Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 174 (2007) 117–130

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Method for Patching Interleaving-Replay Attacks in Faulty Security Protocols[1]

Juan Carlos Lopez Pimentel and Raul Monroy

Computer Science Department
Tecnológico de Monterrey, Campus Estado de México
Carretera al lago de Guadalupe, Km 3.5, Atizapán, 52926, Mexico
{juan.pimentel,raulm}@itesm.mx

Dieter Hutter

DFKI, Saarbrücken University
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
hutter@dfki.de

Abstract

The verification of security protocols has attracted a lot of interest in the formal methods community, yielding two main verification approaches: i) state exploration, e.g. FDR [8] and OFMC [2]; and ii) theorem proving, e.g. the Isabelle inductive method [12] and CORAL [13]. Complementing formal methods, Abadi and Needham's principles aim to guide the design of security protocols in order to make them simple and, hopefully, correct [1]. We are interested in a problem related to verification but far less explored: the correction of faulty security protocols. Experience has shown that the analysis of counterexamples or failed proof attempts often holds the key to the completion of proofs and for the correction of a faulty model. In this paper, we introduce a method for patching faulty security protocols that are susceptible to an interleaving-replay attack. Our method makes use of Abadi and Needham's principles for the prudent engineering practice for cryptographic protocols in order to guide the location of the fault in a protocol as well as the proposition of candidate patches. We have run a test on our method with encouraging results. The test set includes 21 faulty security protocols borrowed from the Clark-Jacob library [5].

Keywords: Fault localization, patching, replay attacks, security protocols, verification.

1 Introduction

Computer security is a major concern for IT. Users are reluctant to deliver confidential information over an insecure, hostile network. Computer crimes have already yielded countless losses. To ensure security, users use protocols. A *security protocol* is a set of rules and conventions whereby one or more agents agree about each others'

[1] We are grateful to Alan Bundy, Graham Steel and the reviewers for their useful comments on an earlier draft of this paper. The research reported here was supported by ITESM CCEM-0302-05.

doi:10.1016/j.entcs.2006.12.034

identity, usually ending up in the possession of one or more secrets [12]. Security protocols consist of only a few messages but amazingly they are very hard to get right. For example, the detection of a flaw in the 3-message Needham-Schroeder public key (NSPK) protocol took roughly 17 years [7].

The verification of security protocols has attracted a lot of interest in the formal methods community, yielding two main verification approaches: i) *state exploration*, e.g. FDR [8] and OFMC [2]; and ii) *theorem proving*, e.g. the Isabelle inductive method [12] and Coral [13]. Model checking tools are capable of determining whether or not a (finite abstraction of a) protocol is valid. The verification process usually takes a few seconds and, in the case of unsatisfiability, a counterexample (a protocol attack) is output. Theorem proving may be slower, but has a wider range of application, as demonstrated by [13].

Complementing formal methods, Abadi and Needham's principles aim to guide the design of security protocols in order to make them simple and, hopefully, correct [1]. Abadi and Needham arrived at their principles by noticing some common features hard to analyse among protocols. If these features are avoided, protocols tend to become more readable and, more importantly, correct.

We are interested in a problem related to verification but far less explored: the correction of faulty security protocols. A flawed protocol is a mal-formulation. Mal-formulations is central to theory refinement. They often become evident by the appearance of a failed proof attempt, possibly yielding a counterexample. The analysis of this evidence often holds the key to the completion of proofs and for the correction of a faulty model.

The correction of faulty security protocols requires to develop a set of patching methods capable of dealing with a general class of faults. In this paper, we introduce a method for patching an interesting class of faulty security protocols, which we have baptised interleaving-replay attacks. A *replay attack* is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. [2] According to the Paul Syverson's taxonomy, [14], *interleaving attack* is the replay of messages from outside the current run of the protocol requiring that two protocol runs overlap in execution.

Roughly, for carrying out our patching method it is necessary to follow a full verification cycle (Section 2). This cycle involves mainly three phases: i) to verify a faulty security protocol and obtain its counterexample; ii) to patch the faulty security protocol through a patching framework; and iii) to verify the newer protocol version. The crux of our patching method lies on phase ii), which rest upon Abadi and Needham's principles for the prudent engineering practice for cryptographic protocols (Section 3). These principles are applied to guide the location of the fault in a protocol as well as the proposition of candidate patches. To automatically patch a faulty security protocol, our method first analyses the protocol description in order to identify the rôle played by all components of each message (Section 4). Then, it analyses the protocol counterexample, obtained in phase i), in order to identify

[2] `http://en.wikipedia.org/wiki/Replay_attack`

non-trivial message parts shared in its runs, and using heuristics, it analyses these observations in order to both diagnose a possible fault and suggest a candidate patch (Section 5). We also present the methodology used in the invention of our patching method (Section 6). Finally, we have run a test on our method with encouraging results (Section 7). The test set includes 21 faulty security protocols borrowed from the Clark-Jacob library [5]. In addition, we also report a large number of results to validate phase iii) using the AVISPA tool.

2 A Full Verification Cycle

A full verification cycle will aid to security protocol designers in creating best security protocols, and thereby, to reduce risks after implementation. This cycle consists of three phases. In a first phase we use a model checker or a theorem prover to detect counterexamples illustrating that the security protocol is faulty. As a result we obtain an interleaving of (in general) several protocol runs that violate security requirements modeled within the specification of the model checker or the theorem prover.

In a second phase we must patch the faulty security protocol through a patching framework. This patching framework is a set of patching methods capable of dealing with a general class of faults. Each patching method has the form presented in figure 1 (name, input, preconditions, flaw and effect). A patching method may be split into two steps.

The first step is to find out those protocol messages which cause the failure of the protocol. Our patching method, for instance, makes interesting observations on the counterexample, using the rôle played by all components of each message. We analyzed various protocols for such situations and formalized them as preconditions for so-called problem location. In section 5 we concentrate on a particular example (the NSPK protocol) in which the identity of principals is crucial for the security of the protocol. The violation of this principle can be observed on a technical level of protocol traces by the fact that message parts that are (in some sense) confidential in one protocol run are reused in a second.

In the second step of the patching framework we have to change the protocol in order to avoid the particular fault. Therefore, we use this technical description of problem location as preconditions for specific rules that resolve such conflicts. Besides these preconditions, the method has a flaw part that consists of a verbose description of the flaw it attempts to fix, and the effect part (patch) that formalizes the changes to be made to the protocol. The changes, among faulty messages, resting upon Abadi and Needham's principles. In our patching method, for instance, principle 3 suggests the solution: we introduce the agent names into the faulty message part to disambiguate the context in which the message part can occur. The result is that the messages modified no longer can be used in both protocol runs.

In a third phase of the full verification cycle we have to verify the new version of the protocol in order to know whether the security protocol is free of flaws, or in the worst case, to be no longer susceptible to the same counterexample. For this, we must again use a model checker or a theorem prover. In this paper particularly we have used the AVISPA tool, a model checker, to deal with the first and the third phase of the full verification cycle. In the following sections we will explain a method for patching interleaving-replay attacks in more detail.

3 Abadi and Needham's Principles

Abadi and Needham's principles are a guideline for the design of security protocols. They are concerned with two main issues: i) the messages involved in a protocol, together with their content; and ii) the trust relations held by the participants.

Principle 1 deals with protocol messages and their content: *every message should say what it means, the interpretation of the message should depend only on its content.* It encompasses principles 3—10. Principle 3, *naming*, prescribes that the agent names relevant for a message should all be derivable either from the encryption keys that have been applied or other data, including the explicit mention of the agent names. Principle 4, *encryption*, is a guideline for the correct use of encryption; it prescribes being clear why encryption is performed (is it done for providing authenticity? or confidentiality?). Principle 5, *signing (encrypted) data*, specifies that the appearance of a signature does not necessarily imply that the signing agent knows the message content. Principle 10, *encoding*, prescribes being careful about message format: principals should be able to associate, from the message content, which step the message corresponds to of whatever protocol they are running. Principles 6—8 are a guideline for establishing message freshness and message association. Principle 6 prescribes being clear about the properties that are being assumed about nonces; principle 7 dictates being cautious about the use of predictable nonces; and principle 8 is an account of aspects of prudent practice in the use of timestamps. Finally, principle 9 prescribes being sure about the acceptability of the use of a key.

Conversely, principle 2 deals with the participants' trust relations: *the conditions for a message to be acted upon should be clearly set out so that someone reviewing a design may see whether they are acceptable or not.* It comprehends principle 11. Trust relations introduce dependencies, e.g. who is to be trusted on the generation of a session key? Whether these dependencies are acceptable should be founded on a policy, instead of a logic.[3]

4 Security Protocols

As pointed out by [1], most flaws in security protocols have their root in an improper use of cryptographic primitives. A protocol designer may miss the rôle played by a message component or overestimate the security guarantees provided by an encrypted message. We are currently developing a theory that will allow us to identify the one or the several rôles played by each message component.

[3] Thus, so far principles 2 and 11 are not used within our mechanism for correcting faulty security protocols.

4.1 Basic Ingredients

Our method makes use of Paulson's formalisation of agent population and message structure [12]. There are three kinds of agents: the server, S, an absolutely trusted agent, the friendly agents $(A, B, \ldots)$, and a Dolev-Yao spy, Spy [6].

Messages comprise agent names, nonces, time-stamps, shared keys (used in symmetric cryptography), public or private keys (used in asymmetric cryptography), and session keys. The notation $\{\!|X_1, \ldots, X_{n-1}, X_n|\!\}$ is used to abbreviate the compound message $\mathsf{MPair}\ X_1 \ldots (\mathsf{MPair}\ X_{n-1}\ X_n)$. The symbols K_{ab}, K_{as} and K_{bs} denote specific shared keys; K_a^+, K_b^+ and K_s^+ denote specific public keys; and K_a^-, K_b^- and K_s^- denote the corresponding private keys. The symbols N_a, N_b and N_c denote nonces; and T_a, T_b and T_s denote time-stamps. We write $\{\!|M|\!\}_K$ to denote the encryption of message M under key K.

Unlike Paulson, we specify a protocol as a sequence of steps, each of which is of the form: $n.\ A \rightarrow B\ :\ M$, meaning that, at step n, A sends B the message M, which B receives. Notice that each step involves two communicating events: the sending and the reception of the associated message. A protocol counterexample, which consists of a number of parallel protocol runs, is then specified as a sequence of session steps, each of which is of the form $S\!:\!n.\ A \rightarrow B\ :\ M$, denoting the nth step of session S.

4.2 Message Parts and their Rôles

An agent may play one of two rôles in a protocol: initiator or responder. An *initiator* is the agent requesting a session in a protocol and a *responder* is any agent answering to that request. Usually, the initiator starts the run in a protocol, thus:

$$\mathrm{initiator}(P) \stackrel{\mathrm{def}}{=} \mathrm{sender}(P, 1)$$

$$\mathrm{responder}(P) \stackrel{\mathrm{def}}{=} \{r\!:\!\mathsf{Agent}\ | r \neq \mathrm{initiator}(P) \wedge r \neq \mathsf{S} \wedge r \in \mathrm{participants}(P)\}$$

where participants(P) (respectively sender(P, n)) return all the participating agents (respectively the sender of the nth step) in protocol P.

In a protocol, each message component carries out a specific rôle. The rôle of a message component can sometimes be found via a syntactic analysis. Following [11]:

A secret distributor is a ciphered message carrying a secret. A *secret* is a message that is never sent in clear during the execution of the protocol. The function symbol secretDist(P) is used to denote the set of all secrets in protocol P; in symbols:

$$\begin{aligned}\mathrm{secretDist}(P) \stackrel{\mathrm{def}}{=} \{\{\!|M|\!\}_K \mid \{\!|M|\!\}_K \in \mathsf{parts}\ \mathrm{msgOf}(P)\\ \wedge\ \exists m \in \mathsf{parts}\ M.\ \forall M_1 \in \mathrm{msgOf}(P).\ m \notin \mathrm{msg2set}(M_1)\}\end{aligned}$$

where parts is Paulson's operation on sets of messages, msgOf(P) returns the messages exchanged in protocol P and where msg2set(M) returns the set of components comprising message M.

An authenticator is a ciphered message component that is used to provide evidence of the message sender's identity.

As shown below, these simple notions allow us to make interesting observations both in a protocol and in one of its counterexamples.

5 A Method for Patching Interleaving-Replay Attacks

To explain the rationale behind our patching mechanism, we shall show an example correction of a faulty security protocol, namely: the NSPK protocol:

$$1\ A \rightarrow B : \{|Na, A|\}_{K_B^+}$$
$$2\ B \rightarrow A : \{|Na, Nb|\}_{K_A^+}$$
$$3\ A \rightarrow B : \{|Nb|\}_{K_B^+}$$

The NSPK protocol seems right at first glance, but it is faulty. Lowe found that an intruder could impersonate one agent holding concurrently a session with another agent [7]:

$$\begin{array}{llcll}
s1 : 1. & A & \rightarrow & Spy & : \{|Na, A|\}_{K_{Spy}^+} \\
s2 : 1. & Spy(A) & \rightarrow & B & : \{|Na, A|\}_{K_B^+} \\
s2 : 2. & B & \rightarrow & Spy(A) & : \{|Na, Nb|\}_{K_A^+} \\
s1 : 2. & Spy & \rightarrow & A & : \{|Na, Nb|\}_{K_A^+} \\
s1 : 3. & A & \rightarrow & Spy & : \{|Nb|\}_{K_{Spy}^+} \\
s2 : 3. & Spy(A) & \rightarrow & B & : \{|Nb|\}_{K_B^+}
\end{array}$$

5.1 Failure Detection

In Lowe's attack, an identical instance of message 2, $\{Na, Nb\}_{K_A^+}$, is used in two independent runs (*traces* $s1$ and $s2$). The deceived agent, A, the initiator of the first trace, is the intended recipient of both instances of message 2, but she cannot distinguish who built or sent it. Thus, while B knows that A has recently participated in a run of the protocol, he cannot tell whether A is running it apparently with him.

In this attack, the spy has accomplished an interleaving-replay attack: after monitoring a (possibly partial) run of a protocol, he has replayed in another different run of the protocol one or more messages, impersonating a friendly agent. As we can see such runs of the protocol interleave in execution. If the corresponding agent does not have any mechanism to distinguish who originated an inward message or whom such a message is intended for, or cannot associate with that message a time line, then she will be deceived. Often, the messages that the spy selects for replay contain one or more secret distributors.

Thus, replay faulty security protocols violate Abadi and Needham's third principle, namely:

Principle 3: *If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message.*

Patching the protocol by adding the name of the agent sending message 2, B in this case, as suggested by the third principle, we arrive at the fixing Lowe has found [8]:

$$1\ A \rightarrow B : \{|Na, A|\}_{K_B^+}$$
$$2\ B \rightarrow A : \{|\mathbf{B}, Na, Nb|\}_{K_A^+}$$
$$3\ A \rightarrow B : \{|Nb|\}_{K_B^+}$$

Our method for patching replay faulty security protocols is shown in Figure 1. It is a 4-tuple, consisting of input information, preconditions, flaw and effects. The method is *applicable* if the method preconditions hold, considering both the input protocol and one of its counterexamples. Preconditions specify properties of either the protocol or the counterexample. They are expressed in a meta-logic involving the symbols introduced in Section 4. The flaw consists of an informal description of the design principle that the protocol is thought to violate. Effects specify the way in which the protocol should be modified. The newer version of the protocol is expected, in the worst case, to be no longer susceptible to an interleaving-replay attack, and, in an ideal case, to be robust enough to survive any attack.

Input: Protocol P, Counterexample C

Preconditions (problem location:)

$$\begin{aligned}
&\{|M|\}_K \in \mathrm{secretDist}(P) \\
&\wedge\ tr_1 \in \mathsf{Traces}(C) \wedge tr_2 \in \mathsf{Traces}(C) \wedge tr_1 \neq tr_2 \\
&\wedge\ S_1 : n_1.\ A \rightarrow B\ : m_1 \in tr_1 \wedge \{|M|\}_K \in \mathsf{parts}\ \mathrm{msg2set}(m_1) \\
&\wedge\ S_2 : n_2.\ A' \rightarrow B' : m_2 \in tr_2 \wedge \{|M|\}_K \in \mathsf{parts}\ \mathrm{msg2set}(m_2) \\
&\wedge\ (\mathrm{initiator}(P) \notin \mathrm{msg2set}(M) \vee \mathrm{responder}(P) \notin \mathrm{msg2Set}(M))
\end{aligned}$$

Flaw: Protocol possibly violates Abadi & Needham's principle #3, the initiator or the responder is omitted in M

Effect (patch:)

For every step in the protocol, $n.\ A \rightarrow B : m$,
where $\{|M|\}_K \in \mathsf{parts}\ \mathrm{msg2set}(m)$,
add, as described in Section 5.2, A and B to M forming M',
and then replace M with M' in m.

Fig. 1. The interleaving-replay patching method

5.2 Patch Formation

In our method, patching an interleaving-replay faulty protocol amounts to adding agent names to a collection of secret distributors. This addition process, however, attempts to avoid the introduction of redundancies and the introduction of confusions.

Sometimes the name of a participant agent can be safely deduced in a cyphered message from the encryption key. So, to avoid the introduction of redundancies, we apply the following well-known consequences of encryption techniques:

Authenticity in asymmetric cryptography: Any message encrypted under K_b^- comes originally from B, as long as B is not compromised.

Confidentiality in asymmetric encryption: Any message encrypted under K_a^+ can be decrypted only by A (as long as A is not compromised) using his private key K_a^-.

Authenticity in symmetric encryption: Anything that A receives encrypted under the long-term key she shares with the trusted sever, K_{as}, comes originally from the server, as long as A did not send it. This applies similarly with A and the server interchanged.

To avoid message confusion, we look back at the protocol description. We need to ensure that the new message component or even the entire new message does not have a structure that is similar to another one in the protocol. In case of a potential message confusion, we introduce all names using some ordering, e.g. sender first and then responder(s), see CCITT X.509(3) protocol in table 2.

6 Development Methodology

We now outline the methodology used in the invention of our patching method.

Firstly, we distinguished two sets of faulty security protocols:

Development: this class consists of a few protocol examples that were used for designing the patching method. For this to make sense, the development faulty protocol must have a similar flaw (at least be subject to the same kind of attack) and they should be different in size, type of cryptography used, participants (involving a trusted server or not), etc. The method was tested by hand on the development set before implementation;

Testing: this class contains example protocols used for testing the robustness of the method, and was considered only when the development was complete. The testing set includes the development set but also contains examples that were not used during development.

Secondly, we attempted to keep the development protocol examples as dissimilar as possible.

Thirdly, we gathered examples from different sources, e.g., books, research reports, etc., and from the Clark-Jacob library, which turned out to be the definitive source.

For the interleaving-replay patching mechanism, the development set included only four protocols, namely: the NSPK protocol, the Wide-Mouth Frog (WMF) protocol, the Denning-Sacco PK protocol, and the Abadi and Needham version of the Otway-Rees protocol.

Id	Protocol Name	Cryptography type	Attack type	Fixed
1	Andrew Secure RPC	Symmetric	CR	
2	BAN concrete ASRPC	Symmetric	IR	√
3	CCITT X.509 (1) (+)	Asymmetric	IR	√
4	CCITT X.509 (3) (+)	Asymmetric	IR	√
5	Denning-Sacco SK	Symmetric	CR	
6	Denning-Sacco PK	Asymmetric	IR	√
7	Kao Chow Auth-V.1	Symmetric	CR	
8	KSL	Symmetric	IR	√
9	Neumann Stubblebine	Symmetric	TF	
10	NSPK (+)	Asymmetric	IR	√
11	Needham Schroeder SK	Symmetric	CR	
12	Otway-Rees	Symmetric	TF	
13	O&R BAN version	Symmetric	IR	√
14	SPLICE/AS	Asymmetric	IR	√
15	Hwang-Chen SPLICE/AS	Asymmetric	IR	√
16	C-J modified SPLICE/AS	Asymmetric	CR	
17	TMN	Symmetric	CR	
18	WMF protocol	Symmetric	IR	√
19	Woo and Lam Mutual	Symmetric	IR	√
20	Woo and Lam Pi (+)	Symmetric	TF	
21	BAN modified Yahalom	Symmetric	IR	√

Table 1
The validation test set

7 Results

Once our method has patched a protocol, it is required to know whether the new version of the protocol is free of flaws, in the worst case, to be no longer susceptible to the same interleaving-replay attack (phase three of the full verification cycle). For this, we have used the AVISPA tool v.1.0 (Automated Validation of Internet Security Protocols and Applications).[4] For verifying a protocol in this tool, one must formulate the protocol and the properties to be verified (e.g. secrecy or/and authentication) in a high-level protocol specification language (HLPSL). Then, we must prove the protocol into one of the four different back-end search engines: the On the Fly Model Checker (OFMC), the Constraint-Logic-Based Attack Searcher (CL-AtSe), the SAT-based Model-Checker (SATMC) and the Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP). The first two back-ends, OFMC [3] and CL-AtSe [15], were used for the experimental proofs corresponding to phases one and three of the full verification cycle.

Tables 1—5 summarise our results. Table 1 includes our validation test set. It consists of 21 faulty protocols, borrowed from the Clark-Jacob library. The Clark-Jacob library comprehends 50 protocols, 26 of which are known to be faulty. So our validation test set contains all but five known faulty security protocols. The flaw in the faulty protocols that were left out from our validation test set have nothing to do with a replay attack.

[4] Available via http://www.avispa-project.org/

In Table 1, protocols annotated with (+) aim to provide authentication; the remaining protocols aim to achieve both authentication and session key distribution. The attack type is annotated with "IR", "TF" and "CR". "IR" abbreviates interleaving-replay, "TF" abbreviates type flaw [5] and "CR" abbreviates a classic replay [6]. Protocols marked with $\surd$ were patched. The patched version of the protocols are described in Tables 2—3.

Protocol Name	**New Description**	**Comment**
BAN concrete ASRPC	$1. A \to B : A, Na$ $2. B \to A : \{\!\|\boxed{\text{A, B}}, Na, Kpab\|\!\}_{Kab}$ $3. A \to B : \{\!\|Na\|\!\}_{Kpab}$ $4, B \to A : Nb$	This patch is similar to that proposed by Lowe in [9].
CCITT X.509 (1)	$1. A \to B : A, \{\!\|Ta, Na, B, Nxa, \{\!\|\boxed{\text{A}}, Nya\|\!\}_{K_B^+}\|\!\}_{K_A^-}$	This patch is similar to that proposed by Abadi and Needham in [1].
CCITT X.509 (3)	$1. A \to B : A, \{\!\|Ta, Na, B, Nxa, \{\!\|\boxed{\text{A}}, Nya\|\!\}_{K_B^+}\|\!\}_{K_A^-}$ $2. B \to A : B, \{\!\|Tb, Nb, A, Na, Nxb, \{\!\|\boxed{\text{A, B}}, Nyb\|\!\}_{K_A^+}\|\!\}_{K_B^-}$ $3. A \to B : A, \{\!\|\boxed{\text{B}}, Nb\|\!\}_{K_A^-}$	Here, one of two experimental patches is presented. This patch is similar to that proposed by Burrows, Abadi and Needham in [4].
Denning-Sacco PK	$1. A \to S : A, B$ $2. S \to A : \{\!\|A, K_A^+, Ts\|\!\}_{K_S^-}, \{\!\|B, K_B^+, Ts\|\!\}_{K_S^-}$ $3. A \to B : \{\!\|A, K_A^+, Ts\|\!\}_{K_S^-}, \{\!\|B, K_B^+, Ts\|\!\}_{K_S^-}, \{\!\|\{\!\|\boxed{\text{B}}, Kab, Ta\|\!\}_{K_A^-}\|\!\}_{K_B^+}$	This patch is the same as that proposed by Abadi-Needham in [1].

Table 2
Patched protocol descriptions, first part

Our method is thus able to identify an interleaving-replay attack and a candidate patch in 12 faulty protocols out of 21. Interestingly, our experiments have shown that even though our patching method suggests a different candidate patch when input two different counterexamples for the same faulty protocol, the verification in AVISPA tool have turned out successful in both candidate patches (protocols 4, 13, 18 and 19).

Protocols that could not be patched lie outside the scope of our method; that is, they are not interleaving-replay faulty protocols. For instance, protocols 5 and 16 lack of aliveness of the responder according to Lowe's hierarchy [10]. Lowe's Patch

[5] A type flaw attack is when an agent has no mechanism to identify a field that was originally intended to have one type and it is subsequently interpreted as having another type. For example, an agent that waits to receive a message of the form $\{\!|A, Kab, Tb|\!\}_{KB}$, receives the message $\{\!|A, Na, Tb|\!\}_{KB}$, in this case, he accepts naively nonce Na by key Kab.

[6] According to Paul Syverson's taxonomy [14], *a classic replay* is an attack not requiring contemporaneous runs. Classic replays have been identified lacking a time reference.

Protocol Name	**New Description**	**Comment**
KSL	$1.A \rightarrow B : Na, A$ $2.B \rightarrow S : Na, A, Nb, B$ $3.S \rightarrow B : \{\|\boxed{B}, Nb, A, Kab\|\}_{K_B}, \{\|Na, B, Kab\|\}_{K_A}$ $4.B \rightarrow A : \{\|Na, B, Kab\|\}_{K_A}, \{\|Tb, A, Kab\|\}_{Kbb}, Nc, \{\|Na\|\}_{Kab}$ $5.A \rightarrow B : \{\|Nc\|\}_{Kab}$ $6.A \rightarrow B : Nma, \{\|Tb, A, Kab\|\}_{Kbb}$ $7.B \rightarrow A : Nmb, \{\|Nma\|\}_{Kab}$ $8.A \rightarrow B : \{\|Nmb\|\}_{Kab}$	This patch is similar to that proposed by Lowe in [9]. Instead of adding the responder name B, in such a secret distributor, Lowe changes the order as follows: $\{\|A, Nb, Kab\|\}_{K_B}$. The effect is the same, to make the secret distributors of step 3 different one another.
NSPK	$1.A \rightarrow B : \{\|Na, A\|\}_{K_B^+}$ $2.B \rightarrow A : \{\|\boxed{B}, Na, Nb\|\}_{K_A^+}$ $3.A \rightarrow B : \{\|Nb\|\}_{K_B^+}$	This patch is the same as that proposed by Lowe in [7].
O&R BAN version	$1.A \rightarrow B : M, A, B, \{\|Na, M, A, B\|\}_{K_A}$ $2.B \rightarrow S : M, A, B, \{\|Na, M, A, B\|\}_{K_A}, Nb, \{\|M, A, B\|\}_{K_B}$ $3.S \rightarrow B : M, \{\|Na, K_{ab}\|\}_{K_A}, \{\|\boxed{A}, Nb, K_{ab}\|\}_{K_B}$ $4.B \rightarrow A : M, \{\|Na, K_{ab}\|\}_{K_A}$	Similar to CCITTX.509 (3) protocol our method proposed two different patches in two different counterexamples (Here, only one is presented).

Table 3
Patched protocol descriptions, second part

is extending the protocols by adding a nonce handshake.

Tables 2—4 show the output of our patching method. For each protocol, they describe the new specification. Changes amount to the inclusion of agent names, enclosed in boxes in the new description of the protocol.

Table 5 shows the total elapsed verification time (TEVT) and the back-end (OFMC or CL-Atse) used to verify the new protocol description. The experiments were carried out on a PC with 1.6 GHz Pentium IV processor and 512Mb RAM.

8 Conclusions and Further Work

In this paper, we have presented a method for patching faulty security protocols that are susceptible to an interleaving-replay attack. Using Abadi and Needham's guidance on agent naming, our mechanism patches a protocol by adding the necessary names to the protocol messages so they no longer can be replayed without notice. The patches proposed by our mechanism are natural, hence making the fault identification and correction processes look pretty trivial. Yet, this is not the case, since this type of design error has systematically appeared throughout the literature, as shown in tables 2–4. A recent example faulty protocol susceptible to an interleaving replay attack is the Asokan-Ginzboorg protocol, as shown by [13].

We have carried out a large number of experiments to validate our method. For instance, it has been tested on 21 faulty security protocols (borrowed from the

Protocol Name	**New Description**	**Comment**
SPLICE/AS	1.$C \rightarrow Sa : C, S, N1$ 2.$Sa \rightarrow C : Sa, \{\|\boxed{S}, Sa, C, N1, K_S^+\|\}_{K_{Sa}^-}$ 3.$C \rightarrow S : C, S, \{\|C, Tc, Lc, \{\|N2\|\}_{K_S^+}\|\}_{K_C^-}$ 4.$S \rightarrow Sa : S, C, N3$ 5.$Sa \rightarrow S : Sa, \{\|\boxed{C}, Sa, S, N3, K_C^+\|\}_{K_{Sa}^-}$ 6.$S \rightarrow C : S, C, \{\|S, f2(N2)\|\}_{K_C^+}$	This patch becomes the version of Hwang-Chen Splice/as protocol. Although this version is faulty, our method also patches it. See following row.
Hwang-Chen SPLICE/AS	1.$C \rightarrow Sa : C, S, N1$ 2.$Sa \rightarrow C : Sa, \{\|Sa, C, N1, S, K_S^+\|\}_{K_{Sa}^-}$ 3.$C \rightarrow S : C, S, \{\|C, Tc, Lc, \{\|\boxed{C}, N2\|\}_{K_S^+}\|\}_{K_C^-}$ 4.$S \rightarrow Sa : S, C, N3$ 5.$Sa \rightarrow S : Sa, \{\|Sa, S, N3, C, K_C^+\|\}_{K_{Sa}^-}$ 6.$S \rightarrow C : S, C, \{\|S, f2(N2)\|\}_{K_C^+}$	
WMF protocol	1.$A \rightarrow S : A, \{\|B, T_a, K_{ab}\|\}_{K_A}$ 2.$S \rightarrow B : \{\|\boxed{B}, A, T_s, K_{ab}\|\}_{K_B}$	See the following row comment.
Woo and Lam Mutual Auth.	1.$P \rightarrow Q : P, N1$ 2.$Q \rightarrow P : Q, N2$ 3.$P \rightarrow Q : \{\|P, Q, N1, N2\|\}_{K_P}$ 4.$Q \rightarrow S : \{\|P, Q, N1, N2\|\}_{K_P}, \{\|P, Q, N1, N2\|\}_{K_Q}$ 5.$S \rightarrow Q : \{\|\boxed{P}, Q, N1, N2, Kpq\|\}_{K_P}, \{\|P, N1, N2, Kpq\|\}_{K_Q}$ 6.$Q \rightarrow P : \{\|\boxed{P}, Q, N1, N2, Kpq\|\}_{K_P}, \{\|N1, N2\|\}_{Kpq}$ 7.$P \rightarrow Q : \{\|N2\|\}_{Kpq}$	Similar to WMF protocol our method proposed two different patches in two different counter-examples (Here, only one is presented).
BAN Yahalom	1.$A \rightarrow B : A, Na$ 2.$B \rightarrow S : B, Nb, \{\|A, Na\|\}_{K_B}$ 3.$S \rightarrow A : Nb, \{\|\boxed{A}, B, Kab, Na\|\}_{K_A}, \{\|A, Kab, Nb\|\}_{K_B}$ 4.$A \rightarrow B : \{\|A, Kab, Nb\|\}_{K_B}, \{\|Nb\|\}_{Kab}$	The patch is similar to that proposed by Paulson in [12].

Table 4
Patched protocol descriptions, third part

Clark-Jacob library) and 25 counterexamples. Our method has successfully dealt with 16 countexamples and has shown to be able successfully to patch 12 of these protocols.

Our method is usually able to patch an interleaving-replay faulty protocol that violates Abadi and Needham's principle 3. But it cannot always deal with replay faulty protocols that violate a time line reference (classic replays) or run internal attacks. For example, it did not succeed in patching the Andrew Secure RPC protocol:

(i) $A \rightarrow B : A, \{\|Na\|\}_{Kab}$

Protocol Name	**Tool**	**TEVT(sec)**
BAN concrete ASRPC	OFMC	34.34
CCITT X.509 (1)	OFMC	0.10
CCITT X.509 (3)	OFMC	2.27
Denning-Sacco PK	OFMC	0.33
KSL	OFMC	17.26
NSPK	CL-AtSe	28.8
O&R BAN version	OFMC	4.15
SPLICE/AS	CL-AtSe	8.43
Hwang-Chen SPLICE/AS	CL-AtSe	35.15
WMF protocol	CL-AtSe	0.70
Woo and Lam Mutual	OFMC	0.35
BAN modified Yahalom	OFMC	0.74

Table 5
The total elapsed verification time

(ii) $B \rightarrow A : \{|Na + 1, Nb|\}_{Kab}$

(iii) $A \rightarrow B : \{|Nb + 1|\}_{Kab}$

(iv) $B \rightarrow A : \{|Kabp, Nbp|\}_{Kab}$

Here, the origin of the error has nothing to do with agent naming, but with a time reference. In particular, notice that nonce Nbp is used as an acknowledgment, but Nbp's structure has no relation with Na.

We plan on further validating our method with other faulty protocols. In addition, we will analyse other faulty protocols, as that presented above, in order to propose new patching methods.

References

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

[2] A. D. Basin, S. Mödersheim, and L. Viganò. An on-the-fly model-checker for security protocol analysis. In D. Gollmann and E. Snekkenes, editors, *ESORICS'03: 8th European Symposium on Research in Computer Security*, number 2808 in Lecture Notes in Computer Science, pages 253–270, Gjøvik, Norway, 2003. Springer-Verlag.

[3] D. Basin, S. Mödersheim, and L. Viganò. OFMC: a symbolic model-checker for security protocols. Technical Report 450, ETH Zürich, Computer Science, June 2004.

[4] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *Proceedings of the Royal Society of London*, 426(1):233–71, 1989.

[5] J. Clark and J. Jacob. A survey of authentication protocol literature: Version 1.0. Technical report, Department of Computer Science, University of York, November 1997. A complete specification of the Clark-Jacob library in CAPSL is available at `http://www.cs.sri.com/ millen/capsl/`.

[6] D. Dolev and A. C. Yao. On the security of public key protocols. Technical Report 2, Stanford University, Stanford, CA, USA, 1983.

[7] Gavin Lowe. An attack on the needham-schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[8] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using FDR. In *TACAs '96: Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 147–166, London, UK, 1996. Springer-Verlag.

[9] Gavin Lowe. Some new attacks upon security protocols. In *CSFW'96: Proceedings of The 9th Computer Security Foundations Workshop*, page 162, Washington, DC, USA, 1996. IEEE Computer Society Press.

[10] Gavin Lowe. A hierarchy of authentication specifications. In *CSFW '97: Proceedings of the 10th Computer Security Foundations Workshop*, page 31, Rockport, Massachusetts, USA, 1997. IEEE Computer Society.

[11] R. Monroy and M. Carrillo. On automating the formulation of security goals under the inductive approach. In M.H. Hamza, editor, *IASTED'03: Proceedings of 21st International Conference on Applied Informatics*, pages 1020–1025, Innsbruck, Austria, 2003. Acta Press.

[12] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal in Computer Security*, 6(1-2):85–128, 1998.

[13] G. Steel, A. Bundy, and M. Maidl. Attacking the asokan-ginzboorg protocol for key distribution in an ad-hoc bluetooth network using coral. In H. König, M. Heiner, and A. Wolisz, editors, *IFIP TC6 /WG 6.1: Proceedings of 23rd IFIP International Conference on Formal Techniques for Networked and Distributed Systems*, volume 2767, pages 1–10, Berlin, Germany, 2003. FORTE 2003 (work in progress papers).

[14] Paul Syverson. A taxonomy of replay attacks. In *CSFW'94: Proceedings of the Seventh Computer Security Foundations Workshop*, pages 187–191, Franconia, New Hampshire, USA, 1994. IEEE Computer Society Press.

[15] Mathie Turuani. *Sécurité des Protocoles Cryptographiques:Décidabilité et Complexité.* PhD thesis, Université Henri Poincaré, December 2003.