On the Automated Correction of Faulty Security Protocols

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

I. 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 security protocols. A *security protocol* is a set of rules and conventions whereby one or more agents agree about each others' identity, usually ending up in the possession of one or more secrets [6]. Security protocols consist of only a few messages but amazingly they are very hard to get right. E.g., the detection of a flaw in the 3-message Needham-Schroeder public key (NSPK) protocol took roughly 17 years [5].

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 [5] and OFMC [2]; and ii) *theorem proving*, e.g. the Isabelle inductive method [6] and CORAL [7]. 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 [7].

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 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-formulation 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 [3].

This research rests upon the following hypothesis:

> *Using Abadi and Needham's principles in the analysis of both the description of a faulty protocol and of one of its counterexample holds the key for pinpointing the root of the fault in the protocol and for suggesting a way of patching it.*

In particular we are interested in the following objectives:

1) to provide a taxonomy of faults based on both the meaning of a step within a security protocol and Abadi and Needham's security protocol design principles;
2) to provide a method capable of pinpointing the root of malfunction in a faulty security protocol;
3) to provide a method capable of suggesting a means for patching a faulty security protocol; and
4) to provide general knowledge for the design of sound security protocols.

II. A General Patching Framework

The correction of faulty security protocols consists of a collection of patch methods capable of dealing with a general class of faults. Each of which aims to locate and then fix a protocol design error. Our patching framework encompasses three steps:

1) It analyses the protocol description in order to identify key components like protocol participants, protocol messages, message components and their rôle;
2) It analyses the protocol counterexample in order to identify non-trivial message parts shared in the runs of the attack;
3) It then uses the observations gathered from steps 1 and 2 in order to identify whether one or more patch methods can be applied. If none, our patching framework terminates with failure. Otherwise, one patch method is applied and then our patching framework terminates with success.

If our patching framework terminates with success, then the protocol must be formally analysed using either a model checker or a theorem prover. This process may indicate that the protocol is correct and then formal development is complete. Otherwise, the protocol is still faulty and so this verification/correction cycle is repeated again, as many times as desired.

A. Patch Method Development Methodology

Firstly, we will collect faulty protocols and then group them according to some taxonomy, either known or of ours.

Secondly, for each group, we will distinguish two sets of protocols:

1) **Development:** this class consists of a few representative protocol examples used for designing the patching method;
2) **Testing:** this class contains example protocols that will be used for testing the robustness of the method.

Thirdly, we will attempt to keep the development protocol examples as dissimilar as possible.

Fourthly, we will gather examples from different sources, e.g., books, research reports, etc., and from the Clark-Jacob library.

III. Patching a Faulty Security Protocol

To give the reader a flavour as to the kinds of reasoning involved in a method, we shall consider the NSPK Protocol, which may be informally specified as follows:

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

where $A \rightarrow B : \{M\}_{K_P^+}$ means agent A sends message $\{M\}_{K_P^+}$ to agent B, which B receives; $\{M\}_{K_P^+}$ means the message M encrypted under public key of principal P.

Roughly speaking, A initiates a session with B by sending him a nonce,[1] Na, indicating the start of a new session. After it decrypts message 1, B sends A back both the nonce Na that apparently A has sent, and a new nonce B has generated, Nb. Upon decryption of message 2, A checks that nonce Na corresponds to the nonce she generated to identify this run and then sends B back his nonce Nb, message 3.

The NSPK protocol seems right at first glance: A seems to know that he is running the protocol apparently with B, because only B could have decrypyted message 1 and sent nonce Na encrypted under A's key.[2] However, this protocol is faulty. Lowe found that an intruder could impersonate one agent holding concurrently a session with another agent [5].

Lowe's attack is shown in Figure 1.[3] There, the two instances of message 2, each of which belongs to a different session, have been enclosed within boxes, pinpointing the root of the problem. Message 2, $\{Na, Nb\}_{K_A^+}$, is being used in two independent runs, meaning that 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. The NSPK protocol therefore violates 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.*

[1] A *nonce* is a random number that has not been used in any previous run.

[2] Notice that, however, B does not know who encrypted and then sent the instance of message 1 he received.

[3] Spy(A) means the Spy impersonating A. A message incoming to Spy(A) denotes interception, while an outgoing one denotes forgery. $S{:}N$ means the Nth message associated with session S.

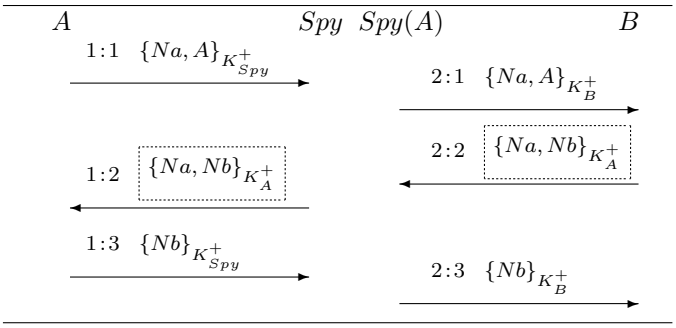

Fig. 1. Lowe's attack involves two parallel runs of the protocol

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 [5]:

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

IV. Conclusions and Further Work

We have already developed a patch method for dealing with interleaving replay attacks [4]. [4] Our interleaving-replay attack method is usually able to patch faulty protocol that violates Abadi and Needham's principle 3.

We plan on further validating our method with other faulty protocols. In addition, we will analyse other faulty protocols 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] I. Lakatos. *Proofs and refutations: The logic of Mathematical discovery*. Cambridge University Press, 1976.

[4] J. C. López-Pimentel, R. Monroy, and D. Hutter. A method for patching interleaving-replay attacks in faulty security protocols. *Electronic Notes in Theoretical Computer Science*, To appear:1–18, 2006. In Proceedings of the 2006 FLoC Verification and Debugging Workshop.

[5] 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.

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

[7] 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).

[4] An *interleaving-replay attack* is a network attack in which a valid data transmission is maliciously repeated or delayed from outside the current run of a protocol requiring that, at least, two runs overlap in execution.