Designing Secure Communication Using Evolutionary Approach

Pavel Ocenasek
Brno University of Technology
Bozetechova 2
612 66 Brno, Czech Republic
+420-604922818
ocenaspa@fit.vutbr.cz

Jiri Ocenasek
Kimotion Technologies
Leuvensesteenweg 200
B-3370 Boutersem, Belgium
+33-484067461
jiri@ocenasek.com

ABSTRACT
This paper proposes an evolutionary method that serves for designing security protocols. The principles of security protocols are outlined, followed by the outline of the evolutionary optimization framework and the techniques that can be used to automatically evolve basic security protocols.

Categories and Subject Descriptors
C.2.2 [**Computer-Communication Networks**]: Network Protocols – *protocol verification.*

General Terms
Algorithms, Performance, Design, Security, Verification.

Keywords
Security Protocols, Authentication, Communication, Design, Verification.

1. INTRODUCTION
The increasing popularity of distributed computing and applications like internet banking and electronic commerce has created both tremendous risks and opportunities. Many of risks stem from security breaches, which can be ruinously expensive. One of the cornerstones of security is the use of security (cryptographic) protocols in which information is exchanged in a way intended to provide security guarantees. Security protocols are becoming widely used and many new protocols are being proposed. Since security protocols are notoriously difficult to design, computer assistance in the design process is desirable.

In this paper we describe an evolutionary technique that might be useful in designing new security protocols. This approach supports the principles of subjects' knowledge and belief [5] [2].

2. MOTIVATION & STATE OF THE ART
There has been much work done in the field of genetic algorithms and security protocols, and their design, but in both areas separately. Current techniques for creating security protocols deal with the human interaction and knowledge [6] [7]. The designer states the security goals and finally creates a corresponding security protocol. As this paper shows below, this process could be highly automated with our new approach

Although the main idea of the presented approach – the use of evolutionary techniques in the security protocols design in general – was introduced by Pavel Ocenasek in 2005 [9] [10], the concrete details of using Genetic Programming and some designing aspect are outlined in this paper.

GECCO'06, July 8–12, 2004, Seattle, WA, USA.

3. SECURITY PROTOCOLS
A protocol is a recipe that describes how subjects should act to achieve certain goal. Protocols are often described using an informal notation, for example as a sequence of instructions explaining the actions taken by the subjects. Each step describes an event A → B: X, which states that A exchanges the message X with B. Messages consists of atoms, like subject names and nonces (randomly generated strings), and are composed by tupling. Moreover, messages may be encrypted using keys of subjects.

3.1 Introduction
The general events of security protocols can be decomposed into elementary instructions. These instructions are e.g. sending a message, or 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 these nonces in the rest of protocol as often needed in cryptographic operations to guarantee a freshness of a message. The example of Needham-Schroeder symmetric protocol follows:

1. $A \rightarrow S$: A, B, Na
2. $S \rightarrow A$: $\{ Na, B, Kab, \{ Kab, A \}_{Kbs} \}_{Kas}$
3. $A \rightarrow B$: $\{ Kab, A \}_{Kbs}$
4. $B \rightarrow A$: $\{ Nb \}_{Kab}$
5. $A \rightarrow B$: $\{ Nb - 1 \}_{Kab}$.

The messages consist of atoms, like subject names and plain texts. Moreover, messages may be encrypted using keys of subjects.

3.2 Protocol Behavior
For successful use of the proposed design approach, we must understand the behavior of security protocol when it is running.

This helps us also in evaluating a quality of each protocol and finding a measure of satisfaction of the presumptions.

While the protocol runs, the messages are exchanged between involved subjects. Let us trace the set of "knowledge" and "belief" for each subject. Each operation affects at least one of this set. The set of knowledge for the specified subject contains all the messages and elements that are known to the subject. Similarly, the set of belief contains all the information about what subject beliefs that other subjects know (have in their set of knowledge). Both sets could be traced while the protocol runs and they are the basic criteria for evaluating the quality of a protocol. For the formal notation we could use some of the modal logics. For example BAN logic [5] [2] [3] directly deals with knowledge and beliefs. The verification process with quality measurement are separate area of interest and the reader is kindly referred to the appropriate literature in references [1] [12].

3.3 Protocol Primitives in the Design Process

During the process of designing a security protocol a number of decisions must be made: How many subjects are involved in communication? What elementary instructions and cryptographic operations we plan to use? And, finally, what will be the sequence of instructions that causes exchange of messages and affects subjects' knowledge and belief?

When stating the involved subjects and elementary operations that could be (not always must be) used in the generated protocol, we have to follow some preparation steps. As mentioned above, the basic events could be usually separated into elementary instructions. Furthermore, subjects' sets of knowledge could be imagined like instruction registers that could be read or written. The subjects has as many register as there are elementary and composed messages.

For example, the operation

```
A → B: { Kab, "A" }Kbs
```

could be finally decomposed into the following elementary instructions:

```
A.concatenate(Kab, "A") into M;
A.encrypt(M, Kbs) into N;
A.send(N) to B
```

When using register-notation, we could write:

```
A.write[M] = A.read[Kab] . "A"
A.write[N] =
    A.encrypt(A.read[M],A.read[Kbs])
B.write[N] = A.read[N]
```

Where N states the new symbol {Kab,"A"}Kbs .

This elementary 3-address code could be further rewritten into elementary instructions using operations that deal only with 1 register.

After choosing the elementary instructions that would be used in the design process, we generate a set of such instructions with all possible arguments that could affect the corresponding sets of knowledge and belief. Note that the subject's set of beliefs has a semantic meaning and is updated as the protocol is being traced in the evaluation step.

4. AUTOMATED DESIGN PROCESS

The whole protocol design process is generally NP-complete. In the past two decades a number of optimization techniques have been proposed to find reasonably good solutions to NP-complete problems. These include Evolutionary strategies, Genetic algorithms, Genetic programming, and others.

4.1 Genetic algorithms

Genetic algorithms (GA) [4] are so-called "uninformed" search algorithms, i.e., they do not incorporate any special knowledge on the problem they solve. All problem specific knowledge is included in the fitness (objective) function that evaluates each solution produced by the GA. The better the solution's fitness, the more likely it survives and reproduces. The following algorithm describes the whole GA flow we use to design security protocols:

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

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

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

4. Performing crossover – the choice of the right locations in chromosomes for crossover is very important. 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 ensure that after crossing, the rest of protocols makes 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 evaluated and replaced to the new population and the evolution loop starts over again from step 3.

The whole design is finished when some individuals (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 operations in the cryptographic protocol.

4.2 Genetic Programming

A Genetic Programming (GP) [4] can be considered as a specialized form of genetic algorithm, which manipulates very specific type of solutions using modified genetic operators. A strong motivation for using such hierarchical representation was

the problem of applying crossover to variable-length chromosomes. Security protocols are obviously of variable sizes. The crossover operation can be used to interchange randomly chosen branches of the parents' trees.

4.3 Heuristic techniques

The described evolutionary methods can be advanced using additional heuristic techniques that are now under our research. We give several examples.

As mentioned, the interchange of parts – subtrees – of security protocols is a difficult process. There is a need for heuristic technique which recognizes what parts could be interchanged without altering the semantics of instruction sequences.

Another example of a problem for employing heuristic techniques is the ordering of instructions. For example, an elementary instruction that needs to read from some register should be executed only after this register is filled with valid data. Thus, read operations must come after write operations. This could be guaranteed by linking these two operations together $\Rightarrow$ the sequence of operations cannot be changed, but other non-destructing instructions might be inserted between them. Alternatively, the problem could be partially solved using proper calculation of fitness function – protocols that contain instructions with invalid operands gain smaller fitness values.

Some heuristics can also serve to limit the size of searched space. The final design goal is known in advance – say "we require that when the protocol finishes, some subject will believe in something". Therefore, heuristic techniques can force subjects to send concrete goal-oriented messages and perform concrete goal-oriented operations.

To calculate the fitness of solutions of generated protocols we have to trace the instruction sequences to simulate their outcomes. In security protocols, we use various measures to quantify a quality of generated protocol. The design and verification of protocol are two different processes. The higher is the fitness, the less security flows exist and the more initial presumptions are satisfied. Methods that could be used to evaluate the fitness were introduced recently, e.g. in [7] [10].

5. CONCLUSIONS

The paper describes the approach that serves for designing security protocols. The idea of security protocols is outlined as well as the evolutionary methods that could be used to design new prescriptions for secure communications. The main part is devoted to the use of genetic programming, which seems to be best suited for performing automated design process. Description of the whole process in details is beyond the scope of this paper, therefore some steps were explained in general.

Our future work will focus on advancing the current techniques and improving the heuristic techniques, which are necessary to achieve initial presumptions in real design applications.

6. ACKNOWLEDGMENTS

This research has been supported by the Grant Agency of the Czech Republic through the following grants: GACR 102/05/0723: A Framework for Formal Specifications and Prototyping of Information System's Network Applications, GACR 102/05/0467: Architectures of Embedded Systems Networks, GACR 102/05/H050: Integrated approach to education of PhD students in the area of parallel and distributed systems, and by the Czech Ministry of Education in frame of MSM 0021630503 Research Intention MIKROSYN: New Trends in Microelectronic Systems and Nanotechnologies.

7. REFERENCES

[1] Abadi M., Needham R., Prudent Engineering Practice for Cryptographic Protocols, In: Proceedings of the 1994 IEEE Symposium on Security and Privacy, IEEE Computer Security Press, 1994, p. 122-136

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

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

[4] Bentley, P. J., Evolutionary Design by Computers, Morgan Kaufmann Publishers Inc., 1999, San Francisco, CA

[5] Burrows M., Abadi M., Needham R., A Logic of Authentication, ACM Transactions on Computer Systems, 8 (1), 1990, p. 18-36

[6] Gritzalis S., Security protocols over open networks and distributed systems: Formal methods for their Analysis, Design and Verification, University of Aegean, Greece, Computer Communications, 22 (8), 1999, p. 695-707

[7] 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

[8] Nossal, R., Galla, T.M., Solving NP-Complete Problems in Real-Time System Design by Multichromosome Genetic Algorithms, Vienna Institut of Technology, AT, 1997

[9] Očenášek, P.: Evolutionary Approach in the Security Protocols Design, In: Proceedings of the First European Conference on Computer Net-work Defence, University of Glamorgan, GB, Springer, 2005, p. 147-156, ISBN 1-84628-311-6

[10] Očenášek, P., The Security Protocol Design Using Genetic Algorithms Paradigms, In: Proceedings of the 11th Conference and Competition STUDENT EEICT 2005, Brno, CZ, p. 576-580

[11] Očenášek, P., Towards Selected Problems in the Security Protocol Design and Verification, 1st Doctoral Workshop on Mathematical and Engineering Methods in Computer Science MEMICS 2005, Znojmo, CZ, p. 9

[12] Shmatikov, V., Stern, U., Efficient Finite-State Analysis for Large Security Protocols, In: Proceedings of the 11th IEEE Computer Security Foundation Workshop, IEEE Computer Society Press, 1998, p. 10