Collusion in a Multi-party Communication Protocol for Anonymous Credit Cards *

Steven H. Low
Nicholas F. Maxemchuk
Sanjoy Paul

AT&T Bell Laboratories
Murray Hill, NJ 07974

January 13, 1994

Abstract

We proposed in [8] a novel scheme to implement an *anonymous credit card* that protects privacy while providing the security, record-keeping and charging mechanism of conventional credit cards. The key idea is to use cryptographic techniques to allow two parties to communicate without knowing each other. In this paper, we present a formal method to study collusion in the multi-party communication protocol in [8]. Application of the method to our protocol leads to a simplified implementation of anonymous credit cards that is equally secure.

1 Introduction

As computing and data storage become less expensive, it is easier to assemble more information on individuals and to invade their privacy. With the increasing use of credit and bank cards, a profile of individuals' spending habits can be compiled by associating purchases with a customer. At some point, the ability to collect information compromises an individuals' right and willingness to deviate from the norm. Cash solves the privacy problem, since a bank note contains no attribute pertaining to its owner. This anonymity, however, also creates a security problem that deters many people from carrying a large amount of cash. In this paper, a mechanism is described that protects privacy and retains the desirable characteristics of credit cards, such as security against loss or theft, purchase statement, limit on loss to the credit line, and detection of abnormal spending patterns, etc.

*Submitted to IEEE/ACM Transactions on Networking

Reference [3] suggests a very interesting approach that uses a cryptographic protocol to create electronic *cash*, thus making it impossible in an automated transaction to associate a purchase with the customer. The scheme, however, may be insecure against fraud; see Appendix. In addition, 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. Reference [6] designs a simple protocol to perform payment transactions between mutually distrustful parties over an insecure network. It does not, as we do, focus on preventing all parties involved in the transaction from compiling purchase habits.

We propose an alternative in [8] that creates an anonymous *credit card*, in which a customer transfers funds from his/her bank directly to the store's bank during a purchase. Our protocol uses cryptographic techniques to perform the transfer without either bank knowing the other. Since the bits representing money are never in the possession of individuals, our scheme avoids the problem of double-spending. In this paper, we present a method to study collusion in the multi-party communication protocol in [8].

In §2, we present the protocol that implements an anonymous credit card. The basic idea is explained in §2.1; the cryptographic tools, message formats and notations are given in §2.2; a brief description of the protocol is in §2.3.

The protocol guarantees that no party involved in a transaction by itself can associate the customer's identity with the purchase. The need to extend credit, however, makes it necessary for the card-issuing bank to know the identity, and leads to the possibility of collusion against the customer. This is studied in §3. In §3.1 we state our assumptions; in §3.2 we propose a formal method to study collusion in a general multi-party communication protocol; in §3.3, we apply the method to the protocol in §2 to show that a customer cannot be compromised unless all parties collude against the customer. The analysis leads to a simplified implementation of anonymous credit cards that is 'equally secure', which is explained in §4.

Our protocol makes essential use of two public-key cryptographic tools [5, 12] (see also [13]) : public-key encryption $E_A(msg)$ and signature $Sgn_A(msg)$. In such a system, each party A that wishes to communicate has a key pair. One of the pair, called the public key, is made public, and the other, called the secret key, is known only to A. To ensure secrecy of communication one sends $E_A(msg)$ to A, which is the message msg encrypted with A's *public* key so that only A can decrypt it (with its secret key). To sign a message msg A generates $Sgn_A(msg)$, which is a copy of msg followed by the encryption of msg with A's secret key, so that everyone with A's public key can verify the signature but only A could have produced it.

2 Protocol

2.1 Basic idea

To protect privacy, 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 or credits that have been deposited in the account. Since it does not extend credit, bank B_p knows the customer only by the pseudonym P. In the following, we will use freely P and C to refer to the customer.

At the beginning of a billing period, bank B_c places credits in P's account (B_p, P) at B_p. 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. B_s then notifies store S, which releases the merchandise to P. At the end of a billing period, B_p presents bank B_c with a bill, which B_c pays. B_c presents the bill to C. When C pays the bill, B_c places additional credits in (B_p, P).

The key idea is to use cryptographic techniques to enable 1) banks B_c and B_p to communicate without knowing each other or the account for which the transaction is destined; and 2) bank B_p to transfer funds to B_s without knowing each other. This is done with the aid of an intermediary for all communications, which we call the communication exchange cx. To achieve 1, C stores the encrypted address $E_{cx}(B_p, E_{B_p}(P))$ at bank B_c. Here, $E_{B_p}(P)$ is the account owner P encrypted with bank B_p's public key, and $E_{cx}(B_p, \cdot\cdot)$ is the destination bank B_p (and account owner) encrypted with cx's public key. To place credits in (B_p, P), B_c sends a message together with the encrypted destination $E_{cx}(B_p, E_{B_p}(P))$ to cx; cx determines, and forwards the message to, the destination bank B_p; only B_p can determine the destination account. Note that P is hidden from cx, and both B_p and P are hidden from B_c. A similar technique is used for sending billing statement from B_p to B_c without one knowing the other. To achieve 2, B_p is given by the customer P at the store the encrypted address $E_{cx}(B_s, E_{B_s}(S))$, and uses it to transfer funds to (B_s, S) in a similar way, as detailed in §2.3.

The communication exchange cx keeps an account for every bank. A fund transfer from a bank to another bank is recorded in their accounts at cx. This is possible since cx knows the source and destination banks for all fund transfers. Periodically, 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 by cx to provide an audit trail if a dispute arises.

Since all fund transfers take place electronically between banks, the bits representing money are never in the possession of individuals, who may otherwise duplicate and spend these bits over and over again.

2.2 Notations

The protocol is executed by the players C, B_c, P, B_p, S, B_s, and cx. It uses two public-key cryptographic tools — public-key encryption $E_A(msg)$ and signature $Sgn_A(msg)$ — explained in §1. The message msg is always appended with a random number before *every* encryption; e.g. $E_{cx}(B_p, E_{B_P}(P))$ contains two random numbers, one for E_{cx} and the other for E_{B_p}. This prevents a player that has all the information in the message but not the message itself from computing the encrypted version $E_A(msg)$; see footnote in §2.3.

The communication from a source src to a destination dst is accomplished by a message pair $m_1(src, dst)$ and $m_2(src, dst)$. A source src, or a destination dst, is one of C, B_c, P, B_p, S, and B_s. Four types of message formats are summarized in Table 1. $m_1(src, dst)$ is sent by src to cx and is formatted in $F(msg, dst)$ or $F_{src}(msg, dst)$; $m_2(src, dst)$

Notation	*Definition*	*Explanation*
$F(msg, dst)$	$E_{cx}(msg, dst)$	Message msg to be forwarded to destination dst by cx, encrypted with cx's public key. dst may be encrypted as $D^e(B_x, X)$ defined below.
$F_{src}(msg, dst)$	$E_{cx}[src, Sgn_{src}(msg, t(src, dst)), dst]$	Message msg timed-stamped and signed by source src to be forwarded to dst by cx, encrypted with cx's public key. The time-stamp $t(src, dst)$ increases for each successive message between src and dst and prevents replay attacks.
$X(msg, dst)$	$E_{dst}(msg)$	Message msg sent by cx to dst, encrypted with dst's public key.
$X_{src}(msg, dst)$	$E_{dst}[src, Sgn_{src}(msg, t(src, dst))]$	Message msg time-stamped and signed by source src sent by cx to dst, encrypted with dst's public key.
$D^e(B_x, X)$	$E_{cx}(B_x, E_{B_x}(X))$	Encrypted destination for account (B_x, X), X=C, P or S. By sending $D^e(B_x, X)$ to cx, a source can send a message to the account without knowing B_x or X. cx decrypts $D^e(B_x, X)$ to determine B_x, and forwards the message to B_x without knowing X. B_x decrypts $E_{B_x}(X)$ to determine account owner X.

Table 1: Message formats and notations

is sent by cx to dst and is formatted in $X(msg, dst)$ or $X_{src}(msg, dst)$.

Also defined in the table is the encrypted address for anonymous communications. C stores $D^e(B_p, P)$ at B_c to allow B_c to place credits into account (B_p, P) without knowing

B_p or P, and stores $D^e(B_c, C)$ at B_p to allow B_p to send bills for account (B_c, C) without knowing B_c or C, as explained in §2.1. During a purchase, P provides B_p with $D^e(B_s, S)$ to allow B_p to transfer funds into store's account without knowing B_s or S.

Finally, $E_p(msg)$ in the protocol is a message m encrypted with P's *public* key. It is used by P to store information at B_p that only P can decrypt. For additional security, B_p may ask the customer questions and compared the answers received. For this, P stores a set of questions Q_i and answers A_i at B_p as $\{E_p(Q_i), E_p(A_i)\}$. The questions are personal and known only to P. To authenticate P, B_p picks at random an $E_p(Q_i)$ and sends it to P at the store. P's anonymous credit card decrypts the question and presents Q_i to P. P answers it, the card encrypts it and sends $E_p(A_i)$ to B_P for comparison with the stored encrypted answer. Depending on the amount of purchase, the bank may ask a different number of questions.

2.3 Protocol specification

In this subsection, we give a brief description of our protocol; for a complete specification, see [8].

The protocol consists of three distinct subprotocols. The first subprotocol is used when B_c places credits into P's account at B_p. The second part is used during a purchase in which the customer requests B_p to transfer funds to (B_s, S). The third part is used when B_p sends B_c a billing statement for the customer's purchases. The first and third subprotocols are typically executed once every billing period, while the second subprotocol is executed during every purchase.

As noted earlier, all messages are logged by cx to provide an audit trail when dispute arises.

Subprotocol 1: Extension of credit

1. B_c sends cx the signed message

   $$m_1(B_c, B_p) = F_{B_c}(\delta_{CR}, D^e(B_p, P))$$

   to place a credit δ_{CR} into (B_p, P).

2. B_p receives from cx

   $$m_2(B_c, B_p) = X_{cx}[(\delta_{CR}, E_{B_p}(P)), B_p]$$

   signed and guaranteed by cx.

3. B_p increments P's credit by δ_{CR}.

The second subprotocol consists of two phases. In the first phase, B_p authenticates P at the store after receiving a transaction request. In the second phase, B_p transfers funds to (B_s, S).

Subprotocol 2: During purchase

Phase 1: Authentication of P

1. P at store S sends to cx

$$m_1(P, B_p) = F[X_P(R, B_p), B_p]$$

to request a transaction with B_p. Here,

$$R = [\delta, \mathit{PIN}, D^e(B_s, S), E_{cx}(S), E_p(\mathit{record})]$$

δ is the amount of credit that P requests. *PIN* is the personal identification number used to identify P to bank B_p. $D^e(B_s, S)$ is (the encrypted) S's account at B_s to be used by B_p for fund transfer. It is computed by P's anonymous credit card from (B_s, S).[1] $E_{cx}(S)$ is the encrypted address of the automatic cash register at S through which P and B_p communicate. $E_p(\mathit{record})$ is a personal record P associates with the purchase to trace questionable purchases. It can be decrypted only by P. It is sent to B_c by B_p with the bill at the end of a billing period (see below).

2. B_p receives the message from cx

$$m_2(P, B_p) = X_P(R, B_p)$$

signed by P requesting a fund transfer.

Depending on the amount of the transaction, B_p may accept a *PIN* or may make one or more challenges in a challenge-response phase described in steps 3-7.

3. B_p randomly selects and sends a question Q_i to P in

$$m_1(B_p, P) = F[X_{B_p}(E_p(Q_i), P), E_{cx}(S)]$$

4. P receives the message

$$m_2(B_p, P) = X[X_{B_p}(E_p(Q_i), P), S]$$

Note that cx appends $X_{B_p}(E_p(Q_i), P)$ with a random number and encrypts it before forwarding, in order to break a link for collusion between B_p and S that eavesdrops. Otherwise, if cx simply forwards $X_{B_p}(E_p(Q_i), P))$, then this message will serve for B_p and S that eavesdrops as a link unique to the transaction and allows them to combine their information; see §3.

5. P sends the answer A_i to B_p in

$$m_1'(P, B_p) = F[X_P(E_p(A_i), B_p), B_p]$$

[1] Note that even if $B_p = B_s$, B_p cannot compare the received $D^e(B_s, S)$ with precomputed encrypted addresses of all its store customers in order to associate P with S. This is because a random number is appended before each encryption; see §2.2.

6. B_p receives the message

$$m_2'(P, B_p) = X_P(E_p(A_i), B_p)$$

7. If P's answer is correct and if the account (B_p, P) has enough credit, B_p starts the fund-transfer phase.

Phase 2: Fund transfer

8. B_p sends cx the signed message

$$m_1(B_p, B_s) = F_{B_p}[(1-\alpha)\delta, D^e(B_s, S)]$$

$\alpha\delta$ is the service charge to the store.

9. After verifying the banks B_c and B_s and the signature, cx transfers the amount $(1-\alpha)\delta$ from B_c's account to B_s's account at cx. It then sends B_s the message

$$m_2(B_p, B_s) = X_{cx}[((1-\alpha)\delta, E_{B_s}(S)), B_s]$$

signed and guaranteed by cx.

10. On receiving $m_2(B_p, B_s)$, B_s sends a signed confirmation of the transfer

$$m_1(B_s, S) = F[X_{B_s}(rcvd\ \delta, S), S]$$

to cx, which forwards it as

$$m_2(B_s, S) = X_{B_s}(rcvd\ \delta, S)$$

to S. S then releases the merchandise to P.

At the end of a billing period, suppose P has charged an amount Δ, and B_p has given the amount $(1-\alpha)\Delta$ to store's banks. B_p has B_c's guarantee to pay the debt, and it is B_c's responsibility to collect Δ from C. By negotiation, the banks have agreed that B_p deserves a fraction β of the excess funds $\alpha\Delta$ and B_c deserves $1-\beta$ of it. The following subprotocol is used to bill (B_c, C).

Subprotocol 3: Billing (B_c, C)

1. B_p sends cx the signed message

$$m_1(B_p, B_c) = F_{B_p}[(Rqst, E_p(record)), D^e(B_c, C)]$$

Here, $Rqst = (1-(1-\beta))\alpha\Delta$ is the amount B_p is requesting from B_c, and Δ is the amount B_c should bill C. $E_p(record)$ is the encrypted itemized record of purchases generated by C's anonymous credit card. It can only be decrypted by C.

2. After verifying B_p and B_c and the signature, cx transfers the amount $Rqst$ from B_c's account to B_p's account at cx. It then sends B_c the message

$$m_2(B_p, B_c) = X_{cx}[(Rqst, E_p(record), E_{B_c}(C)), B_c]$$

signed by cx.

B_c sends C a bill for Δ amount together with purchase record $E_p(record)$. If C challenges the bill, there is a complete audit trail of messages logged by cx. When C pays PAY amount, B_c extends PAY additional credit to (B_p, P) using subprotocol 1.

Other features of conventional credit cards can also be implemented. For instance, if the credit card is reported lost by C, B_c can use subprotocol 1 to reduce the credit in (B_p, P) to zero. The customer has the option of setting triggers on account (B_p, P), so that B_p can use subprotocol 3 to notify the customer, through B_c, when abnormal spending patterns are detected.

3 Collusion

In this section, we show how collusion among players B_c, B_p, B_s, S and cx can compromise the privacy of C. Privacy is compromised if a player (other than C) knows both the customer's identity C and the merchandise M, or both C and S. We assume the underlying cryptosystem is secure; in particular, we assume the underlying cryptographic algorithm is not compromised by the protocol, as exemplified in [10]. Privacy can be compromised only when the players share their information on the transaction. Relevant information is the customer's identity C, his/her pseudonym P, the personal identification number PIN, questions and answers $\{Q_i, A_i\}$, purchase records $record$, store S, merchandise M, and the banks B_c, B_p, B_s.

We give our assumptions in §3.1. Since the analysis may be applicable to similar protocols, we describe our method in general terms in §3.2. Finally, we apply the method to study the protocol in §2.

The proofs of Propositions 1 and 2 below are simple and collected in the Appendix.

3.1 Assumptions

Before any parties collude, B_c, B_p, B_s and S have no unique information, pertaining to C, in common. Since they have no way of knowing which pieces of information pertain to the same customer, if they try to collude, they cannot combine their information. The players B_c, B_p, B_s and S do have the messages that have been transmitted in common with cx. By colluding with cx, they can learn more about the party at the other end of the communications. For instance, when B_c places a credit into (B_p, P), a unique message was sent to cx. By colluding with cx, B_c can learn the B_p that message was forwarded to as well as the unique message that was sent for B_p.

We tacitly assume here that the amount of a transaction cannot be used as a piece of data unique to the transaction that allows two parties to combine their information. That is, we assume that collating δ_{CR} or *Rqst* does not help collusion between B_p and B_c, collating the price δ of the merchandise does not help collusion between B_p and S or B_s, and collating $rcvd\ \delta$ does not help collusion between B_s and S. This is justified if there are a large number of fund transfers in the system so that the dollar amounts will be the same in many transactions.

Under this assumption, no collusion among the banks and store can compromise the customer if cx is trusted, as [8, Section 4] shows. The random number included in every encryption ensures that the source and the destination of a message transfer cannot collude on the message content, without first colluding with cx. We hence focus on the situation after players have colluded with cx on messages that they possess.

Recall that information exchange is always accomplished by two messages, one from the source to cx and the other from cx to destination. When a player colludes with cx on the message it possesses, it learns the information contained in both messages (except the information that cannot be decrypted by the player or by cx). Hence, since we are interested in the situation *after* collusion with cx, we consider each message pair as a whole.

3.2 A method

Our analysis proceeds in three steps. First, a formal model for studying collusion is derived from the *step-by-step description* of the protocol. A model of the protocol is a triple of boolean matrices [7] (I_0, L, N) that describe the situation after the purchase but before colluding with cx. I_0 describes the initial information the players have; L describes the message pairs they possess; N describes the information contained in the message pairs. Second, simple operations (equations (1-2) below) are applied to these boolean matrices to derive information the players have after collusion with cx (matrix I_1), and to determine whether two players share a link unique to the transaction that allows them to combine their information (matrix U). Third, matrices I_1 and U define a collusion graph. Questions such as minimal set of players that need to collude in order to compromise C can be posed as familiar spanning tree problems on the graph.

We next describe our method in detail. Let *Player* be a set of players. *Msg* be a set of message pairs exchanged during the transaction. *Info* be a set of information contained in the message pairs. *Comp* be a subset of *Info*. C is said to be *compromised* if a player knows all the information in *Comp*. For our protocol in §2, $Player = \{B_c, B_p, B_s, S\}$,[2] $Info = Player \cup \{C, P, M, PIN, Q_i, A_i, record\}$. We may choose $Comp = \{C, S\}$, or $Comp = \{C, M\}$.

For the purpose of collusion analysis, the protocol can be formally modeled by three boolean matrices:

[2] Note that we do not include cx because we are primarily interested in the situation after collusion with cx.

1. I_0 is an $|Info| \times |Player|$ matrix that describes the information each player has *before* colluding with cx: $I_0(i,j) = 1$ if player j knows information i, and $I_0(i,j) = 0$ otherwise.

2. L is a $|Msg| \times |Player|$ matrix that describes the message pairs each player possesses after the transaction: $L(i,j) = 1$ if player j possesses message pair i, and $L(i,j) = 0$ otherwise. (We consider a player to possess a message pair if it possesses one of the pair.)

3. N is an $|Info| \times |Msg|$ matrix that describes the new information a message pair provides to its possessors if they collude with cx on that pair. For each message pair j and information i contained in j, $N(i,j) = 1$ if the pair (i,j) satisfies the following condition and $N(i,j) = 0$ otherwise:

   (a) cx can decrypt i (possibly with the help of *any single* player possessing j), or

   (b) all players possessing j already know or can decrypt i.

   Here, player j *possesses* message pair i if and only if $L(i,j) = 1$. Player j *already knows* information i if and only if $I_0(i,j) = 1$. See below for an example on deriving N from protocol specification.

Let I_1 be an $|Info| \times |Player|$ matrix that describes the information each player has *after* colluding with cx. Then, I_1 can be computed from I_0, L, and N by

$$I_1 \quad = \quad I_0 + N \cdot L \tag{1}$$

Here '$\cdot$' is the matrix multiplication for boolean matrices, which is the same as regular matrix multiplication except that '+' is interpreted as 'logical OR', i.e. $0 + 0 = 0$, $0 + 1 = 1 = 1 + 0$, $1 + 1 = 1$. For example,

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 + 1 \cdot 1 \\ 0 \cdot 1 + 1 \cdot 1 \end{bmatrix} = \begin{bmatrix} 1 + 1 \\ 0 + 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Two players i and j can share their information only if they possess a link unique to the transaction. They possess such a link if they possess the same message pair, even if one or both cannot decrypt information contained in the message pair.[3] Hence let U be a $|Player| \times |Player|$ matrix that indicates which players have common pieces of data, such that

$$U(i,j) \quad = \quad \begin{cases} 1 & \text{if for some row } k,\ L(k,i) = L(k,j) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Note that U is necessarily symmetric, i.e. $U(i,j) = U(j,i)$. We write $U(L)$ when we want to emphasize that U is a function of L.

[3]Even though B_c and B_p may use $D^e(B_p, P)$ stored at B_c or $D^e(B_c, C)$ stored at B_p as a unique link to combine C and P, $D^e(B_p, P)$ and $D^e(B_c, C)$ are also contained, respectively, in the messages $m_1(B_c, B_p)$ and $m_1(B_p, B_c)$. Hence, we may as well assume that only message pairs serve as unique links for the players.

Clearly, we may remove the ith rows of N and I_0 if *both* are zero vectors (and remove that information from the set *Info*). This is because information i is known to none of the players before the transaction and is not revealed in any message pair. Q_i and A_i in our protocol are such kind of information (see §3.3). We can further simplify N and L by applying the following proposition. Denote by $A(j)$ the jth column of matrix A, and by A^T its transpose. For two vectors x and y, we say $x \leq y$ if their components $x_i \leq y_i$ for all i.

Proposition 1 *If $N(i) \leq N(j)$ and $L^T(i) \leq L^T(j)$, then $N \cdot L = N' \cdot L'$ and $U(L) = U(L')$, where N' and $(L')^T$ are obtained from N and L^T, respectively, by removing their ith columns.*

The proposition says that we may use the reduced matrices N' and L' in (1) and (2) to compute I_1 and U that are used below to analyze collusion. It implies that the players do not have to log the removed message pair i in order to compromise C. This is because any information contained in message pair i is also contained in message pair j and any player possessing i also possesses j, so no player gains more information from collusion on i than on j. Furthermore, if i serves as a unique link for a pair of players, so does j for the same pair.

The matrices I_1 and U define a collusion graph $G = (V, E)$:

1. V has $|Player|$ vertices. Vertex j is labelled by player j together with all the information $i \notin Player$ that j knows ($I_1(i,j) = 1$ for $i \notin Player$).
2. There is a directed edge from vertex j to vertex $i \neq j$ if $I_1(i,j) = 1$, $i, j = 1, \ldots, |Player|$, with the interpretation that 'player j knows player i'.
3. A directed edge from vertex j to vertex $i \neq j$ is solid if and only if $U(i,j) = 1$.

Hence, if there is some vertex j whose label contains the information in *Comp*, then C is compromised if player j colludes with cx. A solid directed edge from j to i means that player j knows player i and they share a link unique to the transaction that allows them to combine their information. A broken directed edge means that j knows i but they cannot combine their information pertaining to the transaction.

If *Comp* does not contain any player in $Player$, e.g. $Comp = \{C, M\}$, then to link the information in *Comp*, it is necessary to find a directed tree in G with *all* solid edges such that the labels of all vertices in the tree combined contains *Comp*. A 'minimal' such tree gives the minimal set of players (all vertices in the tree) that need to collude to compromise C. Note that a directed path is a special case of a directed tree.

If *Comp* contains players, e.g. $Comp = \{C, S\}$, it is necessary to find a ('minimal') directed tree such that 1) the labels of all vertices in the tree combined contains *Comp*; and 2) all edges of the tree are solid, except possibly the in-edges to the players in *Comp* that are leaves (without children). The tree gives the set of players (all vertices in the tree except possibly those in *Comp*) that need to collude to compromise C.

The next proposition allows us to ignore player i in finding a minimal set of players to compromise C. This property can also be visualized in the graph (see Section §3.3).

Proposition 2 *If $I_1(i) \leq I_1(j)$ and $U(i) \leq U(j)$, then any information that can be obtained by player i through collusion can also be obtained by player j through collusion.*

3.3 Analysis of protocol

Recall that for our protocol, $Player = \{B_c, B_p, B_s, S\}$, and $Info = Player \cup \{C, P, M, PIN, Q_i, A_i, record\}$. $Msg = \{B_cB_p, B_pB_c, PB_p, B_pP, B_pB_s, B_sS\}$, where B_cB_p denotes the message pair $m_1(B_c, B_p)$ and $m_2(B_c, B_p)$, etc.[4] We may choose $Comp = \{C, S\}$, or $Comp = \{C, M\}$.

We first derive the formal model (I_0, L, N) from the step-by-step description of the protocol in §2.3. The information matrix before collusion with cx is

$$I_0 \;=\; \begin{array}{c} \\ B_c \\ B_p \\ B_s \\ S \\ C \\ P \\ M \\ PIN \\ Q_i \\ A_i \\ record \end{array} \begin{array}{c} \begin{array}{cccc} B_c & B_p & B_s & S \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

I_0 says that, before collusion with cx but *after* the transaction, B_c, B_p knows no other players, and B_s and S know each other. Furthermore, B_c knows C, B_p knows P and PIN, and S knows M. The message pairs possessed by the players are described by

$$L \;=\; \begin{array}{c} \\ B_cB_p \\ B_pB_c \\ PB_p \\ B_pP \\ B_pB_s \\ B_sS \end{array} \begin{array}{c} \begin{array}{cccc} B_c & B_p & B_s & S \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & e \\ 0 & 1 & 0 & e \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

In the last column, $e = 1$ if S eavesdrops.

[4] We have combined the messages pairs $m_i(P, B_p)$ and $m_i'(P, B_p)$, $i = 1, 2$, into PB_p.

The new information contained in the message pairs is described by

$$N = \begin{array}{c} \\ B_c \\ B_p \\ B_s \\ S \\ C \\ P \\ M \\ PIN \\ Q_i \\ A_i \\ record \end{array} \begin{array}{c} \begin{array}{cccccc} B_cB_p & B_pB_c & PB_p & B_pP & B_pB_s & B_sS \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-e & 1-e & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-e & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

As an example on how to construct N, consider column B_cB_p, corresponding to the message pair in subprotocol 1. From the specification, B_cB_p contains information B_c, B_p, and P. From row B_cB_p of L, players B_c and B_p possess the message pair B_cB_p. It can then be checked that information B_c, B_p satisfy condition (3a) or (3b) in §3.2, and hence the first two entries of column B_cB_p are '1'. Neither cx nor B_c can decrypt P from message pair B_cB_p, hence a '0' in the corresponding entry of column B_cB_p. As another example, consider column PB_p corresponding to messages pairs $m_i(P, B_p)$ and $m_i'(P, B_p)$, $i = 1, 2$, in subprotocol 1, phase 1. From the specification, the message pairs contain the information B_p, B_s, S, P, PIN, A_i and $record$. If S does not eavesdrop ($e = 0$), then from row PB_p of matrix L, only player B_p possesses message pair PB_p. It can then be checked that all information except A_i satisfies condition (3a) or (3b). If S eavesdrops ($e = 1$), then both B_p and S possess PB_p. Since S cannot decrypt P or PIN or $record$, neither (3a) nor (3b) are satisfied, hence '0's in the corresponding entries of column PB_p.

Remove the last three rows of N and I_0 since all are zero vectors. Invoke Proposition 1 to further reduce N and L: remove column 1 of N and row 1 of L (because $N(1) \leq N(2)$ and $L^T(1) \leq L^T(2)$); remove column 4 of N and row 4 of L (because $N(4) \leq N(3)$ and $L^T(4) \leq L^T(3)$). This yields the reduced matrices:

$$I_0 = \begin{array}{c} \\ B_c \\ B_p \\ B_s \\ S \\ C \\ P \\ M \\ PIN \end{array} \begin{array}{c} \begin{array}{cccc} B_c & B_p & B_s & S \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array} \qquad N = \begin{array}{c} \\ B_c \\ B_p \\ B_s \\ S \\ C \\ P \\ M \\ PIN \end{array} \begin{array}{c} \begin{array}{cccc} B_pB_c & PB_p & B_pB_s & B_sS \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1-e & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1-e & 0 & 0 \end{bmatrix} \end{array} \quad (3)$$

and

$$L \quad = \quad \begin{array}{c} \\ B_pB_c \\ PB_p \\ B_pB_s \\ B_sS \end{array} \begin{array}{c} \begin{array}{cccc} B_c & B_p & B_s & S \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & e \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \tag{4}$$

Second, derive from (I_0, L, N) the matrices I_1 and U. Using (1) and (4), the information each player has after collusion with cx is

$$I_1 \;=\; I_0 \;+\; N \cdot L \;\;=\; \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & e \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Comparing I_0 and I_1 confirms that none of B_c, B_p, B_s, S knows more about C, P, M after colluding with cx than before.

Using (2) and (4), the matrix U is

$$U \quad = \quad \begin{array}{c} \\ B_c \\ B_p \\ B_s \\ S \end{array} \begin{array}{c} \begin{array}{cccc} B_c & B_p & B_s & S \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & e \\ 0 & 1 & 1 & 1 \\ 0 & e & 1 & 1 \end{bmatrix} \end{array}$$

Observe that if $U(i,j) = 1$, then $I_1(i,j) = I_1(j,i) = 1$. This says that if players i and j share a link unique to the transaction, then they must know each other after colluding with cx, and hence can always combine their information if they choose to collude.

Finally, we use I_1 and U to construct the collusion graph shown in Figure 1. To associate C with M, it is necessary to find a tree that spans vertices labelled by (B_c, C) and by (S, M). The path $B_c \rightarrow B_p \rightarrow B_s \rightarrow S$ of solid edges allows us to draw the following conclusion.

Proposition 3 *1. If S does not eavesdrop, then all players have to collude to associate C with M.*

2. If S eavesdrops, then all players except B_s have to collude to associate C with M.

In addition, if S does not eavesdrop, then B_p knows that P shops at S (but S does not know B_p), but it has to collude with B_c to relate C to S.

Notice that all of the above argument holds even if B_s happens to be collocated with B_p or B_c, or both.

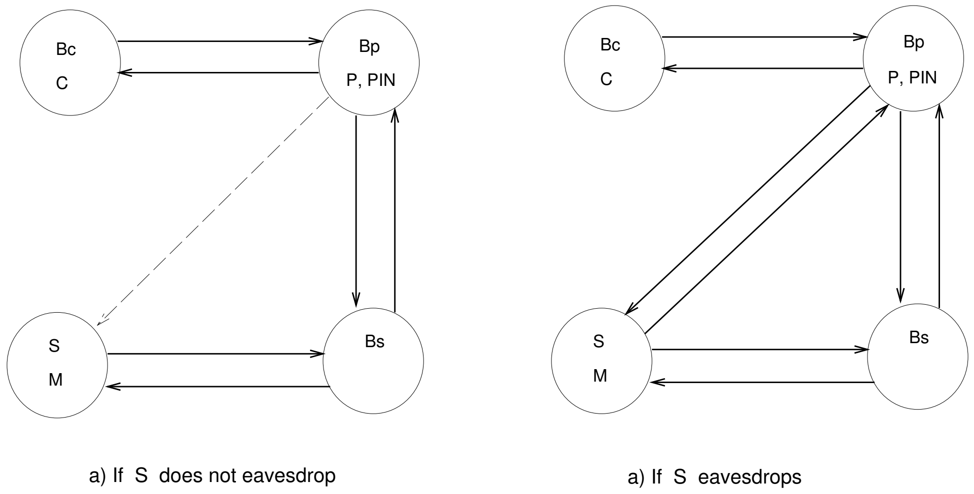

Figure 1: Collusion graph

4 Simplified implementation

Notice from matrix I_0 in §3.3, which describes the information players have after the purchase but before collusion with cx, that no party knows both P and M, or both P and S. This means that if P were the true identity rather than a pseudonym, the customer cannot be compromised if cx is trusted. Moreover, Proposition 3 still holds, as can be seen by coalescing vertices labelled by (B_c, C) and by (B_p, P, PIN) in the collusion graphs of Figure 1. This suggests the following simplified implementation of anonymous credit cards.

In the simplified implementation, the customer C need maintain an account only at the card-issuing bank B_c. This eliminates the need for subprotocols 1 and 3 to place credits in (B_p, P) by B_c and to bill (B_c, C) by B_p. Only a simplified protocol is needed during each purchase to transfer fund from B_c to the store's account (B_s, S). It is the same as subprotocol 2 in §2.3, except that all references to P and B_p are replaced by C and B_c, respectively.

We want to emphasize that, even though the simplified implementation is equally secure in the sense of Proposition 3, the original protocol may offer additional protection not considered in our model. For instance, if each party independently decides whether or not to collude, the customer using the complete protocol may have a smaller chance of being compromised.

5 Conclusion

We have presented a formal method to analyze collusion in a multi-party protocol, and have applied it to the protocol for anonymous credit cards proposed in [8]. Our analysis suggests a simplified implementation that is 'equally secure' in that, in both the original and the simplified implementations, all parties have to collude in order to compromise the customer; and that if the store eavesdrops, then all parties but the store's bank need collude to compromise the customer. We are currently implementing a prototype anonymous credit card.

Acknowledgements: We are grateful to D. Moews and A. Odlyzko for the analysis in the Appendix of the set N of note numbers in the scheme for electronic cash of [3].

6 Appendix

6.1 Scheme for Electronic Cash

Reference [3] proposes a scheme to create electronic cash, In the scheme, a bank keeps a set of integer pairs (p, s) as cryptographic keys, a pair for each denomination of the currency, such that $ps = 1 \pmod{\phi(m)}$ where m is a large composite integer. p is made public and s is kept secret to the bank. It is assumed as usual that it is infeasible to compute s from p. To buy electronic cash from the bank, a customer chooses a note number n and a random number r, and sends nr^p to the bank, where (p, s) is the pair corresponding to the requested amount. The bank returns to the customer $(nr^p)^s = n^s r$. To use the cash, the customer strips off the random number r, and gives the store n^s as payment, which the store can redeem at the bank. The random number r prevents the bank from using the note number n to associate the customer with the purchase.

As alluded to in [4, Section 2.2], the scheme may be insecure against fraud: a customer may be able to combine electronic cash obtained from the bank to generate new cash, e.g., multiply n_1^s and n_2^s to generate new cash $(n_1 n_2)^s \pmod{m}$. The customer will not be able to cheat if the set N of note numbers satisfies the property that, for all $n_1, \ldots, n_k$ in N, not necessarily distinct, $k = 2, 3, \ldots$,

$$n_1 \cdots n_k \quad \in \quad N^c \cup \{n_1, \ldots, n_k\} \tag{5}$$

where N^c is the complement of N. Such a set N, however, is very small, as the following result of [9] shows.

Proposition 4 *For $m = pq$, p, q prime numbers, the maximum cardinality of N satisfying (5) is upper bounded by $2f(p-1) + 2f(q-1) + 4$, where $f(x)$ = number of distinct prime factors of x.*

Since the publication of [3], there are many extensions to the basic idea; see e.g. [2, 11, 1] and references therein. However, the above problem is still not unconditionally resolved.

6.2 Proof of Proposition 1

Note that the kth column of $N \cdot L$ is given by

$$(N \cdot L)(k) \quad = \quad L(1, k) \cdot N(1) + \ldots + L(|Msg|, k) \cdot N(|Msg|) \tag{6}$$

It is easy to verify that, given $a_1, a_2 \in \{0, 1\}$ and two binary vectors v_1, v_2, if $a_1 \cdot v_1 \leq a_2 \cdot v_2$, then $a_1 \cdot v_1 + a_2 \cdot v_2 = a_2 \cdot v_2$. This allows us to remove $L(j, k) \cdot N(j)$ from the right-hand side of (6) which then equals $(N' \cdot L')(k)$, the kth column of $N' \cdot L'$. That $U(L) = U(L')$ follows from (2) and $L^T(j) \leq L^T(k)$. □

6.3 Proof of Proposition 2

Now, i can obtain a piece of information k (not in $Player$) if either $I_1(i, k) = 1$ or if there is a vertex l in the collusion graph G labelled by information k that is reachable from i through a directed path consisting of solid edges. Since $I_1(i) \leq I_1(j)$ and hence $I_1(i, k) = 1$ $\Rightarrow I_1(j, k) = 1$, it is enough to prove that if vertex i can reach vertex l through such a path, so can vertex j.

By construction of G, $I_1(i) \leq I_1(j)$ implies that if there is an edge from i to any other vertex, then there is also an edge from j to that vertex. Moreover, if the edge from i is solid, so is the edge from j since $U(i) \leq U(j)$. Hence, any vertex reachable from i by solid edges is also reachable from j by solid edges. □

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] David Chaum. Privacy protected payments: unconditional payer and/or payee untraceability. In D. Chaum and I. Schaumüller-Bichl, editors, *SmartCard 2000*. North Holland, 1989.

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

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

[7] Ki Hang Kim. *Boolean matrix theory and applications.* Marcel Dekker, Inc., 1982.

[8] S. H. Low, N. F. Maxemchuk, and S. Paul. Anonymous credit cards. Submitted to IEEE Symposium on Security and Privacy, 1994.

[9] David Moews, 1993. Personal communication.

[10] Judy H. Moore. Protocol failures in cryptosystems. *Proceedings of the IEEE*, 76(5):594–602, May 1988.

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

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

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