Analysis of Authentication Protocols with Scyter: Case Study

Očenášek Pavel

Faculty of Information Technology,
Brno University of Technology,
Božetěchova 2, 612 66 Brno, Czech Republic
ocenaspa@fit.vutbr.cz
http://www.fit.vutbr.cz/~ocenaspa/

Abstract. This paper describes one of possible implementations of method used for protocol design described in a paper from 1998 by L. Buttyán, S. Staamann and U. Wilhelm which proposes a backward search (regression) when synthesizing an authentication protocol. Furthermore, the approach presented in this paper allows a designer to define participants who are trustworthy enough to transfer information between other two participants without existence of a direct channel to achieve basic routing functionality.

Keywords: Logic, Authentication, Key Distribution, Security Protocol, Routing.

1 Introduction

Communication between the various types of electronic devices from computers to mobile phones to the payment terminal is one of the basic requirements of today. Transfer information between devices is needed, however, clearly defined rules. These rules are collectively referred to as a communication protocol. Information transmitted using these protocols through the network may have high value and are now under attack. Therefore represent a special area of security protocols deployed at places where requires a secure, encrypted communications. These are the protocols carrying sensitive information such as payment orders or authentication data. Design, testing and proper understanding of communication and security protocols to requires specific tools and environment to enable their simulation. Some instruments can detect possible errors or security vulnerabilities in the draft.

This paper presents the partial results of the research published within the thesis [4].

Security protocols are a separate area of communication protocols. They are mainly used in situations where we need to transmit sensitive information such as secret keys, authentication data or transactions on your bank account. These protocols provide the mechanisms using cryptographic (encryption), the basic security features

M.J. Smith, G. Salvendy (Eds.): Human Interface, Part I, HCII 2011, LNCS 6771, pp. 359–365, 2011.

such as authentication, authorization, integrity, confidentiality and sometimes unquestionable. Security features can be applied to each of the layers of the reference ISO / OSI model.

2 Scyter

Scyther [1] is a free downloadable tool designed primarily for verification and analysis of security protocols. It is based on an algorithm that provides a condensed representation (infinite) set of traces. This tool helps in analyzing the classes of possible attacks and protocol behavior. Tool to prove the correctness of indeterminate number of connections. It can operate in parallel with several (sub) protocols.

The program is written in Python and has a control console and graphical interface. The expected entry is the protocol description in the SPDL language. The program can determine whether they complied with safety requirements in the protocol if the safety requirements to generate and subsequently verified. It can analyze the protocol interaction by participants.

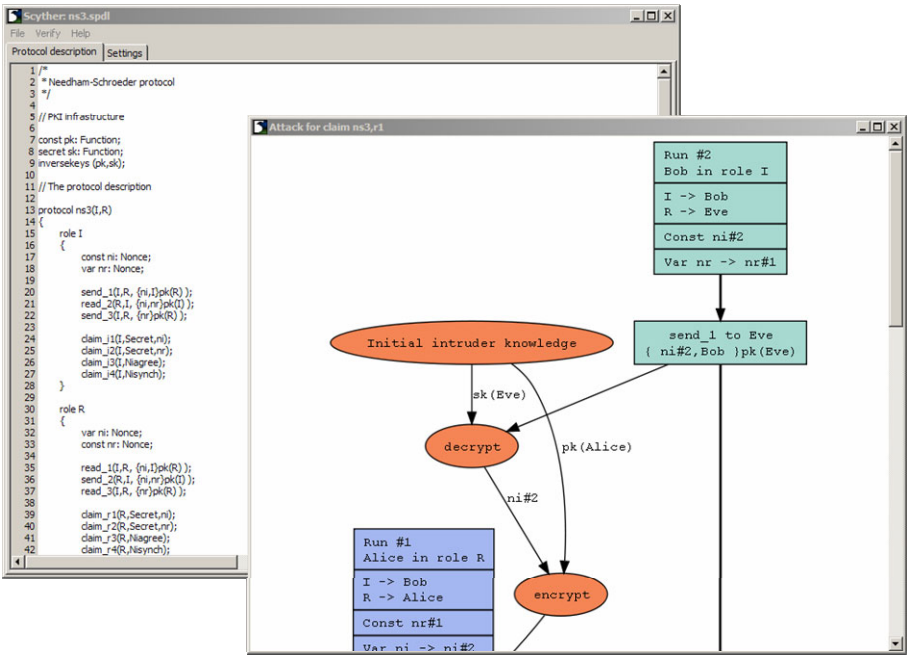

Fig. 1. Scyther GUI with the protocol example (Needham Schroeder protocol)

The program can work interactively and can not be changed at run-time conditions. It is necessary to modify the source file (eg, the built-in editor) and then run the analysis. Due to its speed, clearness and simplicity fits as a teaching aid.

3 Needham-Schroeder Public Key Protocol

As a case study, we use a version of the Needham-Schroeder protocol [2] based on asymmetric cryptography. It provides mutual entity authentication using a trusted key server and public keys. The server distributes keys upon request. The protocol was designed to communications over insecure network. The authors are Roger Needham and Michael Schroeder.

1. $A \rightarrow S: A, B$
2. $S \rightarrow A: \{ K_B, B \}_{K_S}^{-1}$
3. $A \rightarrow B: \{ N_A, A \}_{K_B}$
4. $B \rightarrow S: B, A$
5. $S \rightarrow B: \{ K_A, A \}_{K_S}^{-1}$
6. $B \rightarrow A: \{ N_A, N_B \}_{K_A}$
7. $A \rightarrow B: \{ N_B \}_{K_B}$

Fig. 2. Needham-Schroeder Public Key Protocol: Common syntax

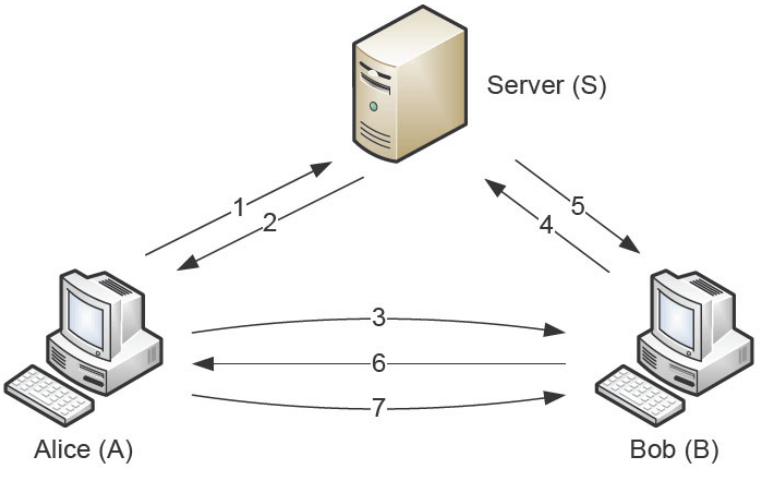

Fig. 3. Needham-Schroeder: graphical representation

4 Implementation in Scyther

The tool Scyther uses its own language SPDL (Security Protocol Language Descritpion).

We start with the keys declaration:

```
const pk: Function,
secret sk: Function,
inversekeys(pk,sk); // Pair of asymmetric keys
```

where const means the global constant. The specification of protocol behavior follows:

```
protocol NeedhamSchroederPK(I,R,S) {}
```

Within the Needham Schroeder protocol, we have to define roles for all the participating subjects (I, R, S). For example, we include the definition of role I:

```
role I {
const Ni: Nonce;    // Global variable
var Nr: Nonce;     // Local variable
send_1(I,S,(I,R)); // Message from I to S
                              // includes (I,R)
read_2(S,I, {pk(R), R}sk(S));
...
claim_I1(I,Secret,Ni);
claim_I2(I,Secret,Nr);
claim_I3(I,Nisynch);
}
```

In the final part of the protocol specification, we include the trustworthy participants of the communication, those participant we cannot trust and secret data.

```
const Alice,Bob,Server,Compromised: Agent;
untrusted Compromised;
const nc: Nonce;
compromised sk(Compromised);
```

The SPDL language is case sensitive. The comments can be written in different way (like in C language). One-line comment # or //, multi-line comments starts with /* and end with */ . The whitechars (newline, space) are ignored.

5 Verification Results

The protocol specification in SPDL language is an input for Scyther verification.

If we have a final protocol described and the main parameters are set, we can start the verification in the menu: *Verify / Verify protocol (F1).* A window appears, where each row represents one requirement (or claim). On the line the information is listed in the order: name of protocol, role (party identification), ID requirements, and the request type parameter, informing the state to find an attack, and the button to display graphic attack.

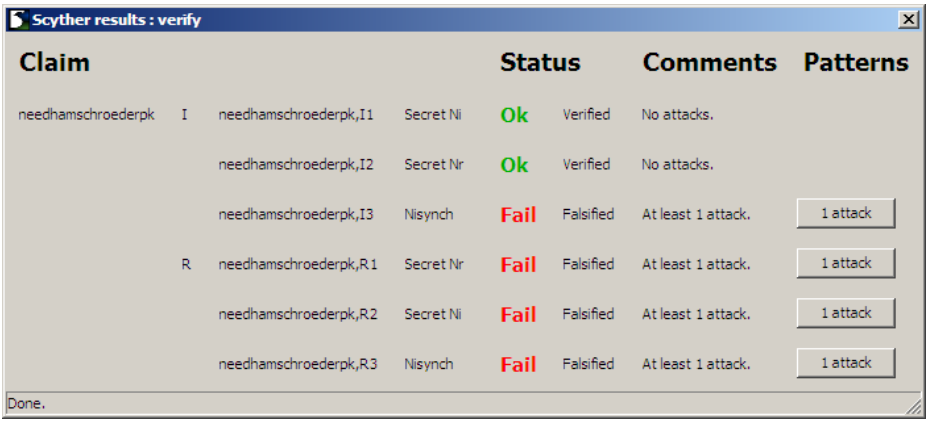

Fig. 4. Scyther: verification results

Often, the program finds does not find the attack and ends with the result "No attack within bounds". In the verification process Scyther scans tree of all possibilities but is limited by the number of runs (program settings). It is important to choose an appropriate value. Too large value means more time needed for verification. A small amount may case that the attack is not found.

If you choose to show the attack found or any of the roles, the new window is opened. The main elements are rectangles connected by arrows. Arrows represent the order. Rectangle shows the new run, run event and the event caused by the security condition. The vertical axis represents the running protocol (roles occurence).

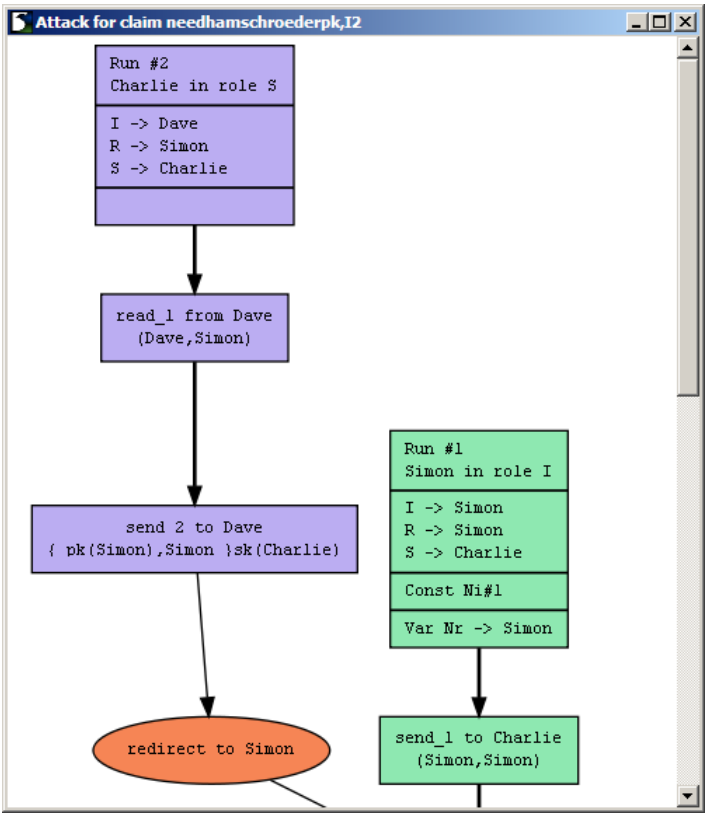

Fig. 5. Scyther: attack found

The resulted attack on Needham-Schroeder PK protocol found by the Scyther can be discussed with the attack presented in [3].

6 Conclusions

The aim of the paper was the presentation of Needham Schroeder specification in SPDL language. This language is the input for Scyther tool. This tool can be easily used for security protocol verification. The main advantage of the Scyther tool are the verification speed and the support of large number of communication connections. It can also verify the (sub)protocols in parallel. The main disadvantage is that we cannot change the settings within the protocol run and it is necessary to compile the protocol before we start the analysis.

Acknowledgements. This study was supported by the Ministry of Education, Youth and Sports of the Czech Republic (MSM0021630528 and MSM0021630503) and projects FIT-S-11-1, FIT-S-11-2 and FEKT/FIT-S-11-2.

References

1. Cremers, C.J.F.: The scyther tool: Verification, falsification, and analysis of security protocols. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 414–418. Springer, Heidelberg (2008)
2. Needham, R., Schroeder, M.: Using Encryption for Authentication in Large Networks of Computers. CACM 21/12 (December 1978)
3. Lowe, G.: An Attack on the Needham-Schroeder Public-Key Authentication Protocol. Information Processing Letters (56), 131–133 (1995)
4. Mikus, P.: Tools for Environment for the Simulation of Communication, Master's thesis, FIT Brno University of Technology, Brno, Supervisor: Pavel Ocenasek (2010)