Advanced Science and Technology Letters
Vol.29 (SecTech 2013), pp.25-29
http://dx.doi.org/10.14257/astl.2013.29.05

Formal Modelling of Network Security Properties (Extended Abstract)

Gyesik Lee

Hankyong National University,
Dept. of Computer & Web Information Engineering
Anseong-si, Gyonggi-do, 456-749 Korea
gslee@hknu.ac.kr

Abstract. Designing and implementing security protocols are error-prone. Moreover, security protocols are supposed to work securely even over insecure networks. Recent research progress has shown that applying formal methods can help in designing and implementing security protocols. The main objective of this paper is to present a general idea of using formal methods in the verification of security protocols. In particular, we show how to formally model intruders and security properties such as secrecy.

Keywords: Security protocols, secrecy, formal methods

1 Introduction

Designing and implementing security protocols are error-prone. Moreover, security protocols are supposed to work securely even over insecure networks, i.e., even in the presence of hostile agents that have access to the network.

Consider the following security protocol which is the core part of Needham-Schroeder's public key protocol [12]. It describes a public key mutual authentication.

$$(M1) \quad A \to B : \{A, Na\}_{pk(B)}$$
$$(M2) \quad B \to A : \{Na, Nb\}_{pk(A)}$$
$$(M3) \quad A \to B : \{Nb\}_{pk(B)}$$

It seems that the authentication process is secure because the encrypted message can only be read by a person possessing the corresponding private keys. However, 18 years after, a flaw of the protocol was discovered by Lowe [10]. The well-known man-in-the-middle attack is described as in Figure 1 below.

Interestingly, the attack is found not by a manual inspection, but by applying a formal method. Indeed, verifying security protocols manually is very hard. This is because security protocols are usually based on cryptographic primitives, and their analysis is one of the most challenging tasks. It involves many subfields such as cryptosystems, signature schemes, secure hash functions, transfer

ISSN: 2287-1233 ASTL

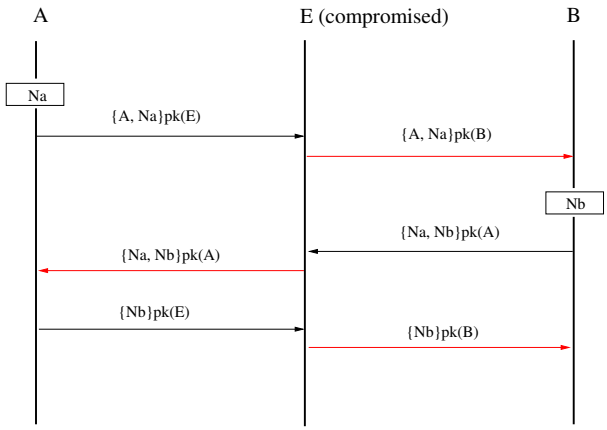

Fig. 1. Lowe's man-in-the-middle attack

mechanisms, and secure multiparty function evaluation methods. Furthermore, it is vulnerable to intruders in the network who may have control of one or more network principals. As demonstrated above, security protocols are often subject to non-intuitive attacks.

Fourtunately, recent research progress has shown that applying formal methods can help in achieving security goals such as authentication and secrecy in data exchange [1, 2, 13, 3]. Moreover, the ISO/IEC 29128 [8] states a standard which provides definitions of different protocol assurance levels where the importance of the application of formal methods are explicitly mentioned.

It seems nowadays inevitably required to verify that a security protocol satisfies its requirements based on a formal method. A formal method is based on a combination of a mathematical or logical model of a system and its requirements. Actually, the application of formal methods to cryptographic protocol analysis has been investigated since almost 30 years [11]. An important area is the development of tools for automatic verification of security protocols allowing unbounded number of sessions based on some intruder models such as Dolev-Yao model [7]. There are many tools for automated verification of security protocols such as ProVerif [4, 5] and Scyther [6], to name a few. See also Lee et al. [9] for an application of such tools.

In this paper, we present a general idea of using formal methods in the verification of security protocols. The following sections show how to formally model intruders and security properties, in particular secrecy.

2 Modelling of protocol specification

A language $\mathcal{L} = (\mathcal{V}, \mathcal{C}, \mathcal{R}, \mathcal{F})$ for security protocols should be given. It consists of symbols for constructing terms like nonces, roles, functions, etc: a set $\mathcal{V}$ of variables to store received messages, a set $\mathcal{C}$ of local constant symbols for such

as nonces and session keys, a set $\mathcal{R}$ of role name symbols, a set $\mathcal{F}$ of function symbols for such as global constants or hash functions.

Given a language, terms are defined inductively. If t_1 and t_2 are terms, (t_1, t_2) stands for composition of pairs and $\{t_1\}_{t_2}$ for encryption of t_1 by using t_2. It is assumed that encryption is perfect. There is a set $\mathcal{E}$ of equations over terms which specify identities among terms such as equations respecting Diffie-Hellman key assignments, etc.

Given a language, a protocol specification, shortly a protocol, P describes the behaviour of each of the roles such as initiator, responder, key server, etc. In the specification, the behaviour of each role is formalised as a transition system describing how to create messages, how to react to the received messages, and how to manipulate them.

It can be assumed that any agent in a role r could see the pattern of any message: nonces, agent names, session keys, pairs, encrypted messages, etc. Therefore, agent including the intruder can infer new knowledge from his initial knowledge together with the received messages. The knowledge inference can be inferred formally as follows:

$$\frac{t \in \mathcal{K}}{\mathcal{K} \vdash t} \qquad \frac{\mathcal{K} \vdash t_1 \quad t_1 = t_2 \in \mathcal{E}}{\mathcal{K} \vdash t_2}$$

$$\frac{\mathcal{K} \vdash t_1 \quad \mathcal{K} \vdash t_2}{\mathcal{K} \vdash (t_1, t_2)} \qquad \frac{\mathcal{K} \vdash (t_1, t_2)}{\mathcal{K} \vdash t_1} \qquad \frac{\mathcal{K} \vdash (t_1, t_2)}{\mathcal{K} \vdash t_2}$$

$$\frac{\mathcal{K} \vdash t \quad \mathcal{K} \vdash k}{\mathcal{K} \vdash \{t\}_k} \qquad \frac{\mathcal{K} \vdash \{t\}_k \quad \mathcal{K} \vdash k^{-1}}{\mathcal{K} \vdash t}$$

3 Modelling of operating environment

The network can be partially or completely under control of an intruder. Based on his knowledge, he can e.g. catch, eavesdrop, or fake messages. He can also interrupt or disturb the protocol running.

The initial knowledge $\mathcal{K}_A^0$ of an agent A in a role consists of e.g. the names and public keys of all agents and his secret key of his role. The initial intruder knowledge $\mathcal{K}_I^0$ consists of the initial knowledge of all untrusted agents including their secret keys. The knowledge of an agent including the intruder will grow during the running of the protocol whenever he receives or catch messages.

The configuration state at some point during running a protocol P is composed of the local intruder knowledge and the local knowledge of every possible agent A_n, where n varies over natural numbers. The list of agents is made infinite such that it reflects the fact that the intruder could initiate new session at any step and perform unlimited sessions. In the initial state, every agent is in his initial state, i.e. his initial knowledge and initial control state. If an agent A_n is not active yet, then his initial knowledge is empty.

The operational semantics of the protocol P is the description how configuration states changes during the protocol running.

- If an agent performs a new instance event, then he adds a new role instance to his state. If the agent is compromised, then he shares all the knowledge with the intruder.
- If an agent performs a sending event, the sent message m is added to the intruder knowledge. Then he moves to some state where he is waiting for another sending or receiving event, or stops.
- If an agent performs a receiving event, the agent performs some computations. Depending on the computation result, he moves to some state where he is waiting for another sending or receiving event, or stops.

A state transition is the conclusion of finitely many applications of the rules above, starting from the initial state. A trace of a protocol P is the description of any possible state transition starting from the initial state:

$$(\mathcal{K}_I^0, \langle \mathcal{K}_{A_n}^0 \rangle_n) \xrightarrow{m_1} \cdots \xrightarrow{m_\ell} (\mathcal{K}_I^\ell, \langle \mathcal{K}_{A_n}^\ell \rangle_n) \xrightarrow{m_{\ell+1}} (\mathcal{K}_I^{\ell+1}, \langle \mathcal{K}_{A_n}^{\ell+1} \rangle_n) \xrightarrow{m_{\ell+2}} \cdots$$

Here ℓ denotes the ℓ-th transition step and m_ℓ is the exchanged message at the ℓ-th transition. $\mathcal{K}_A^\ell$ and $\mathcal{K}_I^\ell$ stand for the local knowledge of the agent A and of the intruder I, respectively, at the ℓ-th step. $\langle \mathcal{K}_{A_n}^\ell \rangle_n$ is an abbreviation for the infinite list of $\mathcal{K}_{A_n}^\ell$, where n varies over natural numbers. The relationship between m_ℓ and m_i, $i < m$, is decided by the protocol specification.

4 Modelling of secrecy property

Secrecy expresses that certain information cannot be revealed to any other agent or the intruder except the honest agents who have run the protocol, even though the protocol is executed in an untrusted network. More formally, a protocol P satisfies secrecy of a message m among some honest agents $A_{n_1}, ..., A_{n_p}$ if and only if in an arbitrary trace, m cannot be inferred from the knowledge of anybody else, i.e.,

$$\mathcal{K}_I^\ell \nvdash m \quad \text{and} \quad \mathcal{K}_A^\ell \nvdash m$$

for any ℓ, where A is not one of $A_{n_1}, \ldots, A_{n_p}$.

Secrecy defined as it stands can be referred as weak secrecy, since it does not care about partial disclosure of the message content. There are also probabilistic secrecy, undistinguishability, etc. But they are out of scope here.

5 Conclusion

It is stressed that in the last years lots of important progresses has been made in applying formal methods to verifying security protocols. And there have been invented fully automated tools for the verification security protocols. We presented a general idea of using formal methods in the verification of security protocols. In particular we showed how to formally model intruders and security properties such as secrecy. In that way we tried to demonstrate that applying formal methods can help protocol designers and implementers to improve the quality of security protocols.

References

1. M. Abadi and A. D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. In *ACM Conference on Computer and Communications Security*, pages 36–47, 1997.
2. M. Abadi and P. Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). *J. Cryptology*, 15(2):103–127, 2002.
3. K. Bhargavan, C. Fournet, M. Kohlweiss, A. Pironti, and P.-Y. Strub. Implementing TLS with Verified Cryptographic Security. In *IEEE Symposium on Security and Privacy*, pages 445–459. IEEE Computer Society, 2013.
4. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *CSFW*, pages 82–96. IEEE Computer Society, 2001.
5. B. Blanchet. A Computationally Sound Mechanized Prover for Security Protocols. *IEEE Trans. Dependable Sec. Comput.*, 5(4):193–207, 2008.
6. C. J. F. Cremers. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In *CAV*, volume 5123 of *LNCS*, pages 414–418. Springer, 2008.
7. D. Dolev and A. C.-C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
8. ISO/IEC 29128. *Information technology – Security techniques – Verification of cryptographic protocols*, 2011. `http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=45151`.
9. G. Lee, H. Oguma, A. Yoshioka, R. Shigetomi, A. Otsuka, and H. Imai. Formally Verifiable Features in Embedded Vehicular Security Systems. In *First IEEE Vehicular Networking Conference, VNC 2009*, IEEE Xplore database, 2009.
10. G. Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. *Software - Concepts and Tools*, 17(3):93–102, 1996.
11. C. Meadows. Open Issues in Formal Methods for Cryptographic Protocol Analysis. In V. I. Gorodetski, V. A. Skormin, and L. J. Popyack, editors, *MMM-ACNS*, volume 2052 of *LNCS*, page 21. Springer, 2001.
12. R. M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Commun. ACM*, 21(12):993–999, 1978.
13. A. Pironti, D. Pozza, and R. Sisto. Automated Formal Methods for Security Protocol Engineering. In *Cyber Security Standards, Practices and Industrial Applications: Systems and Methodologies*, page 138–166. IGI Global, 2011.

