paper Award, was made a fellow of the IEEE in 1989, and received the 1996 R&D 100 Award for his work on document marking.



Sanjoy Paul (M'92) received the B. Tech (Hons.) degree in electronics and electrical communications engineering from the Indian Institute of Technology, Kharagpur, India, in 1985, and the M.S. and Ph.D. degrees from the University of Maryland, College Park, both in electrical engineering, in 1988 and 1992, respectively.

He worked as an Assistant Systems Engineer in India for CMC Limited from 1985 to 1986. He joined the Distributed Systems Research Department at AT&T Bell Laboratories, Murray Hill, NJ, in 1992. Currently, he is a Research Member of the Technical Staff in the Wireless Networking Research Department at Bell Laboratories, Holmdel, NJ. He holds several U.S. patents, has published several conference and journal papers, and has been on the program committees of several IEEE conferences. His research interests include multicasting, transport, network, and link-layer protocol issues, formal methods, security, and mobile networking.