Jan Długosz University of Częstochowa
Scientific Issues, Mathematics XI, Częstochowa 2006

Verifying Untimed Version of The WMF Protocol Using Networks of Automata*

Kamil Grondys, Mirosław Kurkowski

Anna Sowik, Izabela Szczypior

Institute of Mathematics and Computer Science
Jan Długosz University of Częstochowa
al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland
e-mail: m.kurkowski@ajd.czest.pl

Abstract

In this paper we present some results of symbolic verification of untimed version of The Wide-Mouth Frog Protocol. This protocol is designed for achieving authentication between communicating sides in the computer network and exchange a new session cryptographic key. For our investigation we model executions of this protocol by a network of synchronized untimed automata. We invesigate suitable protocols properties by testing reachablity of some distinguished states in the defined network. We use VerICS [5] - the symbolic model checker - for searching in our network.

Keywords: Cryptographic authentication protocols, verification, model checking.

*Extended version of a talk presented at the X Conference "Applications of Algebra in Logic and Computer Science", Zakopane, March 6–12, 2006. This research is partially supported by the Ministry of Science and Information Society Technologies under the grant number 3 T11C 011 28.

1. Introduction

Cryptographic authentication protocols [3, 11] are specially designed tools for achieving authentication in largely distributed computer network. These protocols are precisely defined sequences of actions (communication and computation steps) which use encryption and decryption. It is well known that the design of authentication cryptographic protocols is not easy. We can find many examples of errors in protocols in the literature (e.g. see [9]). Due to this it is necessary to have some methods of analysis and properties verification of these protocols. In a case of formal investigations sometimes special models are defined and usually searched automatically looking for states which correspond with properties or errors of the correctness. It is obvious that the proposed models should express all important properties and ideas of verified protocols. Specification of characteristics of cryptographic protocols and their later verification is one of the hardest challenges that the protocols designers must deal with. The difficulty here is not only in the adequate modeling behavior of the "honest sides" but also in deceiving the Intruder. To reflect all possible behaviors the constructed models are usually quite large, impossible to be searched trough by traditional simulation methods. The best results of verifications are given by methods consisting of constructing and adequate searching of various formal models reflecting behavior of participants of the given protocol [1, 2, 4, 7, 8, 10].

In [6] a new methodology of formal, symbolic verification of untimed crypto protocols has been introduced. In suitable finite space all executions of a given protocol are automatically computed (there are finite numbers of executions in finite space). Each of these executions and knowledge of communicating to each of the users are modeled by a network of synchronized untimed automata. In the set of all executions we distinguish some of them which correspond with the flawed ones. Automatic verification/searching of this network shows if there exists the sequence of executions which goes to the suitable state.

The rest of this paper is organized as follows. In the second section we describe and discuss the WMF Protocol. Using this protocol we want to show the method of verification of correctness propert for cryptographic protocols and in particularly its susceptibility to an attack by the strange side - Intruder. Next, we briefly introduce the

usage of the veryfying system of the correctness and safety of cryptographic protocols. Lastly, we present the experimental results of the check of availability received after introduction networks of automata to the verifying system.

2. The Wide-Mouth Frog Protocol

Protocols of exchange of session key are designed for establishing a new secret cryptographic key, which later enables secure (encrypted) exchange of information between communicating sides. This key is called a session key, because it is used only in one session of contact. Session keys are useful, because they exist only during the communication period. As it was mentioned before the authentication protocols are used for confirming an identity of communicating sides.

The Wide-Mouth Frog Protocol is the communication cryptographic protocol applied to achieve authentication between two sides and exchange a new session key. This protocol was created by Michael Burrows [3]. It is probably the simplest symmetrical protocol of management the key and very good example for scientific investigations. In this protocol two users and third trusted side take part; the third trusted side is usually the main server which has to facilitate authentication and exchange session key, which will be later used by sides to code the sent messages. Apart from each other, two sides A and B use a common secret key with Trent (trusted server). These keys are used for a distribution of the keys and are unused for coding of any message sent between the sides. By the use of two messages a sessional key is sent from A to B.

By A and B we mean principals (users) of protocol, by S third trusted side (trusted server). This protocol consists of two steps, its aim is the authentication of the principal A by the principal B and the exchange of session key. Both sides A and B use a common secret key shared with S.

The idea of protocols execution:
$1.A \rightarrow S : I_A, < T_A, I_B, K >_{K_{AS}}$
$2.S \rightarrow B :< T_S, IA, K >_{K_{BS}}$

Here:
A, B are the principals of protocol, S is an authentication server, T_A is timestamp generated by A , T_S is timestamp generated by S, I_A is the name of the principal A, I_B is the name of the principal B, K is the new session key, K_{AS} is the private key shared with A and S, K_{BS} is the private key shared with B and S. Symbols $<$, $>$ mean encryption under a suitable key.

Protocol analysis:

1. In the first step of protocol the principal A joins his timestamp T_A with participant's identifier I_B with whom he wants to communicate I_B and random sessional key K and he encrypts the whole message using the secret key shared with (common from) S. Next A sends encrypted (coded) package to trusted server together with his name I_A.

2. In the second step S decodes a message received from A, and then joins new timestamp T_B, name A (IA) as well as random sessional key K. After constructing such a message it is encrypted by the secret key shared from B, and sent to participant B.

In this protocol the most essential assumption is as follows: the participant B believes in participant's A sufficient competence of generating suitably good sessional key. One should turn attention to the fact that it is very difficult to generate large random numbers. It would cross the possibility of participant A. The timestamp has to guard this protocol against attacks by repetition.

3. Modeling

The protocol described by us is not however, as it turns out, resistant to intruder's attack. Verification of correctness property of cryptographic protocol investigated by us runs in following steps:

1. We create knowledge networks automata for particular principals and for Intruder and protocol realizations automata.

2. We translate these automata into the special formal language and next put to the model checker VerICS.

3. We transform the networks of automata to Boolean formula, and next the verifying system checks its availability.

At present, model checking investigated by us is considered as the most spectacular usage of theoretical computer science in computer practice.

The correctness of this communication protocol is the most important matter, which has to be decided before it will be hit to public use. The guarantee of this can be only the use of entirely automatic method, that means a programme which, when receiving the tested protocol on entry, will answer to the question whether it guarantees full safety and resistance on attack or not. The main idea of model checking depends on performance of this protocol as automata, which states make up the studied model and performance of specification as logical formula. The model is checking algorithmic whether a formula is true or not. It may be done directly or by constructing automata which accept all models for given formula and also checks if a model set by the programme can be accepted.

4. Experimental results

In this section we give experimental results using SAT-based model checker VerICS [5]. We verify the correctness property for the WMF protocol.

The computational structure (see [6]) is modeled by 15 automata for all the executions of the participants and 12 knowledge automata.

The experiments consisted in checking reachability (in the product automaton) of the final states of the automata representing attacking executions.

The experimental results (obtained using PC Pentium 2,4 GHz, RAM 512 MB under Linux) are shown in the table below.

Table 1. Experimental results for UTWMF Protocol

Trace lenght	Variables	Literals	Clauses	Time (s.)	Result
2	2061	3633	12601	0,21	UNSAT
3	3513	5479	15639	0,43	UNSAT
4	4965	6325	19677	0,64	UNSAT
5	6417	7771	22715	0,86	UNSAT

5. Conclusion and future work

In this paper we have presented some results of symbolic verification of untimed version of The Wide-Mouth Frog Protocol. For our investigation, we have modeled executions of this protocol into a network of synchronized untimed automata. We have invesigated suitable protocols properties by testing the reachablity of some distinguished states in the defined network. For searching in our network we have used VerICS [5] – the symbolic model checker.

Our next step is to see what are the limits of our method in terms of the number of sessions as well as in the number of participants for all the protocols which satisfy our restrictions. Then, we are going to relax the assumption on non-nesting ciphers and again conduct experiments with multi-session and multi-participant security protocols.

References

[1] A. Armando et al. *The AVISPA tool for the automated validation of internet security protocols and applications.* In: CAV 2005, 17th Int. Conf. on Computer Aided Verification, Edinburgh, Scotland, UK, pp. 281–285, 2005.

[2] G. Bella, L.C. Paulson. *Using Isabelle to prove properties of the kerberos authentication system.* In: H. Orman, C. Meadows (editors), Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997. http://dimacs.rutgers.edu/Workshops/Security/

[3] J.A. Clark, J.L. Jacob. *A survey of authentication protocol literature.* Technical Report 1.0, 1997. http://www.cs.york.ac.uk jac/papers/drareview.ps.gz

[4] E.M. Clarke, S. Jha, W. Marrero. *Verifying security protocols with Brutus.* ACM Transactions on Software Engineering and Methodology, **9** (4), 443–487, 2000.

[5] P. Dembiński et al. *VerICS : A tool for verifying timed automata and Estelle specifications.* In: Proc. 9th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03), vol. 2619 of LNCS, Springer-Verlag, pp. 278–283, 2003.

[6] M. Kurkowski, W. Penczek. *Verifying cryptographic protocols modeled by networks of automata.* In: Proceedings of XV CS&P (Concurrency, Specification and Programming), Humboldt University Press, Berlin, pp. 292–303, 2006.

[7] G. Lowe. *Breaking and fixing the Needham-Schroeder public-key protocol using FDR.* In: Proceedings of TACAS, Springer Verlag, pp. 147–166, 1996.

[8] G. Lowe, B. Roscoe. *Using CSP to detect errors in the TMN protocol.* IEEE Transaction on Software Engineering, **23**, No. 10, 659–669, 1997.

[9] G. Lowe. *A Family of Attacks upon Authentication Protocols.* Technical report 1997/5, Department of Mathematics and Computer Science, University of Leicester, 1997.

[10] C. Meadows. *The NRL protocol analyzer: An overview.* Journal of Logic Programming, **26**, (2), 13–131, 1996.

[11] R. Needham, M. Schroeder. *Using encryption for authentication in large networks of computers.* Communications of the ACM, **21**, (12), 993–999, 1978.