Collusion Analysis of Cryptographic Protocols

Steven H. Low

Department of EEE
University of Melbourne
VIC 3052 Australia

Nicholas F. Maxemchuk

Bell Laboratories
Lucent Technologies
NJ 07974 USA

Abstract

As network applications such as electronic commerce proliferate, complex communications protocols that employ cryptographic building blocks, such as encryption and authentication, will become more common. 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 after a protocol is executed. Earlier we introduced a model for cryptographic protocols and its collusion analysis, and solved a special case of the collusion problem. In this paper we present an algorithm that solves the general case.

1 Introduction

In a recent paper [8] we define and formally analyze a new kind of cryptographic property we call collusion. This is motivated by our earlier work [9] and [10] in which we designed multiparty communications protocols to implement anonymous credit cards and health insurance systems. In the anonymous credit card context, for instance, the protocol uses standard cryptographic techniques to hide different pieces of transaction information from different parties so that at the end of a credit card transaction, no single party except the cardholder can associate the cardholder's identity with where or what she purchases. Moreover it takes all but one party to collude in order to compromise the cardholder's privacy. For our purposes, a cryptographic protocol is a process by which some information is transferred among some users and hidden from others. The collusion problem determines whether it is possible for a subset of users to discover, through collusion, the information that is to be hidden from them after a protocol has been executed.

It is not always possible for two users to collude. In order to collude they might have to share a common, unique piece of information pertaining to the protocol run. This might be necessary because, in practice, the sender and the receiver, such as banks in the anonymous credit card context, may have exchanged a large number of messages corresponding to different protocol runs within a short time interval. In order to combine their knowledge pertaining to a particular protocol run, they need to share a unique link pertaining to that run. There are two possible such links: 1) a unique message that is exchanged during that protocol run, or 2) a unique piece of data, e.g, two banks may have the unique social security number of a customer and hence can combine their knowledge about the customer.

In [8] we solve the collusion problem for the special case where collusion is only allowed between two colluders sharing unique messages. In this paper we solve the general case where collusion can also occur if they share unique data. The formulation here includes the special case in which the only prerequisite for collusion is that the colluders know each other; see §2.

Cryptographic protocols are notoriously hard to design and their correctness is harder to prove [17]. Numerous cryptographic protocols have been published and later found to contain security flaws, see, e.g., [14, 2, 15, 18, 16, 11, 13, 1]. As surveyed in [13] these often subtle failures do not require eroding the integrity of the underlying cryptoalgorithm and hence are weaknesses of the protocols. Traditionally the soundness of a cryptographic protocol is evaluated by having experts attempt to find flaws in the protocol. When flaws are found the protocol is often modified and then the cycle repeats. Examples abound in the literature that have survived extensive and intensive scrutiny, only to have been shown later to contain a protocol failure. They clearly demonstrate the need for formal methods to verify cryptographic properties of protocols; see e.g. [4, 3, 1, 6, 12, 7, 11].

In §2, we present our model and problem statement. In §3 we review the special case where collusion is allowed only between two colluders sharing unique messages. The solution illustrates the structure of the problem and leads to the solution for the general case when collusion is allowed between colluders sharing unique data as well. This is explained in §4. Due to space limit, we omit all proofs, which can be found in the full version of this paper.

2 Model and Problem Formulation

In this section we present our model and formulate the collusion problem.

0-7803-3336-5/96 $5.00

2.1 Notation

We use $(y_j, j \in J)$ to denote a vector with components y_j, j spanning the index set J; the jth component y_j is sometimes denoted $y.j$. For any set A, $|A|$ denotes the number of elements in A, 2^A denotes the collection of subsets of A, and A^* denotes its Kleen closure.

If d is a piece of data and k is a cryptographic key, then $k(d)$ denotes the encryption of d with k. A string $k_n \cdots k_1$ represents successive application of keys $k_1, \ldots, k_n$ in order. We use ϵ to denote the identity key: for any piece of data d, $\epsilon(d) = d$.

For any key k, k^{-1} denotes its inverse with the cancellation rule $k^{-1}\ k = k\ k^{-1} = \epsilon$. For example, The keys k and k^{-1} are identical in secret-key cryptosystems, but not in public-key cryptosystems. When we refer to a string γ of keys, we always assume that γ is in reduced form that cannot be further simplified by application of the cancellation rule.

A *transition system* is a triple $\Theta = (Q, \Sigma, \delta)$, where Q is a set of states, Σ is a finite set of transition labels, and $\delta : Q \times \Sigma \rightarrow Q$ is a transition (partial) function. For example, a finite state machine is a special transition system in which Q is finite. A *path* is a sequence of transitions. A concatenation of two paths ρ_1 and ρ_2 is denoted $\rho_1 \cdot \rho_2$.

2.2 Model

A protocol is executed by a set of users who exchange information by transmitting messages. The users who can use the information is controlled by encrypting the information with a set of keys. In each step of the protocol a sender transfers a message to a receiver. The message is encrypted by a string of keys and may contain pieces that the receiver can decrypt, those that it cannot decrypt, as well as keys that can decrypt part of its current information. Each message is made unique to prevent replay attack. We refer to a message by a unique message number. The receiver's knowledge is increased by the content of the message. Moreover, the sender and the receiver now share a unique message that can be used for collusion purposes, as we will see below. A user's knowledge is the set of information and the set of messages that it possesses, and is modified as protocol proceeds. The protocol specifies all possible sequences of message exchanges that are allowed and the evolution of the users' knowledge. Protocol execution terminates when the users' knowledge reaches a certain prespecified pattern. For example, in an on-line payment protocol, e.g. [5, 9], execution is complete when the store receives confirmation from its bank that funds has been transferred from the customer account to the store's account.

We are interested in whether a subset of protocol users can discover, during or after the protocol's execution, the information that is being hidden from them. They discover the hidden information by combining the information that they possess. Combining the information is modeled as sending additional messages among the subset of users we call colluders. We now make these notions precise.

Collusion is carried out in an *environment* described by the 5-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 colluders can collude; see condition (2) below.

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

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

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.

If $A = \{k^{-1}, k(d)\}$ then the key k^{-1} can be used to decrypt $k(d)$ and hence A is 'equivalent to' $\{k, d\}$. In general if $A \subseteq I$ then A can be reduced to an 'equivalent' set by repeated application of the cancellation rule. We abstractly describe this transformation by the function $\Delta : 2^I \rightarrow 2^I$. Δ represents the decryption of a set $A \subseteq I$ of information by the keys included in A such that if $k_1 \cdots k_n(d) \in \Delta(A)$ then $k_1^{-1} \notin \Delta(A)$.

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. User u's knowledge is denoted $w_u \in W$. We naturally assume that $u \in w_u.I$ for all $u \in U_p$.

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.

As explained in §1 in order for two colluders s and r to collude, not only must the sender s know the receiver r, they must also share a unique message that is exchanged during

that protocol run, or a unique piece of data in L. 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:

$$w_s.N \cap w_r.N \neq \phi \quad (1)$$
$$w_s.I \cap w_r.I \cap L \neq \phi. \quad (2)$$

If $L = U_c$ then since $u \in w_u$ for all u, $r \in w_s.I$ implies condition (2). Hence the above conditions 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:

$$w'_y = w_y, \quad \text{if } y \neq r$$
$$w'_r.N = w_r.N \cup w_s.N, \quad w'_r.I = \Delta(w_r.I \cup w_s.I)$$

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

Note that the set of users that can collude can increase as users collude and information is combined. For instance, a sender can have encrypted information that includes the identity of a user and a unique piece of data that the sender and that user shares, and a receiver may have the key to decrypt that information. After the sender transfers its information to the receiver, the receiver can collude with the user that was hidden in the encrypted information.

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.*

2.3 Problem formulation

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 exists 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.*

3 Special case: collusion on unique messages only

In this section we consider the special case in which $L = \phi$, i.e., two users can collude only if they share a unique message that was exchanged during the protocol run (condition (1)). This problem is solved in [8]. Here we review the main results there that illustrate the structure of the problem. The general case will be treated in the next section.

For the rest of this sectoin, fix an environment (U_p, D, K, U_c, L) where $L = \phi$, an initial state $w(0)$ and a target information set T. It turns out that, to solve the collusion problem, it is necessary and sufficient to find a set of colluders whose *initial* information in state $w(0)$, when combined, yields T and to find a way for all of them to communicate their information to the same colluder. Moreover, such a set of colluders, if exist, must have exchanged messages among themselves in the protocol phase, and hence can be determined from the initial state $w(0)$. This provides an exact characterization in terms of $w(0)$ of when a collusion path exists.

Define $F = (U_c, E(w(0)))$ as an undirected graph, depending on the initial state $w(0)$, that describes all the events that are initially enabled by condition (1). F contains all colluders as its nodes. There is an edge (u, v) in $E(w(0))$ if and only if (u, v) is enabled in the initial state $w(0)$.

Theorem 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 $\Delta(\cup_{u \in V} w_u(0).I) \supseteq T$.*

The theorem gives the condition under which the collusion problem has a solution but it does not suggests any collusion paths. The next result clarifies the simple structure of collusion paths.

Theorem 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.

These two theorems suggest an algorithm to solve the collusion problem. It first constructs the graph F that describes all events that are initially enabled by condition (1). Then it finds each connected component $F' = (V, E)$ of F using breadth-first search while, at the same time, constructing a possible collusion path with the structure given in Theorem 2. The path has visited every node in the connected component when the search on that connected component is complete. Hence the last recipient on the path knows the combined knowledge of every colluder in the connected component:

$$\Delta(\cup_{u \in V} w_u(0).I).$$

This combined knowledge is then checked to see if it contains the target information set T. A collusion path is found if

it does; otherwise the search is repeated on a different connected component of F. When all connected components have been searched without producing a collusion path, Theorem 1 guarantees that none exists.

4 General case

When collusion can be enabled by unique data in L as well, the condition under which a solution for the collusion problem exists is no longer a simple expression as in Theorem 1 for the special case, but it can still be determined from just the initial state $w(0)$. In this section we sketch an iterative algorithm that verifies if a solution exists. The algorithm includes the condition in Theorem 1 as its first iteration.

Consider again the graph $F(0) = (V(0), E(0))$, $V(0) := U_c$, defined in the last section, that describes all the events that are enabled in $w(0)$ by shared message, i.e., there is an edge (u, v) in $E(0)$ if and only if (v, u) satisfies condition (1) in state $w(0)$. Note that $E(0)$ does *not* include (u, v) that satisfies only condition (2) but not (1) in state $w(0)$. From the last section we know that

1. initial knowledge of *all* colluders in the same connected component can be combined, and
2. this combined knowledge can be transmitted to *any* colluder in the connected component.

The second statement is true since, by construction of the path, the last recipient knows the combined knowledge and every colluder in the connected component, and hence can transmit the combined knowledge to any one of them.

Suppose $F(0)$ has $k(0)$ connected components $F_i(0) = (V_i(0), E_i(0))$, $i = 1, 2, ..., k(0)$. For each connected component $F_i(0)$ define

$$W_i(0) \quad := \quad \Delta(\cup_{u \in V_i(0)} w_u(0).I).$$

We call $V_i(0)$ a *group* of colluders and $W_i(0)$ its *collective knowledge*. We identify the connected component $F_i(0)$ with $(V_i(0), W_i(0))$.

Theorem 1 implies that it is not possible to collude across two connected components *only* on unique messages, i.e., if collusion is allowed only between (u, v) that satisfies condition (1). Hence after colluders in each connected component have already combined their knowledge further collusion can only be enabled by (u, v) satisfying condition (2) for some u and v in *different* connected components. As noted above we can assume that u and v know the collective knowledge of their respective groups. Hence collusion between two connected components identified by $(V_i(0), W_i(0))$ and $(V_j(0), W_j(0))$ is possible if

$$W_i(0) \cap V_j(0) \quad \neq \quad \phi \qquad (3)$$

$$W_i(0) \cap W_j(0) \cap L \quad \neq \quad \phi. \qquad (4)$$

The first condition says that there is a $u_j \in V_j(0)$ that is known to (some colluder in) the group $V_i(0)$; the second condition says that the two groups share a unique data in L that enables collusion.

Note that the first condition implies that there is a $u_i \in V_i(0)$ which knows some $u_j \in V_j(0)$, but not necessarily vice versa, and thus u_i can transmit its complete knowledge to u_j, but not necessarily vice versa. After this transmission, however, u_j also knows u_i and hence can transmit the combined knowledge back to u_i. Hence when two groups $V_i(0)$ and $V_j(0)$ collude it is again possible to combine their collective knowledge and transmit it to *any* colluders in $V_i(0) \cup V_j(0)$.

More generally we can construct a new graph $F(1) = (V(1), E(1))$ from the collection of connected components of $F_i(0)$, $i = 1, \ldots, k(0)$, of $F(0)$ as follows. There are $k(0)$ nodes in $V(1)$, one defined for each connected component $F_i(0)$ and identified with $(V_i(0), W_i(0))$, the group of colluders in $F_i(0)$ and their collective knowledge. There is an ***undirected*** edge between two nodes $(V_i(0), W_i(0))$ and $(V_j(0), W_j(0))$ if conditions (3–4) are satisfied. Undirected edges indicate that it is possible to transmit the combined knowledge $\Delta(W_i(0) \cup W_j(0))$ to either group $V_i(0)$ or $V_j(0)$.

It is then possible to collect the knowledge of all colluders in a connected component of the *new* graph $F(1)$ and transmit it to *any* colluder in the connected component. After such collusion new user identities and unique data may be revealed which enable further collusion across connected components, and the cycle repeats.

We summarize out discussion in the following algorithm (sketch). The result in the previous section for the special case is indeed embodied in the first iteration of the algorithm.

Algorithm

Input: An initial state $w(0)$ and a target information set T.

Output: YES if the collusion problem has a solution; NO otherwise.

1. Construct from $w(0)$ the undirected graph $F(0) = (V(0), E(0))$, $V(0) = U_c$, that describes all events initially enabled by unique messages.

2. Construct connected components $F_i(0) = (V_i(0), E_i(0))$ of $F(0)$, $i = 1, \ldots, k(0)$. Identify each connected component $F_i(0)$ with the pair $(V_i(0), W_i(0))$ where $W_i(0) := \Delta(\cup_{u \in V_i(0)} w_u(0).I)$.

3. If there is a connected component $F_i(0)$ with $W_i(0) \supseteq T$, then **Return** YES. Otherwise set $n = 0$.

4. Construct undirected graph $F(n+1) = (V(n+1), E(n+1))$ from graph $F(n)$ as follows. $V(n+1)$ contains $k(n)$ nodes, one for each connected component $F_i(n)$ of $F(n)$. Each node in $V(n+1)$ is identified with $(V_i(n), W_i(n))$. Two nodes $(V_i(n), W_i(n))$ and $(V_j(n), W_j(n))$ in $V(n+1)$ is connected if and only if $W_i(n) \cap V_j(n) \neq \phi$ and $W_i(n) \cap W_j(n) \cap L \neq \phi$.

If $E(n+1)$ is empty, **Return** NO.

5. Construct connected components $F_i(n+1) = (V_i(n+1), E_i(n+1))$ of $F(n+1)$, $i = 1, \ldots, k(n+1)$. Identify each connected component $F_i(n+1)$ with the pair $(V_i(n+1), W_i(n+1))$. Here, we abuse notation to use $V_i(n+1)$ to denote the constituent colluders $\{ u \mid u \in V_j(n), (V_j(n), W_j(n)) \text{ in } F_i(n+1) \}$. $W_i(n+1) := \Delta(\cup_{u \in V_i(n+1)} w_u(0).I)$.

6. If there is a connected component $F_i(n+1)$ with $W_i(n+1) \supseteq T$, then **Return** YES. Otherwise increment n and goto Step 4.

Theorem 3 *The algorithm terminates. Moreover it terminates with* YES *if the collusion problem has a solution and* NO *otherwise.*

The above algorithm can be augmented so that it not only determines whether a collusion path exists, but also constructs one when it does.

References

[1] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.

[2] D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, August 1981.

[3] D. Dolev, S. Even, and R. M. Karp. On the security of Ping-Pong protocols. *Information and Control*, 55:57–68, 1982.

[4] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(2):198–208, March 1983.

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

[6] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about belief in cryptographic protocols. *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, pages 234–248, May 1990.

[7] Richard A. Kemmerer. Analyzing encryption protocols using formal verification techniques. *IEEE Journal on Selected Areas in Communications*, 7(4):448–457, May 1989.

[8] S. H. Low and N. F. Maxemchuk. A Model for Collusion in cryptographic protocols. Submitted to *First Workshop on Information Hiding*, University of Cambridge, June 1996.

[9] S. H. Low, N. F. Maxemchuk, and S. Paul. Anonymous credit cards. *Proceedings of the 2nd. ACM Conference on Computer and Communications Security*, November 2–4 1994.

[10] N. F. Maxemchuk and S. H. Low. The use of communications networks to increase personal privacy. *Proceedings of Infocom'95*, pages 504–512, April 1995.

[11] Catherine Meadows. A system for the specification and analysis of key management protocols. *Proceedings of the 1991 IEEE Symposium on Security and Privacy*, pages 182–195, May 1991.

[12] Jonathan K. Millen. The Interrogator: A Tool for Cryptographic Protocol Security. *Proceedings of the 1984 IEEE Symposium on Security and Privacy*, pages 134–141, May 1984.

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

[14] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[15] Roger M. Needham and Michael D. Schroeder. Authentication revisited. *ACM Operating System Review*, 21(1):7–7, January 1987.

[16] G. J. Simmons. How to (selectively) broadcast a secret. *Proceedings of the 1985 IEEE Symposium on Security and Privacy*, pages 108–113, May 1985.

[17] G. J. Simmons. Proof of soundness (integrity) of cryptographic protocols. *Journal of Cryptology*, 7(2):69–77, Spring 1994.

[18] M. Tatebayashi, N. Matsuzaki, and D. B. Newman. Key distribution protocol for digital mobil communication systems. In G. Brassard, editor, *Advances in Cryptology - CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 324–333. Springer–Verlag, New York, 1991.