A Sufficient Condition for Secure Ping–Pong Protocols

Masao Mori
Department of Informatics, Kyushu University, Fukuoka 33, 812–8581, Japan
masa@i.kyushu-u.ac.jp

September 21, 2001

Abstract

A sufficient condition for secure ping–pong protocols is repretsented. This condition, called *name–suffixing*, is essentially to insert identities of participants in messages. We prove its sufficiency and discuss the feature of security in terms of name–suffixing.

Keywords: Cryptography; Verification of cryptographic protocols

1 Introduction

The state machine approach verifying cryptographic protocols by Dolev and Yao [5], and Dolev, Even and Karp [4], however it is aimed at a simple type of cryptographic protocols, called *ping–pong protocols*, has been a fundamental model of some contemporary verifying techniques [8], namely NRL Protocol Analyzer and Interrogator etc. In their approach the state machines are designed to accept strings of cryptographic operations which express both legitimate executions of protocols and sabotuers' devices. Verification is done by an algorithm seeking a binary relation of states which indicates vulnerability of cryptographic protocols.

That algorithm by Dolev et al. does not give any advice to revise vulnerable protocols but only verifies them. Designing cryptographic protocols some guiding principles for security, such as [2], is required. In this paper we represent a sufficient condition, called *name–suffixing*, and give its proof in order to design secure ping–pong protocols. This condition is simply to insert identities of participants in messages . The similar our result has been found by Lowe [6] who points out a vulnerability of the Needham–Schroeder protocol and gives its correction. That is essentially to include participants identities in messages. Insertion of identities in messages is an instance of Principle 3 in [2].

In addition we will introduce examples in which insertion of identities is more effective rather than digital signatures. Sabotures are supposed to accomplish attacking without being noticed by anyone. We have examples which is determined to be vulnerable in the sence of [4] but such smart attack has not been found for those examples. In section 5 we mention not only the above discussion but also the way to consider secure conditions for more contemporary cryptographic protocols with respect to *name–insertion*.

2 Preliminary

Names of *legitimate participants*, *initiator* and *responder*, are denoted by A and B respectively which belong to the set $\{0,1\}^*$ of finite bit–strings. The name $S \in \{0,1\}^*$ denotes the *sabotuer* who can participate in the network as a legitimate user. As all of participants can act as either initiator and responder, it is necessary to describe protocols with variables of participants. Variables of participants, X and Y, range over the set of participants $\mathcal{U} = \{A, B, S\}$. We use small characters for subscripts of symbols, i.e., x, y range over $\mathcal{U} = \{a, b, s\}$. For instance, when the sabotuer impersonate a user X, we write S_x.

Each participant possesses *private keys* D_x and *public keys* E_x, which are defined on $\{0,1\}^*$. When a *message* $M \in \{0,1\}^*$ is concatinated with a name X, denoting MX, we use a *name–suffix* operator i_x defined as $i_x(M) = MX$. On the other hand, deleting a name–suffix from MX, we use a *name–cancellation* operator d_x defined if X is a suffix of T, i.e., $T = MX$, $d_x(MX) = M$; else $d_x(T)$ is undefined. We call all of i_x and d_x by *name–operations.* All of operators E_x, D_x, i_x, d_x for each user X are defined on the finite set $\{0,1\}^*$ of bit–strings. If a participant receives (or sends) a cryptographic message, it is necessary to *decode* (or *encode*, respectively) the message. We call decoding and encodeing in one step execution by *procedure*.

By means of cryptographic functions and name–operations, a set of *cancellation rules* is given as follows. Let ε be an identity function on a set of messages. The cancellation rules are $E_x D_x = \varepsilon$, $D_x E_x = \varepsilon$, and $d_x i_x = \varepsilon$ for each user X. Note that the cancellation rules of d_x and i_x cannot be symmetric. For a sequence φ of operators we denote a *reduced form* $\overline{\varphi}$ if no cancellation rule is applicable. The identity function ε can be regarded as the empty sequence (identitiy) in terms of a rewriting system on strings of cryptographic functions.

It is necessary to specify a set Σ_x of available operations for each user X. Define

$$\begin{aligned} \Sigma_x &= \{D_x\} \cup \{E_y, i_y, d_y \mid y \in \mathcal{U}\} \\ \Sigma &= \bigcup \{\Sigma_x \mid x \in \mathcal{U}\}. \end{aligned}$$

Actually name–cancellation operators would not be used in encoding procedure. Moreover each participant would not use their own public key because no one could decode the associated message. So that a set Δ_x of available operations in encoding and a set Δ_x^{-1} of their inverse is defined as follows.

$$\begin{aligned} \Delta_x &= \Sigma_x - (\{d_y \mid y \in \mathcal{U}\} \cup \{E_x\}) \\ \Delta_x^{-1} &= \{f^{-1} \mid f \in \Delta_x\} \end{aligned}$$

where $i_x^{-1} = d_x$ and $d_x^{-1} = i_x$.

Ping–pong protocols are given as a series of sequences consisting of cryptographic functions and name–operations. Following [5] and [4] we assume to fix a message in the whole communication, i.e., participants decode received messages and send encoded messages where those messages are the same message that the initiator has sent. If one receives an message, he needs to know that it was dealt in the legitimate procedure of encoding, signature and name–operation in the protocol. We assume each procedure to have two parts, receiving operations and sending operations.

Definition 1
A ping–pong protocol $P(X, Y)$ *is a finite series* $\{\alpha_k^{xy} \mid x \neq y, 1 \leq k \leq n\}$ *of sequence of operators such that* $\alpha_k^{xy} \in \Delta_x^*$ *if k is an odd number, or* $\alpha_k^{xy} \in \Delta_y^*$ *if k is an even number. We call each sequence* α_k^{xy} *of operators and its application* procedure.

In a communication between A and B, denoting their protocol by $P(A, B)$, the initiator A sends the first message $\alpha_1^{ab}(M)$, the responder B applies $(\alpha_1^{ab})^{-1}$ to the received message and obtain M. Next B sends the second message $\alpha_2^{ab}(M)$ back to A, the initiator gets M using $(\alpha_2^{ab})^{-1}$ and sends the third message $\alpha_3^{ab}(M)$ again. By the k–th step the message M has been applied operations as following: $\{\alpha_k^{ab}(\alpha_{k-1}^{ab})^{-1}\} \cdots \{\alpha_2^{ab}(\alpha_1^{ab})^{-1}\}\{\alpha_1^{ab}\}(M) = \alpha_k^{ab}(M)$ where $k \geq 2$.

We distinguish terminology initiator and responder from *sender* and *receiver*. Initiator and responder would be fixed in executions of protocols. We call a sender (receiver) to those who sends (receives, respectively) a message in one procedure.

3 Examples of attack

The purpose of the sabotuer is to read other participant's messages making use of flaws in cryptographic protocols, moreover eavesdropping must be done without being noticed by legitimate participants. Since the sabotuer may take part in the network as a legitimate user, he is supposed to follow procedures in the protocol and does not know other's private keys. However, we assume that the sabotuer can intercept and substitute transferring messages, impersonate legitimate participants, and initiate the protocol. In the following examples the notation $X \to Y : \varphi(\psi(M))$ means that a received message $\psi(M)$ is sent by X with operation φ. In each step the message should be revealed successfully. So that we assume that it must hold that $\overline{\varphi''\psi} = \varepsilon$ where $\varphi = \varphi'\varphi''$.

The simplest ping–pong protocol, called *echo* protocol, $P_0(X, Y) = \{\alpha_1^{xy} = E_y,\ \alpha_2^{ab} = E_x\}$ is vulnerable because it is impossible for a receiver to verify and the message which is indeed sent by a legitimate sender. If the sabotuer succeeds to intercept a message and impersonates the initiator, then the attack would be done successfully. Now one would try to improve the echo protocol using adding digital signature, i.e., adding private key D_x in each procedure, but with no success.

Example 1
The protocol

$$P_1(X,Y) \quad = \quad \{\alpha_1^{xy} = E_y D_x,\ \alpha_2^{xy} = E_x D_y\}$$

is vulnerable.

$$\begin{array}{llcl}
(1.1) & A \to B & : & \underline{E_b} D_a(M) \\
(2.1) & S \to B & : & E_b D_s(\underline{E_b} D_a(M)) \\
 & & & \{\ S \text{ intercepts (1.1) and sends to } B.\ \} \\
(2.2) & B \to S & : & E_s D_b E_s D_b (E_b D_s(\underline{E_b} D_a(M))) \\
 & & & = E_s D_a(M) \\
 & & & \{\ \text{Eavesdropping has been successful.}\ \} \\
(3.1) & S \to B & : & E_b D_s E_a D_s (E_s D_a(M)) \\
 & & & = E_b D_s(M) \\
 & & & \{\ \text{Preparation for responding to } A\ \} \\
(3.2) & B \to S & : & E_s D_b E_s D_b (E_b D_s(M)) \\
 & & & = E_s D_b(M) \\
(1.2) & S_b \to A & : & E_a D_s (E_s D_b(M)) \\
 & & & = E_a D_b(M) \\
 & & & \{\ A \text{ also successfully received the reply message.}\ \}
\end{array}$$

The sabotuer S impersonate B in the final session to terminate the session beginning at (1.1). The underlined operator $\underline{E_b}$ is the target operator for the sabotuer. Note that in (2.2) the signature D_b by B is abused for decryption of $\underline{E_b}$.□

Dolev, Even and Karp[4] represented an $O(n^3)$ verification algorithm for ping–pong protocols where n is the number of operators appearing in a protocol. The protocol $P_2(X,Y) = \{\alpha_1^{xy} = E_y i_x,\ \alpha_2^{xy} = E_x\}$ was verified by the algorithm in [4]. However if we add a digital signature D_y to α_2^{xy} in $P_2(X,Y)$ then it becomes insecure.

Example 2
The protocol

$$P_3(X,Y) \quad = \quad \{\alpha_1^{xy} = E_y i_x,\ \alpha_2^{xy} = E_x D_y\}$$

is vulnerable. One can attack in the following way. The underlined $\underline{E_b}$ is a target operator for the sabotuer.

$$\begin{array}{llcl}
(1.1) & A \to B & : & \underline{E_b} i_a(M) \\
(2.1) & S \to B & : & E_b i_s(\underline{E_b} i_a(M)) \\
 & & & \{\text{Intercept and apply } E_b i_s.\ \} \\
(2.2) & B \to S & : & E_s D_b d_s D_b (E_b i_s \underline{E_b} i_a(M)) \\
 & & & = E_s i_a(M) \\
 & & & \{\ S \text{ can obtain } M.\ \} \\
(3.1) & S \to B & : & E_b i_s d_a D_s (E_s i_a(M)) \\
 & & & = E_b i_s(M) \\
(3.2) & B \to S & : & E_s D_b d_s D_b (E_b i_s(M)) \\
 & & & = E_s D_b(M) \\
(1.2) & S_b \to A & : & E_a D_s (E_s D_b(M)) \\
 & & & = E_a D_b(M) \\
 & & & \{\ A \text{ received the message.}\ \}
\end{array}$$

Note that the signature function D_b is abused to decrypt message as well as Example 1.

4 Secure Patterns

We will give a sufficient condition of ping–pong protocols in terms of security. Since the condition is simple, if one finds a security flaw in protocols with some verification algorithms almost all of the insecure protocols can be improved, or one can design a secure protocol satisfying the condition.

The definition of security of ping–pong protocol follows [4]. Let the set Γ to be the sabotuer's devices in a given protocol $P(X,Y)$, that is,

$$\Gamma \quad = \quad [\Sigma_s \cup \{\alpha_1^{xy} \mid x, y \in \mathcal{U}\} \cup \{(\alpha_k^{xy})(\alpha_{k-1}^{xy})^{-1} \mid x, y \in \mathcal{U},\ 2 \le k \le n\}]^*.$$

where $x, y \in \mathcal{U}$. In examples of the previous section, the sabotuer attempts to lead legitimate participants to reduce messages using protocols $P(A, B)$, $P(S, A)$ and $P(S, B)$ except $\alpha_1^{ab}(M)$.

Definition 2
A protocol $P(A, B)$ is vulnerable *if there exists $\gamma \in \Gamma$ such that $\overline{\gamma\alpha_1^{ab}} = \varepsilon$.*

The first procedure of the next protocol includes a name-suffix operator. It is impossible for the sabotuer to crack the protocol. The next proposition leads us to the general idea of a secure design of protocols.

Proposition 1
Let a protocol

$$P_4(X, Y) = \{\alpha_1^{xy} = E_y i_x,\ \alpha_2^{xy} = E_x\}.$$

$P_4(A, B)$ is secure.

[Proof] Suppose $P(A, B)$ to be insecure, i.e., $\exists\gamma \in \Gamma$ such that $\overline{\gamma\alpha_1^{ab}} = \overline{\gamma E_b i_a} = \varepsilon$. The B's private key D_b which cancel with E_b in α_1^{ab} appears in subsequences of γ, that is, $\alpha_2^{ab}(\alpha_1^{ab})^{-1}$ (for case 1.) or $\alpha_2^{sb}(\alpha_1^{sb})^{-1}$ (for case 2.).

1. Assume that E_b is suppose to cancel with D_b in $\alpha_2^{sb}(\alpha_1^{sb})^{-1}$. Then we have the following: $\gamma E_b i_a = \varphi\alpha_2^{sb}\tau E_b i_a = \varphi E_s d_s D_b \tau E_b i_a$ where $\gamma = \varphi\alpha_2^{sb}(\alpha_1^{sb})^{-1}\tau$ for some $\varphi, \tau \in \Gamma$. By assumption the subsequence $D_b \tau E_b$ would be reduced to ε. But there is no i_s in the right side from d_s, which should be cancelled with d_s. This contradicts that $\overline{\gamma\alpha_1^{ab}} = \varepsilon$.

2. Next assume that E_b is suppose to cancel with D_b in $\alpha_2^{ab}(\alpha_1^{ab})^{-1}$. We have $\gamma E_b i_a = \varphi\alpha_2^{ab}(\alpha_1^{ab})^{-1}\tau E_b i_a = \varphi E_a d_a D_b \tau E_b i_a$ where $\gamma = \varphi\alpha_2^{ab}(\alpha_1^{ab})^{-1}\tau$ for some $\varphi, \tau \in \Gamma$. As the subsequence $d_a D_b \tau E_b i_a$ is cancelled, it must holds that $\overline{\varphi E_a} = \varepsilon$. Then we have two cases that E_a is suppose to be cancelled with D_a which appears in $\alpha_2^{sa}(\alpha_1^{sa})^{-1}$ or $\alpha_2^{ba}(\alpha_1^{ba})^{-1}$. In case of $\alpha_2^{sa}(\alpha_1^{sa})^{-1}$ it contradicts the assumption like case 1. The case of $\alpha_2^{ba}(\alpha_1^{ba})^{-1}$ leads us to contradiction which conflicts finiteness of γ.

Both case 1. and case 2. lead to contradiction. The proof completes.□ □

It seems enough for secure protocols to have a name–suffix function in α_1. However there is a counterexample shown in Example 2. Now we will state that protocols in which each procedure has a name–suffix function at the first operation are secure. In a *name–suffixed* protocol $P(X, Y) = \{\alpha_k^{xy} \mid x \neq y, 1 \leq k \leq n\}$, each encoding is of the following *reduced* form:

$$\alpha_k^{xy} = \xi_k^{xy} E_v \pi_k^{xy} i_u$$

where

$$\xi_k^{xy} \in \begin{cases} \Delta_x^* & \text{if } k \text{ is odd,} \\ \Delta_y^* & \text{if } k \text{ is even,} \end{cases}$$

$$\pi_k^{xy} \in \begin{cases} (\Delta_x - \{E_y\})^* & \text{if } k \text{ is odd,} \\ (\Delta_y - \{E_x\})^* & \text{if } k \text{ is even,} \end{cases}$$

and

$$(u, v) = \begin{cases} (x, y) & \text{if } k \text{ is odd,} \\ (y, x) & \text{if } k \text{ is even.} \end{cases}$$

That is, each encoding begins with a name–suffixed operator of the sender and has at least one encryption function. The next lemma is important.

Lemma 1
For every procedure α_k^{ab} in a name–suffixed ping–pong protocol $P(A, B)$, it holds that $\overline{\gamma\alpha_k^{ab}} \neq \varepsilon$ for any $\gamma \in \Gamma^*$.

[Proof] Suppose that there exists $\gamma \in \Gamma$ such that $\overline{\gamma\alpha_k^{ab}} = \overline{\gamma\xi_k^{ab} E_b \pi_k^{ab} i_a} = \varepsilon$ By assumption there is a decryption function D_b which is cancelled with E_b in α_1^{ab}, that is, we can assume that there exists $j \geq 2$ where such D_b is included in subsequences, $\alpha_j^{ab}(\alpha_{j-1}^{ab})^{-1}$ or $\alpha_j^{sb}(\alpha_{j-1}^{sb})^{-1}$ if j is even, else $\alpha_j^{ba}(\alpha_{j-1}^{ba})^{-1}$ or $\alpha_j^{bs}(\alpha_{j-1}^{bs})^{-1}$. Now we prove the case that j is even. Assume that the sabotuer does not take part in the execution, i.e., the sequences neither $\alpha_l^{sb}(\alpha_{l-1}^{sb})^{-1}$ nor $\alpha_l^{bs}(\alpha_{l-1}^{bs})^{-1}$ appears in γ for any $l \geq 2$. Then $\gamma = \varphi\alpha_j^{ab}(\alpha_{j-1}^{ab})^{-1}\varphi'$ for some $\varphi, \varphi' \in \Gamma$. So that

$$\gamma\alpha_k^{ab} = \varphi \cdot \underbrace{\xi_j^{ab} E_a \pi_j^{ab} i_b}_{\alpha_j^{ab}} \cdot \underbrace{d_a (\pi_{j-1}^{ab})^{-1} D_b (\xi_{j-1}^{ab})^{-1}}_{(\alpha_{j-1}^{ab})^{-1}} \cdot \varphi' \cdot \xi_k^{ab} E_b \pi_k^{ab} i_a$$

It is possible to include D_b in either ξ_j^{ab} or $D_b(\xi_{j-1}^{ab})^{-1}$. Consider that the target E_b is cancelled with D_b in ξ_j^{ab}, then the subsequence

$$(\text{the rest of } \xi_j^{ab})E_a\pi_j^{ab}i_bd_a(\pi_{j-1}^{ab})^{-1}D_b(\xi_{j-1}^{ab})^{-1}\varphi'\xi_k^{ab}$$

should corrupts by itself. As d_b does not however exist in the right side from i_b, this is a contradiction. Next consider the target E_b is cancelled with D_b in $D_b(\xi_{j-1}^{ab})^{-1}$. Then E_a in the right α_j^{ab} must be cancelled with D_a in φ. Now we have series of encodings related to that cancellation above. By assumption that the sabotuer does not participate in the execution, γ is of the following form: $\gamma = \cdots\alpha_{j_\nu}(\alpha_{j_\nu-1})^{-1}\cdots\alpha_{j_1}(\alpha_{j_1-1})^{-1}\cdots\alpha_j(\alpha_{j-1})^{-1}$. The participants must be A or B but cannot be determined for $j_1, j_2, j_3, \cdots$. This contradicts to the corruption of the whole string $\gamma\alpha_k^{ab}$ because of the fact that E_a or E_b will remain anyway. Whence the sabotuer must participate in the execution γ. Now assume that E_b in $\gamma\alpha_k^{ab}$ is cancelled with D_b in $\alpha_j^{sb}(\alpha_{j-1}^{sb})$. We have

$$\gamma\alpha_1^{ab} \quad = \quad \varphi\cdot\underbrace{\xi_j^{sb}E_s\pi_j^{sb}i_b}_{\alpha_j^{sb}}\cdot\underbrace{d_s(\pi_{j-1}^{sb})^{-1}D_b(\xi_{j-1}^{sb})^{-1}}_{(\alpha_{j-1}^{sb})^{-1}}\cdot\varphi'\cdot\xi_k^{ab}E_b\pi_k^{ab}i_a$$

It is possible to consider that E_b is cancelled with in either ξ_j^{sb} or $D_b(\xi_{j-1}^{sb})^{-1}$. We can conclude a contradiction that i_b cannot be cancelled for the former case and neither can d_s for the latter case. We can prove the case that j is odd by symmetry.□ □

If there is $k \geq 1$ such that $\exists\gamma' \in \Gamma : \overline{\gamma'\alpha_k^{ab}} = \varepsilon$, then the protocol is vulnerable because we can set $\gamma = \gamma'\alpha_k^{ab}\cdots(\alpha_1^{ab})^{-1}$.

From Lemma 1 it is easy to state the theorem:

Theorem 1
For a name–suffixed ping–pong protocol $P(X,Y)$, $P(A,B)$ is secure.

[Proof] It is clear by the case of $k = 1$ in Lemma 1. □

We need a reconsideration about the definition of security in the sense of [4] since there exists an attack example which would be noticed by legitimate users. We leave this discussion in the next section.

5 Discussion

We represented a sufficient condition of security for ping–pong protocols from a standpoint of [4]. Now we need to discuss attacking methods and definitions of security. Here are examples which are verified to be vulnerable by means of the algorithm in [4], but smart attacks cannot be found.

Example 3
The protocol $P_5(X,Y) = \{\alpha_1^{xy} = E_yD_x,\ \alpha_2^{xy} = E_x\}$ and $P_6(X,Y) = \{\alpha_1^{xy} = E_yi_xD_x,\ \alpha_2^{xy} = E_x\}$ are vulnerable by setting $\gamma = E_aD_sD_s\alpha_1^{sb}(\alpha_1^{sb})^{-1}$ for both P_5 and P_6.

Those examples are determined to be vulnerable by the algorithm in [4], but such smart attack as in Example 1 and 2 have not been found, i.e., during decoding the responder of the first session would notice intervention. In those smart attacks, message–direct digital signatures in the second (and more, maybe) procedure are abused for illegitimate decryption of public keys. For these reasons it is necessary to pay attentions for distinguishing between smart attack, successful attack except that some execution are aborted, and attack which can be noticed because of verifying digital signatures and name–suffixing.

Recalling the proof in Lemma 1 we can conclude two principles about robustness of name–suffixing as follows. Firstly, secure cryptographic protocols are required that each procedure cannot be decoded in illegitimate ways. This is a trivial principle. Secondly name–suffixing contributes security cooperating with encryption by public keys. It is important that decryption in procedures before and after does not distrub machinery of name–suffixing. The condition presented in this paper is one of the most essential and simple instance satisfying this principle, however, more efficient and non-redundant name-suffixing condition would be studied, taking account of simple feature of protocols. As Lowe [6], and Abadi and Needham [2] have pointed out, this is worth while researching with respect to contemporary cryptographic protocols.

References

[1] Martin Abadi. Explicit communication revisited: Two new attacks on authentication protocols. *Software Engineering*, 23(3):185–186, 1997.

[2] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

[3] Whitfield Diffie, Paul C. Van Oorschot, and Michael Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

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

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

[6] Gavin Lowe. An attack on the Needham-Schroeder public key authentication protocol. *Information Processing Letters*, 56(3):131–136, 1995.

[7] Gavin Lowe. Some new attacks upon security protocols. In *PCSFW: Proceedings of The 9th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1996.

[8] Catherine Meadows. Formal verification of cryptographic protocols: A survey. In *ASIACRYPT: Proceedings of International Conference on the Theory and Application of Cryptology*. LNCS 917, Springer-Verlag, 1994.

[9] Catherine Meadows. Open issues in formal methods for cryptographic protocol analysis. In *Proceedings of DARPA Information Survivability Conference and Exposition*, pages 237–250. IEEE Computer Society Press, 2000.