A Lesson on Authentication Protocol Design*

Thomas Y.C. Woo   Simon S. Lam
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

1 Introduction

The purpose of this note is to describe a useful lesson we learned on authentication protocol design. In a recent article [9], we presented a simple authentication protocol to illustrate the concept of a trusted server. The protocol has a flaw, which was brought to our attention by Martín Abadi of DEC.

In what follows, we first describe the protocol and its flaw, and how the flaw was introduced in the process of deriving the protocol from its correct *full information* version. We then introduce a principle, called the *Principle of Full Information*, and explain how its use could have prevented the protocol flaw. We believe the Principle of Full Information is a useful authentication protocol design principle, and advocate its use. Lastly, we present several heuristics for simplifying full information protocols and illustrate their application to a mutual authentication protocol.

2 A Problematic Protocol

The following protocol appears in [9, pp. 42–43].[1] It performs only one-way authentication, authenticating a principal P to another principal Q. We call P and Q respectively the *initiator* and *responder* of the protocol.

*Research supported in part by NSA INFOSEC University Research Program under contract no. MDA 904-91-C7046 and MDA 904-93-C4089, and in part by National Science Foundation grant no. NCR-9004464. Published in *ACM Operating Systems Review*, Vol. 28, No. 3, July 1994. Postscript files of this and other papers of the Networking Research Laboratory are available from http://www.cs.utexas.edu/~lam/NRL.

[1]The protocol specification in [9, pp. 42–43] is more verbose. We have omitted some obvious internal steps here.

In what follows, $\{m\}_k$ denotes the encryption of m by k while "," denotes the concatenation operator. A is a trusted authentication server which shares a secret key k_{XA} with each principal X in the system. A *nonce* is some information that is *fresh*, i.e., has never appeared before. We refer to the following protocol as $\mathbf{\Pi}$.

$$
\begin{array}{llcl}
(1) & P \rightarrow Q & : & \text{“I am } P\text{.”} \\
(2) & Q & : & \text{generate nonce } n \\
(3) & Q \rightarrow P & : & n \\
(4) & P \rightarrow Q & : & \{n\}_{k_{PA}} \\
(5) & Q \rightarrow A & : & \{P, \{n\}_{k_{PA}}\}_{k_{QA}} \\
(6) & A \rightarrow Q & : & \{n\}_{k_{QA}}
\end{array}
$$

$\mathbf{\Pi}$ was invented by us to illustrate the use of a trusted server for *key translation*, a process that takes as input an encrypted message and generates as output a new message with the same content as the input but encrypted using a different key. Key translation is mainly used in a shared-key system, and is typically carried out in such a system by a centralized server that shares a secret key with each principal in the system. Specifically, in the above protocol, A "translates" the message $\{n\}_{k_{PA}}$ sent by P in step (4) into the message $\{n\}_{k_{QA}}$ in step (6), which can be understood by Q.

Protocol $\mathbf{\Pi}$ is said to be *correct* if whenever a responder finishes execution of the protocol, the initiator of the protocol execution is in fact the principal claimed in step (1) of the protocol.

However, $\mathbf{\Pi}$ turns out to be incorrect. This is illustrated by the following scenario. Z denotes a saboteur, who also happens to be a legitimate user in the system. In particular, it shares a key k_{ZA} with A and can initiate authentication exchanges. The label in each line below identifies both the protocol step and the specific authentication exchange the step belongs to. For example, (QR.3) refers to step (3) in an authentication exchange between Q (the initiator) and R (the responder). In a communication step, the principal in parentheses denotes the intended sender or receiver of the message. G denotes some "gibberish" generated by Z.

$$
\begin{array}{llll}
\text{(PQ.1)} & (P)Z \rightarrow Q & : & \text{“I am } P\text{.”} \\
\text{(PQ.2)} & Q & : & \text{generate nonce } n \\
\text{(PQ.3)} & Q \rightarrow (P)Z & : & n \\
\text{(PQ.4)} & (P)Z \rightarrow Q & : & G \\
\text{(PQ.5)} & Q \rightarrow (A)Z & : & \{P, G\}_{k_{QA}}
\end{array}
$$

Z waits until Q initiates an authentication exchange with some principal R.

$$
\begin{array}{llll}
\text{(QR.1)} & Q \rightarrow (R)Z & : & \text{“I am } Q\text{.”} \\
\text{(QR.3)} & (R)Z \rightarrow Q & : & Z, \{n\}_{k_{ZA}} \\
\text{(QR.4)} & Q \rightarrow (R)Z & : & \{Z, \{n\}_{k_{ZA}}\}_{k_{QA}} \\
\text{(QZ.5)} & (Q)Z \rightarrow A & : & \{Z, \{n\}_{k_{ZA}}\}_{k_{QA}} \\
\text{(QZ.6)} & A \rightarrow (Q)Z & : & \{n\}_{k_{QA}} \\
\text{(PQ.6)} & (A)Z \rightarrow Q & : & \{n\}_{k_{QA}}
\end{array}
$$

The authentication of P by Q in (PQ.1)–(PQ.6) succeeds even though Z, and not P, is the initiator in step (PQ.1). That is, Z is able to masquerade as P to Q. The cause of this failure can be attributed to the fact that the reply message from A to Q in step (6) does not carry information to identify the initiator of an exchange.

The above attack requires Q to initiate an outbound authentication when it is waiting for the inbound authentication from P to complete. Another attack scenario, suggested by Martín Abadi [1], does not require this. But it does require two concurrent inbound authentications: one presumably between P and Q (identified by label PQ), and the other between Z and Q (identified by label ZQ).

$$
\begin{array}{llll}
\text{(PQ.1)} & (P)Z \rightarrow Q & : & \text{“I am } P\text{.”} \\
\text{(ZQ.1)} & Z \rightarrow Q & : & \text{“I am } Z\text{.”} \\
\text{(PQ.2)} & Q & : & \text{generate nonce } n_P \\
\text{(ZQ.2)} & Q & : & \text{generate nonce } n_Z \\
\text{(PQ.3)} & Q \rightarrow (P)Z & : & n_P \\
\text{(ZQ.3)} & Q \rightarrow Z & : & n_Z \\
\text{(PQ.4)} & (P)Z \rightarrow Q & : & G \\
\text{(ZQ.4)} & Z \rightarrow Q & : & \{n_P\}_{k_{ZA}} \\
\text{(PQ.5)} & Q \rightarrow A & : & \{P, G\}_{k_{QA}} \\
\text{(ZQ.5)} & Q \rightarrow A & : & \{Z, \{n_P\}_{k_{ZA}}\}_{k_{QA}} \\
\text{(ZQ.6)} & A & : & \text{abort} \\
\text{(PQ.6)} & A \rightarrow Q & : & \{n_P\}_{k_{QA}}
\end{array}
$$

Again, authentication of P by Q in (PQ.1)–(PQ.6) succeeds even though P did not participate in the exchange. The cause of this failure is the inability of the responder (Q in this case) to distinguish messages in concurrent invocations of the protocol.

This in turn can be attributed to the fact that insufficient information is carried in the reply message from A to Q in step (6).

Interestingly, the above attack can be strengthened such that no error is detected by A; instead, either Q would detect an error or a time-out would occur at Q. To achieve this, Z must have recorded a message of the form $\{n_{old}\}_{k_{PA}}$ from (step (4) of) a previous (successful) authentication of P by Q using $\mathbf{\Pi}$, where n_{old} is the nonce generated by P in that exchange. If Z replays $\{n_{old}\}_{k_{PA}}$ in step (PQ.4) in place of G, then the checking at step (ZQ.6) by A would succeed with a reply of $\{n_{old}\}_{k_{QA}}$ to Q. If Z intercepts this reply, a time-out would occur at Q. If the reply gets through to Q, it would be rejected as an error.

3 The Original Protocol

In this section, we retrace the design process of $\mathbf{\Pi}$ and try to identify where and how the flaw illustrated in Section 2 was introduced.

The first protocol we came up with to illustrate the idea of key translation (in the context of one-way authentication) is the following:

$$
\begin{array}{llcl}
(1) & P \rightarrow Q & : & \text{“I am } P\text{.”} \\
(2) & Q & : & \text{generate nonce } n \\
(3) & Q \rightarrow P & : & n \\
(4) & P \rightarrow A & : & P, Q, \{P, Q, n\}_{k_{PA}} \\
(5) & A \rightarrow P & : & \{P, Q, n\}_{k_{QA}} \\
(6) & P \rightarrow Q & : & \{P, Q, n\}_{k_{QA}}
\end{array}
$$

In this protocol, key translation is requested by the initiator (i.e., P) before it sends out its message to the responder (i.e., Q). An "equivalent" protocol in which the key translation request is made by the responder is shown below:[2]

$$
\begin{array}{llcl}
(1) & P \rightarrow Q & : & \text{“I am } P\text{.”} \\
(2) & Q & : & \text{generate nonce } n \\
(3) & Q \rightarrow P & : & n \\
(4) & P \rightarrow Q & : & \{P, Q, n\}_{k_{PA}} \\
(5) & Q \rightarrow A & : & P, Q, \{P, Q, n\}_{k_{PA}} \\
(6) & A \rightarrow Q & : & \{P, Q, n\}_{k_{QA}}
\end{array}
$$

Although the above protocols are theoretically correct, they are susceptible to *chosen plaintext attack*. Specifically, a saboteur Z can obtain encrypted messages of

[2]These two protocols represent respectively the well-known *push* and *pull* models [12].

the form $\{Z, X, m\}_{k_{XA}}$ for any principal X and any message m of its choice.[3] This is because a key translation request can be made by any principal independent of the knowledge or consent of the other principal.

To counter this attack, we modified the protocol to require an additional level of encryption in the key translation request. The resulting protocol, $\mathbf{\Pi}^f$, is shown below:

(1)	$P \rightarrow Q$	:	"I am P."
(2)	Q	:	generate nonce n
(3)	$Q \rightarrow P$	:	n
(4)	$P \rightarrow Q$	:	$\{P, Q, n\}_{k_{PA}}$
(5)	$Q \rightarrow A$	:	$\{P, Q, n, \{P, Q, n\}_{k_{PA}}\}_{k_{QA}}$
(6)	$A \rightarrow Q$	:	$\{P, Q, n\}_{k_{QA}}$

Essentially, chosen plaintext attack is curtailed by using encryption to represent an implicit consent to the key translation. More specifically, in the message sent in step (5), the inner encryption by k_{PA} and the outer encryption by k_{QA} represent respectively P and Q's consent to the key translation request. A verifies both encryption (i.e., "proof of consent" from both parties) before it would carry out the key translation.

While $\mathbf{\Pi}$ is shown to be incorrect in Section 2, $\mathbf{\Pi}^f$ can be formally shown to be correct [11]. (The precise notion of correctness adopted is discussed in Appendix A.)

4 The Making of $\mathbf{\Pi}$

$\mathbf{\Pi}^f$ was then simplified as described in the following because we thought that the simplified protocols would be clearer and more efficient. First from $\mathbf{\Pi}^f$, we deleted the first occurrence of n from the message in step (5) to obtain protocol $\mathbf{\Pi}^1$ below:

(1)	$P \rightarrow Q$	:	"I am P."
(2)	Q	:	generate nonce n
(3)	$Q \rightarrow P$	:	n
(4)	$P \rightarrow Q$	:	$\{P, Q, n\}_{k_{PA}}$
(5)	$Q \rightarrow A$	:	$\{P, Q, \{P, Q, n\}_{k_{PA}}\}_{k_{QA}}$
(6)	$A \rightarrow Q$	:	$\{P, Q, n\}_{k_{QA}}$

From $\mathbf{\Pi}^1$, we deleted all occurrences of Q from the encrypted messages in steps (4) to (6), and obtained protocol $\mathbf{\Pi}^2$ below:

[3] Strictly speaking, this is a restricted chosen plaintext attack as Z has no control over the entire plaintext.

$$
\begin{array}{llll}
(1) & P \rightarrow Q & : & \text{"I am } P\text{."} \\
(2) & Q & : & \text{generate nonce } n \\
(3) & Q \rightarrow P & : & n \\
(4) & P \rightarrow Q & : & \{P, n\}_{k_{PA}} \\
(5) & Q \rightarrow A & : & \{P, \{P, n\}_{k_{PA}}\}_{k_{QA}} \\
(6) & A \rightarrow Q & : & \{P, n\}_{k_{QA}}
\end{array}
$$

Then, from $\mathbf{\Pi}^2$, we obtained $\mathbf{\Pi}^3$ below by deleting P from the message in step (4) (and hence also from the innermost encryption of the message in step (5)):

$$
\begin{array}{llll}
(1) & P \rightarrow Q & : & \text{"I am } P\text{."} \\
(2) & Q & : & \text{generate nonce } n \\
(3) & Q \rightarrow P & : & n \\
(4) & P \rightarrow Q & : & \{n\}_{k_{PA}} \\
(5) & Q \rightarrow A & : & \{P, \{n\}_{k_{PA}}\}_{k_{QA}} \\
(6) & A \rightarrow Q & : & \{P, n\}_{k_{QA}}
\end{array}
$$

And finally we obtained $\mathbf{\Pi}$ in Section 2 by deleting P from the message in step (6).

Similar to $\mathbf{\Pi}^f$, each of $\mathbf{\Pi}^1$, $\mathbf{\Pi}^2$ and $\mathbf{\Pi}^3$ can be formally shown to be correct. Indeed, the correctness proofs for these protocols are similar and involve showing that messages of certain forms cannot be generated by a saboteur. In other words, the deletion of P from the message in step (6) was the fatal mistake: it reduced the "information content" of the message beyond the acceptable limit.

In retrospect, it is clear that the simplification of $\mathbf{\Pi}^f$ to $\mathbf{\Pi}$is not worthwhile. Specifically, the benefits we gain (i.e., slightly shorter messages together with slightly less encryption) are insignificant. However, as shown in Section 2, such simplification would result in protocol error if it is not carried out carefully.

We make two interesting observations. First, if we delete all occurrences of P instead of Q from the encrypted messages of $\mathbf{\Pi}^f$, the resulting protocol would be incorrect. Specifically, the same attack as outlined in Section 2 would apply. This, together with the fact that $\mathbf{\Pi}^f$ is correct, suggest that the roles of the initiator and responder are asymmetric. Therefore if the protocol is extended to perform mutual authentication, neither the name of the initiator nor the name of the responder can be deleted.

Second, if nonces are implemented by tagging a fresh number with the names of its generator and intended receiver,[4] then $\mathbf{\Pi}$ would be "equivalent" to $\mathbf{\Pi}^f$,[5] and hence correct.

[4]This ensures that nonces are globally unique.

[5]Assuming that proper checking is performed.

5 Principle of Full Information

An authentication protocol is said to be *full information* if its initiator and responder[6] include in every outgoing encrypted message all of the information each has gathered so far in the authentication exchange.

For example, $\mathbf{\Pi}^f$ in Section 3 is full information. Specifically, in step (4), the information P (the initiator) has gathered regarding the ongoing authentication consists of the names of both participating principals and the nonce n to be used for the authentication, and it has included them in its outgoing message. Similarly, in step (5), Q (the responder) knows the names of both participating principals, the nonce n, and the reply $\{P, Q, n\}_{k_{PA}}$ it has just received from P, and it has included all such information in its outgoing message. The simplified protocols $\mathbf{\Pi}^1$, $\mathbf{\Pi}^2$, $\mathbf{\Pi}^3$ and $\mathbf{\Pi}$are, however, not full information.

From the discussion in Section 3, we see that the full information version of an authentication protocol can be more resistant to attacks than its simplified variants. Indeed, this is generally true from our experience. Intuitively, this can be explained as follows: Messages in a full information protocol carry more information than their simplified versions. Because of the extra information, these messages are more specific to a particular authentication exchange run, and hence are less likely to be exploited by a saboteur in compromising another run. In addition, more thorough checking using the extra information can be performed by the recipients, thus reducing the likelihood of successful attacks.

We believe that authentication protocols should be designed to be full information. We call this the *Principle of Full Information.* Strict adherence to this principle could have eliminated many of the flaws in published protocols, including our own. Simplification of full information protocols should be attempted only if it represents major savings, either in message length or in message processing time, and must be performed with extreme caution. For example, the simplification from $\mathbf{\Pi}^f$ to $\mathbf{\Pi}$ does not represent major savings and should not have been carried out.

6 Heuristics

When it is deemed appropriate to simplify a full information protocol, we present in the following three heuristics, together with a brief explanation of the intuition behind, that can be of use. We caution, however, that these heuristics are informal and do not necessarily preserve correctness. They are intended as guidelines for

[6]We observe that most authentication protocols typically involve at most three principals: an initiator, a responder and a server. In this paper, we do not consider protocols that have multiple initiators and responders.

performing simplification; their application does not obviate the need for a formal verification of the simplified protocol.

Heuristics 1

> Each message should contain the names of both the initiator and the responder. It should also contain at least the nonce of the intended recipient, if not both the nonces of the initiator and responder.

Intuitively, the quadruple

$$\langle \text{ initiator name, responder name, initiator nonce, responder nonce } \rangle$$

uniquely identifies an authentication run. Hence, if a message includes this quadruple, it would be unique to a particular run, and cannot be exploited by a saboteur for use in another run. The presence of the recipient's nonce allows the recipient to verify the timeliness of the message.

Heuristics 2

> In a protocol based on a symmetric cryptosystem, an encrypted message intended for a principal X (i.e., encrypted using X's secret key) does not need to include the name of X, but should include the name of the other party.

If a message is encrypted under X's secret key, it must either be generated by X or intended for X. In either case, only X's key can be used to recover the content of the message. Hence, the name of X can be implicitly understood to be present.

Heuristics 3

> Encrypted replies from one of the parties need not be nested inside encrypted messages of the other party.

That is, a nested message of the form "$\{M\}_{k_{XA}}, \{M'\}_{k_{YA}}$" can often achieve a desired effect as the message "$\{M', \{M\}_{k_{XA}}\}_{k_{YA}}$". In both cases, "$\{M\}_{k_{XA}}$" is being forwarded by one of the principals (Y) to the other principal (X). The main difference between the two cases is that in the latter, when X receives "$\{M\}_{k_{XA}}$," it can infer that Y must have received "$\{M', \{M\}_{k_{XA}}\}_{k_{YA}}$" earlier. In particular, if M contains similar information as M', then X can infer that Y must also be aware of similar information.

We illustrate application of these heuristics in the next section with an example.

7 Mutual Authentication

In general, mutual authentication may not be achieved if we simply run a one-way authentication protocol twice, once in each direction. This is because messages communicated in one run could be exploited by a saboteur to compromise the run in the other direction. (See [3] for an example.) The situation may improve, however, if the one-way protocol is full information. We illustrate this using $\mathbf{\Pi}^f$.

Consider the following mutual authentication protocol obtained by "composing" two instances of $\mathbf{\Pi}^f$, where each instance represents one-way authentication in one direction:

$$
\begin{array}{llcl}
(1) & P & : & \text{generate nonce } n_P \\
(2) & P \rightarrow Q & : & \text{“I am } P.\text{”}, n_P \\
(3) & Q & : & \text{generate nonce } n_Q \\
(4) & Q \rightarrow P & : & \text{“I am } Q.\text{”}, n_Q, \{P, Q, n_P\}_{k_{QA}} \\
(5) & P \rightarrow Q & : & \{P, Q, n_Q\}_{k_{PA}}, \{P, Q, n_P, \{P, Q, n_P\}_{k_{QA}}\}_{k_{PA}} \\
(6) & Q \rightarrow A & : & \{P, Q, n_Q, \{P, Q, n_Q\}_{k_{PA}}\}_{k_{QA}}, \{P, Q, n_P, \{P, Q, n_P\}_{k_{QA}}\}_{k_{PA}} \\
(7) & A \rightarrow Q & : & \{P, Q, n_P\}_{k_{PA}}, \{P, Q, n_Q\}_{k_{QA}} \\
(8) & Q \rightarrow P & : & \{P, Q, n_P\}_{k_{PA}}
\end{array}
$$

The above protocol is correct in the sense that if P finishes executing its part of the protocol, it believes that it has been talking to Q, and vice versa for Q.

The above protocol is, however, not full information. For example, n_Q is not included in Q's outgoing message in step (4) even though it concerns the current authentication exchange and has already been generated by Q in step (3).

Fortunately, only slight changes are required to obtain a full information version of the above protocol. We show it below:

$$
\begin{array}{llcl}
(1) & P & : & \text{generate nonce } n_P \\
(2) & P \rightarrow Q & : & \text{“I am } P.\text{”}, n_P \\
(3) & Q & : & \text{generate nonce } n_Q \\
(4) & Q \rightarrow P & : & \text{“I am } Q.\text{”}, n_Q, \{P, Q, n_P, n_Q\}_{k_{QA}} \\
(5) & P \rightarrow Q & : & \{P, Q, n_P, n_Q\}_{k_{PA}}, \{P, Q, n_P, n_Q, \{P, Q, n_P, n_Q\}_{k_{QA}}\}_{k_{PA}} \\
(6) & Q \rightarrow A & : & \{P, Q, n_P, n_Q, \{P, Q, n_P, n_Q\}_{k_{PA}}, \\
 & & : & \quad \{P, Q, n_P, n_Q, \{P, Q, n_P, n_Q\}_{k_{QA}}\}_{k_{PA}}\}_{k_{QA}} \\
(7) & A \rightarrow Q & : & \{P, Q, n_P, n_Q\}_{k_{PA}}, \{P, Q, n_P, n_Q\}_{k_{QA}} \\
(8) & Q \rightarrow P & : & \{P, Q, n_P, n_Q\}_{k_{PA}}
\end{array}
$$

There are many redundancies in the above protocol, especially in the messages communicated in steps (5) and (6). Thus we can try applying the three heuristics in Section 6 to see if a simplified yet correct protocol can be obtained.

We enumerate our simplification steps below: (For brevity, we refer to the message communicated in step (i) as message (i).)

Step 1 Using heuristics 3, we simplify the second component of message (5) to

$$\{P, Q, n_P, n_Q\}_{k_{PA}}, \{P, Q, n_P, n_Q\}_{k_{QA}}$$

Since the first component of message (5) is already contained in the above, we can eliminate the first component from message (5).

Step 2 With the simplification in Step 1, it is easy to observe that there is no need for Q to include $\{P, Q, n_P, n_Q\}_{k_{QA}}$ in its message to P in step (4). Thus, we eliminate $\{P, Q, n_P, n_Q\}_{k_{QA}}$ from message (4).

Step 3 Following Step 2, if $\{P, Q, n_P, n_Q\}_{k_{QA}}$ is eliminated from message (4), it will also have to be eliminated from message (5).

Step 4 Using heuristics 3 repeatedly, we can eliminate the nested encryption in message (6). The new message (6) is

$$\{P, Q, n_P, n_Q\}_{k_{PA}}, \{P, Q, n_P, n_Q\}_{k_{QA}}$$

Step 5 To make message (7) different from message (6),[7] we can use heuristics 2 to eliminate one of the names from each component of message (7).

The resulting protocol is shown below:

(1)	P	:	generate nonce n_P
(2)	$P \to Q$	:	"I am P.", n_P
(3)	Q	:	generate nonce n_Q
(4)	$Q \to P$	:	"I am Q.", n_Q
(5)	$P \to Q$	:	$\{P, Q, n_P, n_Q\}_{k_{PA}}$
(6)	$Q \to A$	:	$\{P, Q, n_P, n_Q\}_{k_{PA}}$, $\{P, Q, n_P, n_Q\}_{k_{QA}}$
(7)	$A \to Q$	:	$\{Q, n_P, n_Q\}_{k_{PA}}$, $\{P, n_P, n_Q\}_{k_{QA}}$
(8)	$Q \to P$	:	$\{Q, n_P, n_Q\}_{k_{PA}}$

This turns out to be a correct (albeit not most message efficient) mutual authentication protocol and can be easily extended to carry out key distribution as follows:

[7] This is not strictly necessary if we make the assumption that a principal can recognize and discard its own messages. See Appendix A.

$$
\begin{array}{rlcl}
(1) & P & : & \text{generate nonce } n_P \\
(2) & P \rightarrow Q & : & \text{“I am } P\text{.”}, n_P \\
(3) & Q & : & \text{generate nonce } n_Q \\
(4) & Q \rightarrow P & : & \text{“I am } Q\text{.”}, n_Q \\
(5) & P \rightarrow Q & : & \{P, Q, n_P, n_Q\}_{k_{PA}} \\
(6) & Q \rightarrow A & : & \{P, Q, n_P, n_Q\}_{k_{PA}}, \{P, Q, n_P, n_Q\}_{k_{QA}} \\
(7) & A & : & \text{generate session key } k \\
(8) & A \rightarrow Q & : & \{Q, n_P, n_Q, k\}_{k_{PA}}, \{P, n_P, n_Q, k\}_{k_{QA}} \\
(9) & Q \rightarrow P & : & \{Q, n_P, n_Q, k\}_{k_{PA}}, \{n_P, n_Q\}_k \\
(10) & P \rightarrow Q & : & \{n_Q\}_k
\end{array}
$$

8 Conclusion

We have learned a valuable lesson. While it is possible to simplify an authentication protocol by selectively simplifying encrypted messages, it is not always wise to do so. Besides, shortening a message may not offer much efficiency improvement for the class of authentication protocols considered in this paper.[8] To obtain significant improvements in efficiency, an authentication protocol designer should instead focus on reducing the number of messages in a protocol.

We have introduced the Principle of Full Information. This principle, if followed in our protocol, would have avoided the protocol mistake illustrated in Section 2. We advocate that authentication protocols be designed using encrypted messages that carry full information. From our experience, these protocols tend to be more resistant to attacks than their simplified variants, and are easier to be proved correct.

Our Principle of Full Information can also be viewed as a concrete implementation of Principle 1 in [2]. Specifically, full information specifies that each message should carry as much information as possible regarding the current authentication run, to the extent that it becomes self-contained and uniquely identifiable as belonging to a particular authentication run. Attacks resulting from messages that do not carry full information have also been demonstrated in [7] and [8].

We have also proposed several heuristics for simplifying full information protocols. Although these are informal, we believe they represent useful directions in simplifying authentication protocols and a good first step toward a more formal stepwise refinement procedure.

[8]Authentication protocols intended for lower protocol layers or for connectionless communication may benefit from such simplification. However, even for these protocols, the principle of full information should still be observed implicitly. For example, in [3], a low level authentication protocol is developed in which the names of participating principals are encoded implicitly using the exclusive-or operation.

Notions similar to that of a full information protocol have also been used in other context. For example, the first protocol proposed for solving the distributed consensus problem was "full information" [6]. This protocol was then successively simplified to obtain more efficient solutions.

Acknowledgment

We are grateful to Martín Abadi of DEC for bringing to our attention the error in $\mathbf{\Pi}$ and for comments on this note.

References

[1] M. Abadi. Private communication. March 1993.

[2] M. Abadi and R. Needham. Good engineering practice for security in cryptographic protocols. Manuscript, November 1993.

[3] R. Bird, I. Gopal, A. Herzberg, P.A. Janson, S. Kutten, R. Molva, and M. Yung. Systematic design of a family of attack-resistant authentication protocols. *IEEE Journal on Selected Areas in Communications*, 11(5):679–693, June 1993.

[4] M. Burrows, M. Abadi, and R.M. Needham. A logic of authentication. Technical Report 39, Systems Research Center, Digital Equipment Corporation, February 28 1989. Revised February 22, 1990. An abbreviated version appears in [5].

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

[6] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[7] E. Snekkenes. Roles in cryptographic protocols. In *Proceedings of the 13th IEEE Symposium on Research in Security and Privacy*, pages 105–119, Oakland, California, May 4–6 1992.

[8] P. Syverson. On key distribution protocols for repeated authentication. *ACM Operating Systems Review*, 27(3):24–30, October 1993.

[9] T.Y.C. Woo and S.S. Lam. Authentication for distributed systems. *Computer*, 25(1):39–52, January 1992. See also "Authentication" revisited. *Computer*, 25(3):10–10, March 1992.

[10] T.Y.C. Woo and S.S. Lam. A semantic model for authentication protocols. In *Proceedings of the 14th IEEE Symposium on Research in Security and Privacy*, pages 178–194, Oakland, California, May 24–26 1993.

[11] T.Y.C. Woo and S.S. Lam. Verifying authentication protocols: Methodology and example. In *Proceedings of the International Conference on Network Protocols*, pages 36–45, San Francisco, California, October 19–22 1993.

[12] CCITT Recommendation X.500 The Directory—Overview of concepts, models, and services, 1988.

A Correctness

For the authentication protocols presented in Section 3, our notion of correctness can be informally stated as follows: An authentication protocol is *correct* if for all its executions, whenever an authenticating principal (also the responder) completes an authentication exchange, it must be the case that the authenticated principal (also the initiator) has initiated the exchange earlier.[9]

More concretely, the completion of an authentication exchange by an authenticating principal X can be equated with the receipt of the message in step (5) by X from A; while the initiation of an authentication exchange by an authenticated principal Y can be interpreted as the sending of either the message in step (1) or the message in step (4) by Y.

Take protocol $\mathbf{\Pi}^f$ as an example. The above criterion says $\mathbf{\Pi}^f$ is correct if for all its executions, whenever Q receives a message $\{P, Q, n\}_{k_{QA}}$ in step (6), it must be the case that P has sent message $\{P, Q, n\}_{k_{PA}}$ in step (4) earlier in the execution.

This notion of correctness can be formalized using a *correspondence* property. Roughly speaking, a correspondence property specifies a one-to-one temporal relationship between the occurrences of two transitions. A formal definition of correspondence properties is presented in [10]. Using the proof procedure proposed therein for correspondence properties, we can show that $\mathbf{\Pi}^f$, $\mathbf{\Pi}^1$ and $\mathbf{\Pi}^2$ satisfy the correctness criterion stated in the previous paragraph.

It should, however, be noted that an extra assumption is needed in the verification of $\mathbf{\Pi}^1$ and $\mathbf{\Pi}^2$. That is, a principal is assumed to be able to recognize and hence discard its own messages. In other words, a naive replay of a principal X's outbound messages back to X will be detected and hence ignored. The same assumption is also postulated in BAN logic [4, p. 5]. Without this assumption, protocol $\mathbf{\Pi}^1$ can fail as follows:

[9] In a mutual authentication protocol, this same notion applies except that a principal is an authenticating principal in one direction and an authenticated principal in the other direction.

$$\begin{array}{lll} (P)Z \rightarrow Q & : & \text{“I am } P\text{.”} \\ Q & : & \text{generate nonce } n \\ Q \rightarrow P & : & n \\ (P)Z \rightarrow Q & : & n \\ Q \rightarrow (A)Z & : & \{P, Q, n\}_{k_{QA}} \\ (A)Z \rightarrow Q & : & \{P, Q, n\}_{k_{QA}} \end{array}$$

Similar scenarios can be constructed for $\mathbf{\Pi}^2$. Interestingly, $\mathbf{\Pi}^f$ and $\mathbf{\Pi}$ are not vulnerable to this attack.

From a practical standpoint, the assumption can be understood as follows. The secret key k_{XA} shared between principal X and server A can be viewed as representing two distinct keys k^{in}_{XA} and k^{out}_{XA}, each of which is easily derivable from k_{XA}.[10] For X, k^{in}_{XA} is used whenever it wants to decrypt an incoming message that is supposedly encrypted by k_{XA}, and k^{out}_{XA} is used whenever it wants to send a message that is supposedly to be encrypted by k_{XA}. The opposite convention is followed by A; that is, it uses k^{in}_{XA} to send and k^{out}_{XA} to receive.[11]

With this assumption, protocol $\mathbf{\Pi}^1$ can be more accurately specified as:

$$\begin{array}{llll} (1) & P \rightarrow Q & : & \text{“I am } P\text{.”} \\ (2) & Q & : & \text{generate nonce } n \\ (3) & Q \rightarrow P & : & n \\ (4) & P \rightarrow Q & : & \{P, Q, n\}_{k^{out}_{PA}} \\ (5) & Q \rightarrow A & : & \{P, Q, \{P, Q, n\}_{k^{out}_{PA}}\}_{k^{out}_{QA}} \\ (6) & A \rightarrow Q & : & \{P, Q, n\}_{k^{in}_{QA}} \end{array}$$

And the correctness criterion stated above can be more accurately rephrased as follows: Protocol $\mathbf{\Pi}^1$ is correct if for all its executions, whenever Q receives a message $\{P, Q, n\}_{k^{in}_{QA}}$ in step (6), it must be the case that P has sent message $\{P, Q, n\}_{k^{out}_{PA}}$ in step (4) earlier in the execution.

[10] One simple way to achieve this is to define: $k^{in}_{XA} = k_{XA} \oplus X$ and $k^{out}_{XA} = k_{XA} \oplus A$ where $\oplus$ denotes the exclusive-or operation.

[11] In fact, this can be made symmetric if we postulate that $k_{XA} = k_{AX}$, $k^{in}_{AX} = k^{out}_{XA}$ and $k^{out}_{AX} = k^{in}_{XA}$. This, for example, is satisfied in the definition given in the above footnote.