# THE SECURITY PROTOCOL DESIGN USING GENETIC ALGORITHMS PARADIGMS

Ing. Pavel OČENÁŠEK, Doctoral Degree Programme (2)
Dept. of Information Systems, FIT, BUT
E-mail: ocenaspa@fit.vutbr.cz

Supervised by: Prof. Miroslav Švéda

## ABSTRACT

This paper deals with the evolutionary approach for designing security protocols. In the first part, the introduction to security protocols is outlined followed by the explanation of principles of genetic algorithms. The second part presents the algorithm for designing security protocols. The whole design is determined by the genetic evolution. The genetic approach must be connected with the heuristic strategies to satisfy some design expectations. In the addition to the previous work, the idea of automatic designing tool and other directions in the topics research are outlined. The paper relates closely to the topic of the author's PhD thesis.

## 1 INTRODUCTION

Security protocols (cryptographic protocols) are intended for secure communication of subjects over an insecure network. A goal is to prevent a spy from reading the contents of messages addressed to others (secrecy). Most security protocols also guarantee an authenticity. This means that if a message appears to be sent by subject A, then A sent exactly this message and it contains the indication of its integrity. To ensure protocol security there are many ways for their verification [1] [2] [5].

## 2 SECURITY PROTOCOLS

Security protocol is a sequence of instructions that describes how participated subjects should act to achieve some goal. Protocols are often described using informal notation, for example as a sequence of instructions explaining the actions taken by the subjects.

Each step describes an event $A \rightarrow B: X$, which states that A exchanges the message X with B. Messages consist of atoms, like subject names and nonces (randomly generated strings), and are composed by tupling. Moreover, messages may be encrypted using keys of subjects.

## 3 PROTOCOL BEHAVIOR

For describing a protocol we decided to use the BAN logic. BAN logic (authors are Burrow, Abadi, Needham) [3] [4] is a modal logic with primitives that describes the beliefs of subjects involved in a cryptographic protocol. Using the inference rules of BAN logic, the evolution of the beliefs of subjects during a cryptographic communication can be studied. Here we present some typical rules.

The BAN-formalism is built on three sorts of objects: the subjects involved in a security protocol, the encryption/decryption and signing/verification keys that the subjects possess, and the messages exchanged between subjects. The notation {M}K denotes a message encrypted using a key K. For a symmetric key K we have {{M}K}K = M for any message M i.e., decrypting with key K a message M that is encrypted with K reveals the content M. For a key pair <EK, DK> of a public encryption key EK and a private decryption key DK it holds that {{M}EK}DK = M for any message M. Likewise, for a key pair <SK, VK> of a private signing key SK and a public verification key VK it holds {{H}SK}VK = H for any hash value H. Hash values are obtained by aplying one-way collision-free hash-function. Proving the hash value H(m) of a message m is a mean to demonstrate that m is known, without revealing it.

In BAN we have a number of operators describing the beliefs of subjects for which the usual modal properties apply. For example P believes (A $\rightarrow$ B) $\rightarrow$ (P believes A $\rightarrow$ P believes B). On the top of that we have the operators *sees* and *possesses*. The following rules are the illustration of some of the authentication and encryption rules:

*(1)* $P \text{ believes secret } (K, P, Q) \rightarrow P \text{ sees } \{X\}K$

$\rightarrow P \text{ believes } Q \text{ said } X$

*(2)* $P \text{ believes belongs.to } (VK, Q) \rightarrow P \text{ sees } \{X\}SK$

$\rightarrow P \text{ believes } Q \text{ said } X$

*(3)* $P \text{ believes fresh } (X) \rightarrow P \text{ believes } Q \text{ said } X$

$\rightarrow P \text{ believes } Q \text{ believes } X$

*(4)* $P \text{ possesses } DK \rightarrow P \text{ sees } \{X\}EK \rightarrow P \text{ sees } X$

Intuitively, rule (1) says that if a subject P believes that it shares the symmetric key K with a subject Q, and subject P receives a message encrypted under K, then the subject P believes that Q once said message X. This rule addresses the symmetric encryption. In a similar way, rule (2) models digital signatures. If a subject P believes that the verification key VK belongs to a subject Q, then P concludes, confronted with a message or hash encrypted with the corresponding secret key SK, that a message or hash originates from the subject Q. Regarding rule (3), if a subject P believes that certain information is new, i.e. constructed during the current protocol run, and P furthermore believes that Q conveyed this information, then P concludes that the subject Q believes himself this information. According to (4), if a subject P sees an encrypted message and P possesses a decryption key then P can read the message itself.

## 4 GENETIC APPROACH

Genetic algorithms are inspired by natural evolution. These algorithms encode a potential solution to a specific problem as a chromosome and perform recombination operators on this data structure to propose new solutions. The reproduction process is performed in such a way that the chromosomes representing better solutions are given more chances to reproduce than those representing worse solutions. The goodness/fitness is defined according to the desired property of the final solution..

## 5 PROTOCOL DESIGN

The security protocol is a sequence of rules and conventions that define the communication framework between two or more subjects. These rules may be elementary instructions that consist of operations such as sending a message, encrypting/decrypting a message with a secret key. Additionally, for our approach we should mention a new basic operation: adding a nonce (random number) to the set of knowledge. This allows subjects to send in the rest of protocol these nonces as often needed in cryptographic protocols.

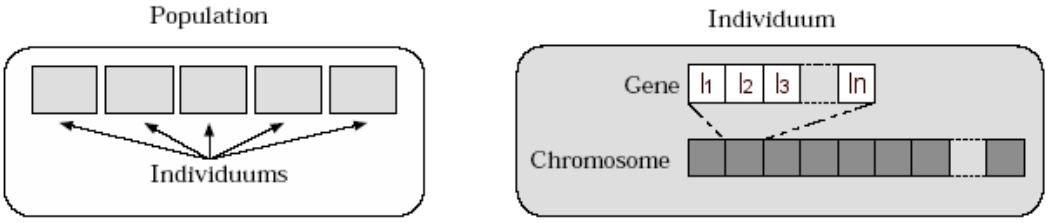


**Fig. 1:** *Structure of a Population.*
*Protocol (individual) is encoded as a sequence of instructions ($I_1$, …, $I_n$).*

While the protocol runs, each instruction affects the corresponding sets of knowledge and belief, as explained in chapter 3.

From the evolutionary-optimization point of view, the sequence of protocol instructions is represented by a chromosome of variable length. There could be generated as many random chromosomes as required for initial population. Each chromosome represents a different protocol and its fitness is computed by simulating its run, according to the changed sets of knowledge and belief.

The following algorithm describes the whole flow we use to design security protocols. The principles can be found in [6] and are explained to appreciate the idea of automatic design:

1. Security goals – in this first step we describe what should or shouldn't contain sets of knowledge and belief for each involved subject, which message is secret and cannot be sent unencrypted, which message can never be sent, number of recursive encryptions etc.

2. Initial population – randomly generated protocols (instruction sequences) are encoded into chromosomes. In this step the fitness of all individuals is calculated. This value depends highly on the security presumptions and satisfaction in each state of the protocol run. The use of additional verification tools for finding certain flaws might be helpful.

3. Parents for mating – like in standard genetic algorithms, the individuals with the best fitness are chosen to be parents for mating.

4. Crossover – the choice of the right locations in chromosomes for crossover is very important. It may highly affect the chances for a generation of chromosome with better fitness. The basic idea for mating is that two chromosomes may be mated at selected states (instructions) if both have corresponding sets of knowledge and belief. This means we have to prove that after crossing the instruction strings, the rest of protocols make sense for both individuals.

5. Performing mutation – avoiding jamming in local minima, the mutation is very useful step. By performing atomic changes in the instructions, mutation may affect both sets of knowledge and belief.

6. Replacing offspring to population - the produced individuals are replaced to the new population and the evolution process starts over again from step 3.

The design is completed when some chromosomes (with best fitness) satisfy the initial presumptions. The result is the chromosome with the best fitness which can be interpreted as a sequence of basic instructions in the cryptographic protocol. Because the automatic design usually cannot satisfy all the security requirements, the protocols should be additionally verified by some of the common verification techniques [1].

## 6 AUTOMATED TOOL

The practical output of our research should be a tool for designing security protocols. This tool would be highly automated. The most important step in the design is to specify the initial security presumptions and requirements. The designer decides which components of the messages are allowed to be sent through the communication and which are forbidden to transfer. There is also important task to correctly initialize the sets of knowledge and beliefs of the participants. Also the final sub-sets that should appear in the participants' knowledge and beliefs sets at the end of the protocols run are correspond with the security of protocols in the evolution process. This means the more precise we specify the final sub-sets, the more secure protocol we obtain.

The whole design is automated and evolves according the algorithm specified above. The result of the design is the instruction sequence that mostly satisfies security requirements.

## 7 CONSLUSIONS

The paper proposed the approach based on genetic algorithms that is used for generation new security protocols. The use of BAN logic for describing subjects' knowledge and belief seems to be sufficient for the present work. The usability for more complex protocols requires addition of heuristic strategies. Our future work will focus on research of such strategies and completing the automated tool for generation of simple security protocols. The research relates closely to the author's PhD topic: Designing and verification of security protocols.

## ACKNOWLEDGEMENTS

The paper has been prepared as a part of the solution of GAČR project No. 102/05/0723: A Framework for Formal Specifications and Prototyping of Information System's Network Applications and GAČR project No. 102/05/0467: Architectures of Embedded Systems Networks.

## REFERENCES

[1] Ma, L., Tsai, J.: Formal Verification Techniques for Computer Communication Security Protocols. Handbook of Software Engineering and Knowledge Engineering, World Scientific Publishing Company, 2000, p. 23

[2] Shmatikov, V., Stern, U.: Efficient Finite-State Analysis for Large Security Protocols. CS Department of Stanford University, 1998, p. 10

[3] Agray, N., van der Hoek, W., de Vink, E.: On BAN Logics for Industrial Security protocols. CCEMAS, 2001, p. 8

[4] Abadi, M., Tuttle, N.: A Semantic for a Logic of Authentication. Proceedings of the ACM Symposium on Principles of Distributed Computing, 1991, pp. 201-216

[5] Očenášek, P.: On Inductive Approach in Security Protocol Verification. Proceedings of the 10th Conference and Competition STUDENT EEICT 2004, Brno, p. 272-276, ISBN 80-214-2635-7