Posets and Protocols—Picking the Right Three-Party Protocol

Siaw-Lynn Ng

***Abstract*—In this paper, we introduce a framework in which we can investigate the possibility of adapting a security protocol in order to obtain optimal efficiency according to the communication channels available. This method is based on the observation that there is a partial order imposed upon the actions of the various parties involved in a protocol. We define operations permitted on the partially ordered set associated with the protocol and obtain transformations of the original protocol while preserving the security properties. Performing these operations on the protocol we enumerate the options available to a system.**

***Index Terms*—Authentication protocols, partially ordered sets, strand space model.**

I. Introduction

THE FOCUS of research in authentication protocols has predominantly been on security. There has not been much work published on the analysis of protocol efficiency in terms of communications. In [1], Gong proved lower bounds on the number of messages and the number of rounds for a protocol in a collection of different settings and goals. This serves as a useful reference for system and protocol designers. An example given in the paper is that a protocol version which has more messages and fewer rounds (hence, faster to complete) may be suitable in the case where a client needs immediate authentication, while in the situation where the network is unreliable the client may wish to run the version with fewer messages.

There are situations, however, where even in an unreliable network, a protocol with fewer but longer messages may not be a good choice. For example, in a channel where errors occur in bursts it may be more efficient to send shorter messages so that on one hand the possibility of errors occurring while transmitting is minimized, and on the other hand if a message has to be sent again at least it is short. Another example where short messages are required is the short message service (SMS), where a single short message is restricted to a maximum length of 140 bytes. In another scenario, the channels available between each pair of the participants in a three-party protocol may be different. For example, a home user may have a slow link to the service provider, who may have a fast link to a third party. In such situations, a round or message-optimal version of a protocol may not be the most suitable version.

In this paper, we introduce a framework in which we can investigate the possibility of adapting a security protocol in order to obtain optimal efficiency according to the communication channels available. This method is based on the observation that there is a partial order imposed upon the actions of the various parties involved in a protocol. We define operations permitted on the partially ordered set associated with the protocol and obtain transformations of the original protocol while preserving the security properties. Performing these operations on the protocol we enumerate the options available to a system. We will describe our method in detail in Section II. In Section III, we discuss the security implications of our operations. We will use the strand space model [2] for this purpose. We conclude with a discussion of possible extensions to our work.

Manuscript received February, 2002; revised July 17, 2002.

The author is with the Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, U.K. (e-mail: s.ng@rhul.ac.uk).

Digital Object Identifier 10.1109/JSAC.2002.806126

II. A Method for Adapting Protocols

In an authentication protocol, messages are exchanged between parties, and some messages must necessarily be sent before others. For example, in a handshake, the session key must be received before a handshake message can be sent. Hence, there is a natural partial order imposed upon the actions of the various parties involved.

We may view these actions and the order imposed on them in a particular protocol as a partially ordered set (*poset*) $\mathcal{A}$ with a strict partial order "$<$" defined as follows: for all $x, y \in \mathcal{A}$, $x < y \iff$ action x must be performed before action y can be.

We will give a more precise definition of action in Section II-A. A poset $(\mathcal{A} = \{a_1, \ldots, a_k\}, <)$ may be represented as $(\mathcal{A}, \mathcal{R})$, where $\mathcal{R} = \{(x,y) | x, y \in \mathcal{A}, x < y\}$, or as a Hasse diagram, which is a graph with vertices corresponding to elements of $\mathcal{A}$, and the vertices x, y are joined by an edge with x "below" y if $x < y$ and there are no vertices "in between." We will use both representations. We say that two elements x, y are *comparable* if $x < y$ or $y < x$. Otherwise, they are *incomparable*. A poset where every pair of element is comparable is a *chain*. The basic definitions of posets and Hasse diagrams may be found in [3]. In addition, to each element of $\mathcal{A}$ we assign a label (X, Y) if it corresponds to a message sent from X to Y.

Now, we have made the observation that, intuitively, an authentication protocol has an associated poset consisting of the protocol actions. This is not a new idea: in [4], Lamport defined a partial ordering of events in a distributed system using a "happened-before" relation. This idea is further developed by Yahalom [5], who used the notion of "verifiable causality" to prove bounds on the number of messages in a protocol. Here, we consider a refinement, in some sense, of the partial ordering described by Yahalom: we consider a partial ordering of the "atomic actions" of sending and receiving *components* of messages. We will give a definition of the relevant concepts in the following and describe how, given a protocol, we may obtain this associated poset. Then, we define certain operations that can be performed on this poset to obtain chains which will correspond to different versions of the protocols. These protocols can then be analyzed for their suitability to the communication channels available to the system. It will be shown in Section III that the security properties are preserved to some extent under these transformations.

A. Deconstructing a Protocol

We consider only three-party authentication protocols, where there are two clients, the protocol initiator and the responder, and an authentication server via which the clients agree on a session key. We also assume that the underlying cryptographic mechanisms used in the protocols guarantee message integrity. Further assumptions on the relationships between the parties are specific to individual protocols but will have no effect on our method.

Using the terminology of [2], the set of *messages* (called *terms*) are generated from two disjoint sets, the set T representing texts such as nonces or names, and the set K representing keys, by means of concatenation and encryption. A *component* is defined to be a term which is not a concatenation of terms. We define an *action* to be the sending of a term from one party to another, and we use the notation "$a\colon X \rightarrow Y\ M$" to denote the action a of sending term M from X to Y. An action is *atomic* if the term sent is a component.

A protocol $\mathcal{P}$ can be decomposed into the associated poset of its atomic actions as follows.

(D1) An action "$X \rightarrow Y\ M$", where M is a message with components $M_1, \ldots, M_h$, is decomposed into actions $a_1, \ldots, a_h$, where a_i, $i = 1, \ldots, h$, is the action "$X \rightarrow Y\ M_i$."

(D2) By inspection, we identify actions corresponding to the sending of a component to the receiving party via a third party, and modify the label of the action accordingly, that is, if there are pairs of actions a and b, where

$$a\colon X \rightarrow Z\ M, \quad b\colon Z \rightarrow Y\ M$$

such that M is simply passed on to Y by Z, we may replace b (and perhaps a, if M is not intended for Z as well) by

$$b'\colon X \rightarrow Y\ M.$$

This will allow us to examine the possibility of sending M in other ways in the reconstruction process.

The partial order on $\mathcal{A} = \{a_1, \ldots, a_k\}$ is determined based on the following simple observations.

1) The responder and the server cannot send out any message before the protocol initiator has sent out a first message.
2) A participant X cannot send out a message containing a nonce generated by another participant Y before X has received the nonce.
3) A participant cannot send out a message encrypted using a new key before it has received either the key or all the partial keys needed to generate the key.

Based on these observations, pairs of actions a and b

$$a\colon X \rightarrow Y\ M, \quad b\colon Y \rightarrow Z\ N$$

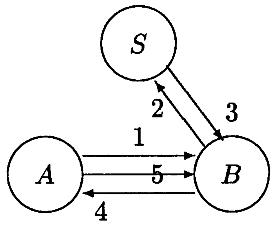
are examined and the relationships determined. We call $(\mathcal{A}, <)$ the *initial poset* associated with the protocol $\mathcal{P}$, and we may refer to $\mathcal{P}$ as the original protocol. The initial poset consists of atomic actions.

We illustrate this deconstruction process with an example.

The Guttman–Thayer protocol was given in [2] as an example of a design process which concentrates on the method of establishing authentication results proposed in the paper. There are three parties involved: A and B achieve mutual authentication, and the session key K established between A and B is generated by a trusted server S, who shares a secret key K_A with A and a secret key K_B with B. We use $[m]_K$ to denote encrypting m using the key K, and we use N_X to denote a nonce generated by X. The protocol is as follows.

Protocol 1A:

$$\begin{array}{ll}
1.\ A \rightarrow B & A, N_A \\
2.\ B \rightarrow S & A, N_A, B, N_B \\
3.\ S \rightarrow B & [N_A, B, K]_{K_A}, [N_B, A, K]_{K_B} \\
4.\ B \rightarrow A & [N_A, B, K]_{K_A}, [N_A]_K, N_B \\
5.\ A \rightarrow B & [N_B]_K.
\end{array}$$



Applying (D1) to the protocol, we obtain the actions $\{a_1, \ldots, a_9\}$

$$\begin{array}{lll}
a_1\colon & A \rightarrow B & A, N_A \\
a_2\colon & B \rightarrow S & A, N_A \\
a_3\colon & B \rightarrow S & B, N_B \\
a_4\colon & S \rightarrow B & [N_A, B, K]_{K_A} \\
a_5\colon & S \rightarrow B & [N_B, A, K]_{K_B} \\
a_6\colon & B \rightarrow A & [N_A, B, K]_{K_A} \\
a_7\colon & B \rightarrow A & [N_A]_K \\
a_8\colon & B \rightarrow A & N_B \\
a_9\colon & A \rightarrow B & [N_B]_K.
\end{array}$$

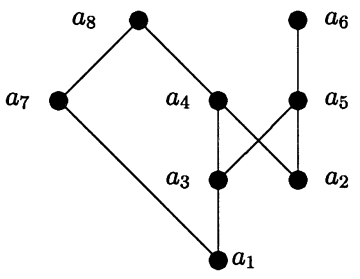
(Note that for simplicity, we treat A, N_A, and B, N_B as single components, since the names A and B are merely practical necessities so that recipients know to whom messages should be sent.)

Applying (D2), we see that a_2 may be modified to $A \rightarrow SA$, N_A, and a_4, a_6 may be replaced by $S \rightarrow A\ [N_A, B, K]_{K_A}$. Relabeling the actions and examining pairs of actions, we obtain the initial poset $\{a_1, \ldots, a_8\}$

$$\begin{array}{lll}
a_1\colon & A \rightarrow B & A, N_A \\
a_2\colon & A \rightarrow S & A, N_A \\
a_3\colon & B \rightarrow S & B, N_B \\
a_4\colon & S \rightarrow A & [N_A, B, K]_{K_A} \\
a_5\colon & S \rightarrow B & [N_B, A, K]_{K_B} \\
a_6\colon & B \rightarrow A & [N_A]_K
\end{array}$$

a_7: $B \rightarrow A \quad N_B$
a_8: $A \rightarrow B \quad [N_B]_K$.

$a_1 < a_3$,
$a_1 < a_6$,
$a_1 < a_7$,
$a_2 < a_4$,
$a_2 < a_5$,
$a_3 < a_4$,
$a_3 < a_5$,
$a_4 < a_8$,
$a_5 < a_6$,
$a_7 < a_8$,



B. Reconstructing Protocols

Given the initial poset $(\mathcal{A}, <)$ associated with a protocol $\mathcal{P}$ we define two operations (M1) and (M2), that may be performed on $\mathcal{A}$ while preserving the partial order on the atomic actions. Let $\mathcal{A} = \{a_1, \ldots, a_k\}$, and $\mathcal{R} = \{(x,y)|x,y \in \mathcal{A}, x < y\}$. Let the protocol initiator be A, the responder B and the server S. Every element a_i of $\mathcal{A}$ has a label $l_i = (X,Y)$, which is one of (A,B), (B,A), (A,S), (S,A), (B,S), (S,B). Operations (M1) and (M2) are defined as follows.

(M1) Two elements of $\mathcal{A}$ may be merged if they have the same label and they are incomparable. More precisely, let $a_x: X \rightarrow YM_1$ and $a_y: X \rightarrow YM_2$. Then we identify a_x with a_y if $(a_x,a_y) \notin \mathcal{R}$ and $(a_y,a_x) \notin \mathcal{R}$. We replace a_x, a_y with $a'_x = a_x|a_y$ to indicate the merging, where $a'_x: X \rightarrow YM_1M_2$, and define a new poset

$$\mathcal{A}' = \mathcal{A} \setminus \{a_x, a_y\} \cup \{a'_x\}$$

with the following partial order:

$$a_i < a_j \text{ if } a_i < a_j \text{ in } (\mathcal{A},\mathcal{R}), \quad a_i, a_j \in \mathcal{A}' \setminus \{a'_x\}$$
$$a'_x < a_i \text{ if } a_x < a_i \text{ or } a_y < a_i \text{ in } (\mathcal{A},\mathcal{R}), \quad a_i \in \mathcal{A}'$$
$$a_i < a'_x \text{ if } a_i < a_x \text{ or } a_i < a_y \text{ in } (\mathcal{A},\mathcal{R}), \quad a_i \in \mathcal{A}'.$$

Let $\mathcal{R}' = \{(x,y)|x,y \in \mathcal{A}', x < y\}$. This corresponds to merging two terms if they are not dependent on each other.

(M2) The element a_x may be merged with a_y and a_z if a_x has label (X,Z), $a_y(X,Y)$, $a_z(Y,Z)$ and the pairs $\{a_x,a_y\}$, $\{a_x,a_z\}$ are not comparable, and $a_z \not< a_y$. More precisely, if

$$a_x: X \rightarrow ZM_1$$
$$a_y: X \rightarrow YM_2$$
$$a_z: Y \rightarrow ZM_3$$

and (a_x,a_y), (a_x,a_z), (a_y,a_x), $(a_z,a_x) \notin \mathcal{R}$ and $(a_z,a_y) \notin \mathcal{R}$, then, we replace a_x, a_y, a_z with $a'_y = a_y|a_x$, $a'_z = a_z|a_x$, where

$$a'_y: X \rightarrow Y\ M_2M_1, \quad a'_z: Y \rightarrow Z\ M_3M_1.$$

We then define the new poset

$$\mathcal{A}' = \mathcal{A} \setminus \{a_x, a_y, a_z\} \cup \{a'_y, a'_z\}$$

with the following partial order:

$$a_i < a_j \text{ if } a_i < a_j \text{ in } (\mathcal{A},\mathcal{R}), \quad a_i, a_j \in \mathcal{A}' \setminus \{a'_y, a'_z\}$$
$$a'_y < a_i \text{ if } a_x < a_i \text{ or } a_y < a_i \text{ in } (\mathcal{A},\mathcal{R}), \quad a_i \in \mathcal{A}'$$
$$a'_z > a_i \text{ if } a_x < a_i \text{ or } a_z < a_i \text{ in } (\mathcal{A},\mathcal{R}), \quad a_i \in \mathcal{A}'$$
$$a_i < a'_y \text{ if } a_i < a_x \text{ or } a_i < a_y \text{ in } (\mathcal{A},\mathcal{R}), \quad a_i \in \mathcal{A}'$$
$$a_i < a'_z \text{ if } a_i < a_x \text{ or } a_i < a_z \text{ in } (\mathcal{A},\mathcal{R}), \quad a_i \in \mathcal{A}'.$$

Let $\mathcal{R}' = \{(x,y)|x,y \in \mathcal{A}', x < y\}$. This corresponds to sending a term M_1 from X to Z via a third party Y.

It is clear that the process of deconstructing a protocol (D1) and (D2) correspond to the inverses of (M1) and (M2). Having defined the permitted operations, we describe the procedure for reconstructing protocols from the initial poset $(\mathcal{A},\mathcal{R})$ of a protocol.

1) Given $(\mathcal{A},\mathcal{R})$, if $\mathcal{R}$ has $\binom{|\mathcal{A}|}{2}$ elements, then $\mathcal{A}$ is a chain and this corresponds to a version of the original protocol.
2) If $\mathcal{R}$ has less than $\binom{|\mathcal{A}|}{2}$ elements, then, we identify the possible operations of types (M1) and (M2) which can be performed on $\mathcal{A}$.
3) Apply one of the possible operations and establish $(\mathcal{A}',\mathcal{R}')$.
4) Repeat the procedure starting from step 1), using $(\mathcal{A}',\mathcal{R}')$ instead of $(\mathcal{A},\mathcal{R})$.

These steps are performed either using all the possible combinations of valid operations, or until a suitable protocol is found. A variation of the original protocol is produced each time we obtain a chain. Since $\mathcal{A}$ is finite, this procedure will terminate at some point. We illustrate this using the Guttman–Thayer protocol described previously.

The initial poset $(\mathcal{A},\mathcal{R})$ of the Guttman–Thayer protocol is $\mathcal{A} = \{a_1, \ldots, a_8\}$ with corresponding labels $l_1 = (A,B)$, $l_2 = (A,S)$, $l_3 = (B,S)$, $l_4 = (S,A)$, $l_5 = (S,B)$, $l_6 = (B,A)$, $l_7 = (B,A)$, $l_8 = (A,B)$, and

$$\begin{aligned}\mathcal{R} = \{&(a_1,a_3),(a_1,a_4),(a_1,a_5),(a_1,a_6),(a_1,a_7),\\&(a_1,a_8),(a_2,a_4),(a_2,a_5),(a_2,a_6),(a_2,a_8),\\&(a_3,a_4),(a_3,a_5),(a_3,a_6),(a_3,a_8),(a_4,a_8),\\&(a_5,a_6),(a_7,a_8)\}.\end{aligned}$$

We see that $|\mathcal{R}| = 17 < \binom{|\mathcal{A}|}{2} = 28$. The only possible (M1) operation is merging a_6 and a_7, while the possible (M2) operations are

i) merge a_3 with a_7 and a_2;
ii) merge a_2 with a_1 and a_3;
iii) merge a_4 with a_5 and a_6;
iv) merge a_4 with a_5 and a_7;
v) merge a_5 with a_4 and a_8;
vi) merge a_7 with a_3 and a_4.

Suppose we apply operation (i). This gives us $\mathcal{A}' = \{a_1, a'_2 = a_2|a_3, a_4, a_5, a_6, a'_7 = a_7|a_3, a_8\}$, with corresponding labels $l_1 = (A,B)$, $l'_2 = (A,S)$, $l_4 = (S,A)$, $l_5 = (S,B)$, $l_6 = (B,A)$, $l'_7 = (B,A)$, $l_8 = (A,B)$, and

$$\begin{aligned}\mathcal{R}' = \{&(a_1,a'_2),(a_1,a_4),(a_1,a_5),(a_1,a_6),(a_1,a'_7),\\&(a_1,a_8),(a'_2,a_4),(a'_2,a_5),(a'_2,a_6),(a'_2,a_8),\\&(a_4,a_8),(a_5,a_6),(a'_7,a'_2),(a'_7,a_4),(a'_7,a_5),\\&(a'_7,a_6),(a'_7,a_8)\}.\end{aligned}$$

This gives $|\mathcal{R}'| = 17 < \binom{|\mathcal{A}'|}{2} = 21$. There is now no possible (M1) operation, while the two possible (M2) operations are merging a_5 with a_4 and a_8, and merging a_4 with a_5 and a_6.

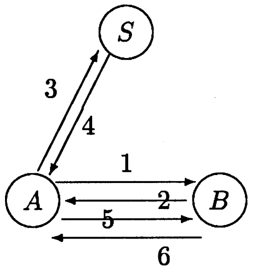
Suppose we apply the first operation (merging a_5 with a_4 and a_8). This gives us $\mathcal{A}'' = \{a_1, a_2' = a_2|a_3, a_4' = a_4|a_5, a_6, a_7' = a_7|a_3, a_8' = a_8|a_5\}$, with corresponding labels $l_1 = (A,B)$, $l_2' = (A,S)$, $l_4' = (S,A)$, $l_6 = (B,A)$, $l_7' = (B,A)$, $l_8' = (A,B)$, and

$$\begin{aligned}\mathcal{R}'' = \{&(a_1,a_2'),(a_1,a_4'),(a_1,a_6),(a_1,a_7'),(a_1,a_8'),\\&(a_2',a_4'),(a_2',a_6),(a_2',a_8'),(a_4',a_6),(a_4',a_8'),\\&(a_7',a_2'),(a_7',a_4'),(a_7',a_6),(a_7',a_8'),(a_8',a_6)\}.\end{aligned}$$

At this stage, we see that $|\mathcal{R}''| = 15 = \binom{|\mathcal{A}''|}{2}$. Hence, $\mathcal{A}''$ is a chain. Expanding $\mathcal{A}''$, we get the following protocol.

Protocol 1B:

1. $A \to B \quad A, N_A$
2. $B \to A \quad B, N_B$
3. $A \to S \quad A, N_A, B, N_B$
4. $S \to A \quad [N_A, B, K]_{K_A}, [N_B, A, K]_{K_B}$
5. $A \to B \quad [N_B, A, K]_{K_B}, [N_B]_K$
6. $B \to A \quad [N_A]_K.$

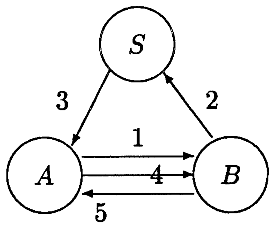


On the other hand, applying the following operations: merge a_6 and a_7 to get $a_6' = a_6|a_7$, merge a_2 with a_1, a_3, and merge a_4 with a_5, a_6' gives the original protocol, Protocol 1A.

Repeating the process using other combinations of the possible operations, we obtain, in addition to Protocol 1A and Protocol 1B, the following two protocols.

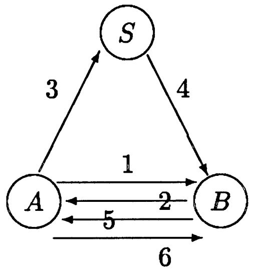
Protocol 1C:

1. $A \to B \quad A, N_A$
2. $B \to S \quad A, N_A, B, N_B$
3. $S \to A \quad [N_A, B, K]_{K_A}, [N_B, A, K]_{K_B}, N_B$
4. $A \to B \quad [N_B, A, K]_{K_B}, [N_B]_K$
5. $B \to A \quad [N_A]_K.$



Protocol 1D:

1. $A \to B \quad A, N_A$
2. $B \to A \quad B, N_B$
3. $A \to S \quad A, N_A, B, N_B$
4. $S \to B \quad [N_A, B, K]_{K_A}, [N_B, A, K]_{K_B}$
5. $B \to A \quad [N_A, B, K]_{K_A}, [N_A]_K$
6. $A \to B \quad [N_B]_K.$



This example illustrates what can be achieved in this method. We may choose to implement Protocol 1A, 1B, 1C, or 1D, according to the type of communication channels that are available between each pair of participants. For example, in Protocol 1A there is no direct communication between A and S, in Protocol 1B there is no direct communication between B and S, while both A and B communicate with S in Protocols 1C and 1D.

In Section III, we show that the security properties of the derived protocols are not significantly different from the original protocol. We will also show how an authentication proof in the strand space model can be derived from the authentication proof of the original protocol.

III. Security Implications

We discuss the security implications of our operations using the strand space model [2]. We give an informal description of the model here and refer the reader to [2] for formal definitions and details.

The strand space model is a model in which cryptographic protocols may be analyzed. In this model, the order in which the penetrator applies the operations available to him is restricted. There are certain components of messages which the penetrator cannot modify. These components are, therefore, a kind of authentication test: if the contents are later received in a modified form then a legitimate participant must have transformed them. Using this idea, authentication tests are developed which establish the extent of participation of the principals in a given protocol.

The model consists of *participants*, *messages* and *strands* as defined below.

Participants are either legitimate (called *regular* participants), or attackers, called *penetrators*. The set of messages is the set $\mathcal{M}$ of elements that can be sent between principals. This set is freely generated from two disjoint sets, the set T representing texts such as nonces or names, and the set K representing keys, by means of concatenation and encryption. The members of $\mathcal{M}$ are called *terms*. A *component* is a term which is not a concatenation of terms. A strand is a sequence of message transmissions and receptions, where the transmission and reception of a term t is represented by $+t$ and $-t$, respectively. A strand element is called a *node*. If s is a strand, $\langle s, i\rangle$ is the ith node on s. The notation $n \Rightarrow^{+} n'$ is used to mean that $n = \langle s, i \rangle$ and $n' = \langle s, j \rangle$ for some $j > i$. The *height* of a strand is the number of nodes in the strand. A term t *originates* at a node $n = \langle s, i \rangle$ if the sign of n is positive, and t is a subterm of n but not of any other earlier node. Similarly, a component t is *new* at n if t is a component of a term at n and not a component of any earlier nodes. When a component occurs new at a node but was a subterm of some previous node then the participant executing that strand must have done some cryptographic work to extract it as a new component. A *strand space* is a set of strands.

A strand represents the local view of a participant in a run of a protocol. For a legitimate participant it represents the messages that participant would send or receive as part of a run of his side of the protocol. Such a strand is called a *regular* strand.

The correctness of a protocol is analyzed as follows.

First, *safe keys* and *penetrable keys* are established. Intuitively, safe keys are keys that are not known to the penetrators and keys that never occur in any node unless encrypted under a safe key, while penetrable keys are either keys that are already known by a penetrator or keys that occur in some node in plaintext or encrypted under penetrable keys.

Then, using certain *tests*, which are segments of regular strands whose presence will guarantee the existence of other regular strands, the extent of involvement of the participants are established. The use of tests are embodied in three lemmas (called Authentication Tests) which we describe informally as follows.

1) Authentication Test 1: Suppose a component a, which does not occur earlier in a protocol run and which is encrypted under a key K not accessible to the penetrator, is transmitted and is later received in a new form (an *outgoing test*), then there must be a regular participant who can receive a and transmit it back transformed.
2) Authentication Test 2: Suppose a component a, which does not occur earlier in a protocol run, is transmitted and is later received encrypted under a key K not accessible to a penetrator (an *incoming test*), then there must be a regular participant who can receive a and send it back encrypted.
3) Authentication Test 3: If a component is received which cannot be generated by a penetrator (an *unsolicited test*), then a regular participant must have generated and sent it.

Using these tests, a regular participant may establish the extent to which the other regular participants have participated in the run of a given protocol. This gives a guarantee of the following form.

$$\left.\begin{array}{l}\text{If there is a regular strand } s_i \text{ of some height}\\ h_i \text{ with a particular set of parameters, then}\\ \text{there exists another regular strand } s_r \text{ of}\\ \text{some height } h_r \text{ with some set of parameters.}\end{array}\right\} \quad (*)$$

By examining the height of the strands and the sets of parameters, the regular participant may then establish the extent of the responder's participation and the actual achievement of a protocol. We refer the reader to [2] for a demonstration of how weaknesses of some protocols are uncovered using this method. In the remainder of this section, we will examine the effects of the deconstruction and reconstruction processes (operations (M1), (M2), and their inverses) on the authentication guarantees of a protocol.

First, we will show in Section III-A that an authentication guarantee of the form $(*)$ is preserved to some extent. Then, we will see in Section III-B that secrecy properties and tests are always preserved, and this allows us to obtain in a straightforward way authentication theorems of derived protocols from the authentication theorems of the original protocol.

A. Preservation of Authentication Guarantees

Let $\mathcal{P}$ be a three-party authentication protocol and let $(\mathcal{A}, <)$ be the initial poset associated with $\mathcal{P}$ as defined in Section II-A. Let $\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k\}$ be the set of protocols obtained from $(\mathcal{A}, <)$ by performing operations (M1) and (M2) in various orders. Then, $\mathcal{P}$ is one of $\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k\}$. Every $\mathcal{P}_i$ has the same associated initial poset $(\mathcal{A}, <)$, and $\mathcal{P}_j$ is obtained from $\mathcal{P}_i$ by performing a combination of operations (M1), (M2), and their inverses.

We make the assumption that all regular participants are essential to the protocol $\mathcal{P}$, in the sense that no participant acts as a sink (only a recipient) or a source (only a transmitter). This means that every regular strand in $\mathcal{P}$ has height at least two, and has at least one term of positive sign and one of negative sign.

Now, suppose that there is a regular strand s_X of some regular participant X, with a particular set of parameters A_X in $\mathcal{P}$. Consider the effect of a single operation of (M1) or (M2) on the height of s_X.

1) The operation (M1) merges the actions $a_1{:}\, P \rightarrow Q\ M_1$ and $a_2{:}\, P \rightarrow Q\ M_2$ if they are incomparable.

   If $X = P$ then the terms $+M_1$, $+M_2$ (which are part of two different positive terms) in s_X becomes part of a single positive term $+M_1M_2$. This decreases the height of s_X by at most one. Since s_X must have at least another term of negative sign, the height of the strand after a single such operation is still at least two. Since the components of the terms are not changed by the operation, the parameter set A_X remains the same.

   Similarly, if $X = Q$, the height of the strand after a single such operation is still at least two and the parameter set remains the same.
2) The operation (M2) merges the action $a_1{:}\, P \rightarrow Q\ M_1$ with the actions $a_2{:}\, P \rightarrow R\ M_2$ and $a_3{:}\, R \rightarrow Q\ M_3$ if $a_3 \not< a_2$ and a_1 is not comparable with either a_2 or a_3.

   By the same argument as above, the height of the strand after a single (M2) operation is still at least two and the parameter set remains the same for all the cases $X = P$, $X = Q$ and $X = R$.

It is also clear that the inverses of operations (M1) and (M2) either preserve or increase by one the height of s_X while leaving the parameter set unchanged. We conclude, therefore, that if $\mathcal{P}$ has a regular strand s_X with a particular set of parameters A_X and height at least two, then any protocol $\mathcal{P}'$ belonging to $\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k\}$ distinct from $\mathcal{P}$ also has a regular strand s'_X of the same participant with height at least two and the same parameter set.

Now let $\mathcal{P}''$ be any other protocol in $\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k\}$ distinct from $\mathcal{P}'$. Applying the same reasoning as above, $\mathcal{P}''$

also has a regular strand s''_X of the same participant, with the same parameter set and height at least two. Since $\mathcal{P}$ is one of $\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k\}$, we have the following.

Lemma III.1: For any distinct $\mathcal{P}_i$, $\mathcal{P}_j$ in the set $\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k\}$, $\mathcal{P}_i$ has a regular strand of participant X with parameter set A_X and height at least two if and only if $\mathcal{P}_j$ has a regular strand of the same participant with parameter set A_X and height at least two.

Suppose now that the following holds in $\mathcal{P}$.

$$\left.\begin{array}{l}\text{If there is a regular strand } s_X \text{ with parameter}\\ \text{set } A_X \text{ and height at least two, then there is}\\ \text{a regular strand } s_Y \text{ with parameter set } A_Y\\ \text{and height at least two.}\end{array}\right\} \quad (**)$$

Let $\mathcal{P}'$ be a protocol in $\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k\}$ distinct from $\mathcal{P}$ and suppose that there is a regular strand s'_X with parameter set A_X and height at least two in $\mathcal{P}'$. By Lemma III.1, this implies that there is a regular strand s_X with the same parameter set A_X and height at least two in $\mathcal{P}$. Since $(**)$ holds in $\mathcal{P}$, there is a regular strand s_Y with parameter set A_Y and height at least two in $\mathcal{P}$. This in turn implies the existence of the regular strand s'_Y in $\mathcal{P}'$. Hence, the following holds.

Theorem III.1: If the existence of a regular strand s_X with a particular parameter set A_X and height at least two implies the existence of a regular strand s_Y with parameter set A_Y and height at least two in $\mathcal{P}$, then the existence of a regular strand s'_X with parameter set A_X and height at least two of the same participant X in $\mathcal{P}'$ implies the existence of the regular strand s'_Y with parameter set A_Y and height at least two.

This shows that if participant X has a guarantee of involvement of participant Y in an execution of the original protocol, then X has a guarantee, to some extent, of the involvement of Y in any other version of the protocol. Section III-B describes how the exact guarantee may be obtained.

B. Derivation of Authentication Theorems

We will examine the effect of (M1), (M2) and their inverses on the secrecy and authentication properties of a protocol, and see how an authentication result of a protocol can be modified to be an authentication result of a protocol derived using these operations.

Firstly, consider the effects of (M1), (M2), and their inverses on secrecy properties. Since neither (M1), (M2), nor their inverses modifies a component, a safe (or penetrable) key remains so under the operations.

Secondly, consider the effects of (M1), (M2), and their inverses on the existence and signs of a component. It is straightforward to verify that operations (M1), (M2), and their inverses preserve both existence and sign of a component in a strand.

Now, we consider the effects of (M1), (M2), and their inverses on outgoing, incoming, and unsolicited tests.

The edge $n_0 \Rightarrow^+ n_1$ is an *outgoing test* for a in the component $t = [h]_K$ if

a) n_0 is a positive node where a first occurred and is transmitted. In addition, t is a component of n_0 and not a proper subterm of a component of any regular node in the strand space, and a is a subterm of t that does not occur in any component of n_0 other than t;

b) n_1 is a negative node where there is a new component t' in n_1 such that a is a subterm of t';

c) The inverse of the key K is not a penetrable key.

The edge $n_0 \Rightarrow^+ n_1$ is an *incoming test* for a in the component $t = [h]_K$ if

a) n_0 is a positive node where a first occurred and is transmitted;

b) n_1 is a negative node where t is a new component of n_1 and not a subterm of any component of any regular node in the strand space, and a is a subterm of t;

c) K is not a penetrable key.

A negative node n is an *unsolicited test* for the component $t = [h]_K$ if

a) K is not a penetrable key;

b) t is not a subterm of any component of any regular node in the strand space.

We show that the existence of these tests is preserved under (M1) and (M2).

Lemma III.2: Let $\mathcal{P}$ be a three-party authentication protocol and let $\mathcal{A}$ be the partially ordered set associated with it. Let $\mathcal{P}'$ be a protocol obtained by applying (M1), (M2) on $\mathcal{A}$. Consider a regular strand s in the strand space belonging to $\mathcal{P}$.

If there is an outgoing test $n_0 \Rightarrow^+ n_1$ for a in the component $t = [h]_K$ in the protocol $\mathcal{P}$, then there is an outgoing test $m_0 \Rightarrow^+ m_1$ for a in the component $t = [h]_K$ in the protocol $\mathcal{P}'$.

Proof: Suppose $n_0 \Rightarrow^+ n_1$ is an outgoing test for a in t in $\mathcal{P}$. Then a first occurred and is transmitted as a subterm of t and no other component. So for any actions

$$a_1\colon A \rightarrow X\ M_1, \quad a_2\colon A \rightarrow Y\ M_2$$

such that t is a component of M_1, and M_2 contains a as a subterm, we must have $a_1 < a_2$.

Let m_0 be the positive node where t first occurred and is transmitted in $\mathcal{P}'$. Such a node exists since (M1) and (M2) preserves a component and its sign. It also follows that a is a subterm of t and t is not a proper subterm of any component of any regular node in the strand space of $\mathcal{P}'$. Since the partial order is preserved, m_0 is also the first positive node where a first occurred and is transmitted. Since any other component containing a must occur after t, a does not belong to any other component in m_0 other than t.

Let m' be the earliest node in the strand space corresponding to $\mathcal{P}'$ containing the component t'. Since (M1), (M2) preserves signs and components, m' is a negative node and t' contains a as a subterm. Since partial order is preserved, m' cannot occur before m_0. Hence, $m_1 = m'$.

Since secrecy properties are preserved, the inverse of K remains inaccessible to the penetrator. Hence, $m_0 \Rightarrow^+ m_1$ is an outgoing test for a in t in $\mathcal{P}'$. □

Using a similar argument, we have the following.

Lemma III.3: Let $\mathcal{P}$, $\mathcal{P}'$ be defined as before. If there is an incoming test $n_0 \Rightarrow^+ n_1$ for a in the component $t = [h]_K$ in the protocol $\mathcal{P}$, then there is an incoming test $m_0 \Rightarrow^+ m_1$ for a in the component $t = [h]_K$ in the protocol $\mathcal{P}'$.

Lemma III.4: Let $\mathcal{P}$, $\mathcal{P}'$ be defined as before. If there is an unsolicited test n for the component $t = [h]_K$ in the protocol

$\mathcal{P}$, then there exists an unsolicited test m for the component $t = [h]_K$ in the protocol $\mathcal{P}'$.

Finally, we consider the effects of (M1) and (M2) on authentication properties. From above we see that outgoing, incoming and unsolicited tests are preserved under (M1) and (M2). From the proof of Lemma III.2, the nodes in the strand space corresponding to $\mathcal{P}'$ containing the relevant test components can be easily identified. Hence, any authentication theorem about protocol $\mathcal{P}$ is easily modified to be a theorem about protocol $\mathcal{P}'$.

As an example, authentication results for Protocol 1B can be obtained from the authentication results for Protocol 1A, with only a slight modification of the proofs. They are as follows, with the results for Protocol 1B in brackets.

a) If there is an initiator strand s_i of height 2 (4) with parameters A, B, N_A, and K, then there is a server strand s_s of height 2 (2) with the same parameters.
b) If there is a responder strand s_r of height 3 (3) with parameters A, B, N_B, and K, then there is a server strand s_s of height 2 (2) with the same parameters.
c) If there is an initiator strand s_i of height 2 (6) with parameters A, B, N_A, and K, then there is a responder strand s_r of height 4 (4) with the same parameters.
d) If there is a responder strand s_r of height 5 (3) with parameters A, B, N_A, N_B, and K, then there is an initiator strand s_i of height 3 (5) with the same parameters.

IV. Conclusion

We have given a framework in which to analyze the possibility of adapting a given protocol to suit a specific communication channel. We are, however, not advocating switching between the various protocols according to the behavior of the network at the time. For example, if the channel between B and S is down we might choose to use Protocol 1B which does not require communication between B and S. Such flexibility would seem to be practical in the less-than-ideal world of communication networks. However it is not clear if this lays the system open to attacks by penetrators interleaving different versions of a protocol. More work is needed in this area of mixing protocols.

There are possible extensions to our method. One possibility is to generalize the method to protocols involving more than three parties. Another possibility is to extend the method to include computations. In this paper, we have only concentrated on one level, that is, we dealt with only the higher level of sending and receiving messages rather than the computational level. We could extend the definition of an action to include computations such as generating a random number, encrypting or decrypting a message, as well as sending a message. This may help us in identifying critical actions, and in deciding whether to generate and store items, or generate at the last minute before sending, according to what sort of resources we have. On the other hand, we could assign weights on actions which depends on the components or the channels (the labels), for example, a weight could be a pair (time, money). We may then choose a protocol that is optimal according to our measure.

Acknowledgment

The authors would like to thank D. Gollmann, C. Mitchell, K. Paterson, and P. Wild for useful comments.

References

[1] L. Gong, "Lower bounds on messages and rounds for network authentication protocols," in *Proc. 1st ACM Conf. Computer and Communications Security*, Nov. 1993, pp. 26–37.
[2] J. D. Guttman and F. J. Thayer Fábrega, "Authentication tests," in *Proc. IEEE Symp. Security Privacy*, May 2000, pp. 96–109.
[3] P. J. Cameron, *Combinatorics: Topics, Techniques, Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1994.
[4] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, pp. 558–565, July 1978.
[5] R. Yahalom, "Optimality of asynchronous two party secure data-exchange protocols," *J. Comput. Security*, vol. 2, pp. 191–209, 1993.

Siaw-Lynn Ng received the B.Sc. (honors) degree in mathematics from the University of Adelaide, Australia, in 1995, and the Ph.D. degree in mathematics from Royal Holloway, University of London, U.K., in 1999.

She was a Post-Doctoral Research Assistant and later a Lecturer at Royal Holloway, University of London. Her main research interests are in combinatorics, finite geometry, and information security