Tatra Mt. Math. Publ. **29** (2004), 137–145

TATRA MOUNTAINS
Mathematical Publications

FORMAL ANALYSIS AND PARALLELISM IN PROTOCOLS

Rastislav Krivoš–Belluš

ABSTRACT. This article lies in the area of security protocol analysis, focusing on data confidentiality and authentication. Security protocols are essential for protecting electronic information. In literature there are many protocols which have been found as insecure in some aspects. Formal analysis is checking the protocol's security (fullfiting its aims, e.g., authentication) and therefore helps in construction of protocols.

We will discuss modeling parallelism. The aim is to capture sequential runs and parallelism in protocols' running using modal checker (type of formal analysis). Role confusion attacks are the most difficult attacks to analyse—either theoretically or in practical proving (because of huge number of combinations).

1. Introduction

Establishing secure communications between computer systems through a public medium such as the Internet and being able to prove that these communications are indeed secure, is not only a growing area in academic research, but also one of great importance in society today. Suppose you have Internet banking facilities and can therefore access your account remotely from home. When you request the website from your bank's server system, how do you know it is the bank's genuine website, as opposed to a fake one? You could naively enter all your bank details with the belief that it will all transfer across the network securely encrypted, when in actual fact your information has been diverted to a malicious intruder unencrypted (or encrypted with a key known by the intruder!). These sorts of security problems (and many derivatives like them) are not the result of flawed cryptography; the problem is that cryptography alone does not always suffice to address security properties such as authentication: For these, we need security protocols.

2000 Mathematics Subject Classification: 94A60, 68Q60.
Keywords: formal analysis, protocol, security, attacks.

Supported by the grant Vega 1/0385/03

Many examples have shown that security protocols may contain subtle errors that remain undiscovered for years, even when the protocols are carefully designed. Therefore, we cannot rely on ad hoc and informal ways of reasoning, and it is crucial to apply formal methods to security protocol analysis. During twenty years of research lot of protocols had been proved as correct (secure) by some type of formal analysis, but attacks on them were discovered. Typical example is Needham-Schroeder Public Key Protocol [10], on which attack was discovered couple of years after proving his security using BAN logic [3]. The most of these attacks were in finding type flaws [7] and in parallelism of protocol [5]. With better understanding of the field new problems arise that strain against the limit of the existing tools. Tools as they exist today are mainly intended to be applied to the problem of showing that the key has been correctly authenticated. This way was usual application for cryptographic protocols in the past, and remains a common one today. But cryptographic protocols are now being applied to new types of problems, such as group communication, financial transactions, and negotiation of algorithms as well as keys. They also face new types of threats, such as denial of service, and older threads, such as traffic analysis, that are becoming more prominent. In many cases existing tools are not capable of dealing with these problems. But we can believe that many of these existing techniques could be adapted so they could be applied successfully. In this paper we describe some types of formal analysis and problems concerning with parallelism of protocols.

First formal analysis was done using formal logics BAN, GNY [11, 6], but in their definition a number of stringent assumptions is required, including the assumption that participants are trustworthy which excludes the open Internet, environment that is most commonly used today.

Modal checkers [2] are very successful especially when they come to find attacks. But, due to the state space explosion, which is common phenomenon in model checking approaches, it is necessary effectively reduce this space.

2. Protocols

As we mentioned above, protocol is a set of messages. Standard notation of sending message is Alice-Bob notation, $A \to B\colon m$, which means that message m is sent from principal A to principal B. Protocol is a sequence of messages, even some of them can be optional or parallel (the order of these messages is not relevant, but protocol continues after sending all of them). But because of security, cryptography is often used. There are different types of specification of the protocol, some were designed for use only with some tools of formal analysis, but the trend is to have protocol specified once and to be able to reuse this specification, good example for this can be CAPSL [1].

For proving the security of protocols, the main idea is to describe all parts of messages—plain text, encryption (symmetric or asymmetric), nonces, timestamps, key. Every protocol has some assumptions for principals as well as a wanted aim. Assumptions are usually known to the principals before starting the run of protocol, e.g., principal A wants to communicate with principal B, they have secret key K, principal A trusts the principal-server S, or capabilities of executing some functions, e.g., generation of new keys or signing any text. For analysis one needs to know if these used functions are one-way or are invertible.

For communication with server one can describe that principals and server know all keys between them, or that there exists function from principal to secret key, every principal knows his value and the server knows the function.

As for possible aims of protocols, we need to mention standard aims like sending some information (new key), establishing connection between principals as well as aims of security protocols, such as confidentiality, authentication, integrity and non-repudiation. Nowadays, with group communication and/or using grids, there grows up new aims for signing by all principals in a group, etc. But every protocol has his own aims, which its developer wants to achieve. It is a good way to mention all aims in the specification.

2.1 Attacks

What is an attack? Successful attack means that protocol has been finished (last message has been sent), but wanted aims were not achieved. When you request the website from your bank's server system, how do you know if it is the bank's genuine website, as opposed to a fake one? It can occur when one principal has run protocol with intruder instead of the expected principal. Why we mentioned last message only sent, not received? Communicating principals will differ in number of finished runs of protocols. And what will happen? If it occurs in the phase of authentication, probably no attack occurs—in false client authentication all next messages will be rejected, in more authentication runs server will wait for communication without receiving any message (what can finally raise to denial of service attack...). But what if it occurs during communication between bank and internet shop—we can buy something without paying, or pay more times for the same thing—and this will be a serious attack.

Attack is done by a malicious intruder (but it can be also a regular principal). What actually can an intruder do? Everything like any other principal, generating random numbers, concatenating parts he already knows, en/de-crypting message with known keys, extracting every message of protocol from the communication in network, and many other ways manipulating principals within protocol runs: replay attack—freshness flaw (stale message), man-in-the-middle attack—interposing an intruder between communications of honest principals, oracle attack—an intruder uses honest principal to decrypt messages that he can later use for attacking, interleaving attack— an intruder engages in a number of runs interleaved sending message received in one run as an input for another. This is not the exhausting list. Of course, an intruder can combine all these strategies in attack against one protocol run, which makes protocols more difficult to analyse.

2.2 Parallelism in protocols

Under parallelism in protocols we (in formal analysis) do not mean real parallelism in receiving bits, even in reality it is now possible (imagine mother boards with two network cards), but message is sent as a whole, and messages from two (or more) runs of protocols can be interleaved. Order of message from one run of protocol is always the same.

The first what you think about attacks using parallelism is that all timestamps are usually fresh. Now, imagine the Yahalom protocol [3]:

1. $A \to B\colon N_A$
2. $B \to S\colon N_B, \{A, N_A\}_{K_{BS}}$
3. $S \to A\colon N_B, \{B, K_{AB}, N_A\}_{K_{AS}}$
4. $S \to B\colon N_A, \{A, K_{AB}, N_B\}_{K_{BS}}$
5. $A \to B\colon \{N_B\}_{K_{AB}}$

At first sight it looks correct, communication between principals A, B is done using trusty principal S. Using two parallel runs, one started by principal A, the second started by intruder, principal A can believe that she has completed run with principal B, even if principal B never took part. An intruder is modeled as a principal I with subscript significating which principal he is playing for (role confusion attack).

1_I. $A \to I_B\colon N_A$

    1_{II}. $I_B \to A\colon N_A$

    2_{II}. $A \to I_S\colon N_B\{B, N_A\}_{K_{AS}}$

2_I. $I_B \to S\colon N_A, \{B, N_A\}_{K_{AS}}$

3_I. $S \to A\colon N_A, \{A, K_{AB}, N_A\}_{K_{BS}}$

4_I. $S \to I_B\colon N_A, \{B, K_{AB}, N_A\}_{K_{AS}}$

    3_{II}. $I_S \to A\colon N_B, \{B, K_{AB}, N_A\}_{K_{AS}}$

5_I. $A \to B\colon \{N_B\}_{K_{AB}}$

It shows, that parallelism it is not only multiple runs of protocol between same principals, but a principal can act as a an initiator (sender of the first message) as well as a responder (receiver of the first message). But attack using parallelism doesn't have to use information gathered from communication between these principals A, B. In some cases, it could be useful to use information from communication between principals B, C.

3. Formal analysis

Many attempts have been made over the last decade to develop effective tools to model protocols and the security properties they claim to achieve. As pointed out by many researchers, this has proved to be an extremely difficult task despite the simplicity of the protocols being modeled. One of the promising directions in this area is the use of automatic tools. Automatic tools have the practical advantage that they are easy to use, and they have been successfully applied to find flaws in some proposed protocols. Unfortunately, previously available tools severely suffer from the state space explosion which prevents them from analyzing complicated protocols [4]. This is mainly due to the complexity of the intruder behavior, asynchronous composition, and symmetry redundancy in the traditional verification approaches. Most of these tools are also limited to checking the properties of a security protocol under small configurations, e.g., with two initiators and two responders.

3.1 Formalism of protocol

In this section we describe how to pre-process a protocol. This pre-processing, formalism, is used for following automatised formal analysis. This part is still done by hand, although some parts can be automated. This is the main problem, when we have a bad description of protocol we can prove that it is correct even if it is not true, or, on the other hand, we can introduce false attacks, attacks which really do not exist.

In Alice-Bob notation, it is used to define parts of messages, like keys. For formal analysis we have to know exactly whom is this key for. Typical (basic) formalism is changing notation $A \to B\colon K$ to the form $A \to B\colon A \xleftrightarrow{K} B$, which means that the key is used for communication (only) between principals A and B.

Using asymmetric cryptography, only parts of a key are sent in messages. A key is a compound from some private (secret) information and a received part of the key from other principal. So, the creator of protocol needs to explicitly specify that both principals after receiving parts of the key have the same key. In the same manner, we have to specify all functions used in protocol, sometimes we need to describe also cryptographic background of generating key. It has to be said who can (know) compute function values, who knows some values, if this function is reversible (if an intruder knowing function value and some parameters can compute unknown parameters which should be secret).

Also, we have to describe aims of protocol. In some protocols, their aim is only to deliver some information (acknowledge of receiving), security protocol's aim is to deliver key, so both principals believe that key is used for communication between them. First modeled aims were $A \models A \overset{K}{\longleftrightarrow} B$ and vice versa. Stronger aims are of type $B \models A \models A \overset{K}{\longleftrightarrow} B$, so Bob believe that Alice believe that key is correct.

3.2 Analysis

As we have shown in the previous section, principals have different roles in protocol. As for Yahalom, there is an initiator, a responder and a server. A protocol is defined as a function of principals, so formally we are talking about Yahalom(A, B, S). This modeling allows us to describe parallel runs and the role confusion as well (which is impossible in belief logic [9]), the example shown in previous section we can describe as Yahalom(I_B, A, S).

In a model checker we describe every role using states which can be in. The sign $\square$ is used to distinguish these states. Receiving the already known information means checking if it is correct (good example are nonces or timestamps).

$$\begin{aligned}\text{Initiator}(A) \triangleq\ & \square B\colon \text{Agent} & \bullet\ & \text{env}.A.B \rightarrow \\ & \square N_A\colon \text{Nonce}_F & \bullet\ & \text{send}.A.B.N_A \rightarrow \\ & \square S\colon \text{Agent} & \bullet\ & \text{receive}.S.A.\{N_B, \{B, K_{AB}, N_A\}_{K_{AS}}\} \\ &&& \rightarrow \text{send}.A.B.\{N_B\}_{K_{AB}} \\ &&& \rightarrow \text{close}.A.\text{Initiator} \rightarrow \text{Initiator}(A)\end{aligned}$$

Analysis is modeling principals, for example A, B, S and one intruder as the principal I. We try to find if intruder can finish protocol run in some combination of principals as parameters to protocol function. Every principal has ability to concatenate, en/decrypt anything with know key, etc. An intruder has a special privilege, he receives all messages sent during analysis (like capturing packets from network). The analysis is done from beginning to the end of protocol, or from the end to the beginning, but there exist also combined algorithms, for example Database and Logic Abduction [8].

4. State space problem

Formal analysis is complex, checking lot of possible states, which introduce the problem of size state space. Even for small number of principals, the size is growing extremely (because of possible combinations of principals). In this section we describe some reduction techniques which are used to minimize this size of state space with some results for analysis of Yahalom protocol.

4.1 Reduction and false attacks

Aim of reduction techniques is to reduce the size of state space with not changing properties of the model. Reduction is mapping the set of all objects from analysis to a "smaller set", grouping some elements into one representative. Let us imagine that principal A owns key K_1. Let us have message m encrypted with key K_2, which is unknown to the principal A. That means, the principal A is unable to decrypt original message m from $\{m\}_{K_2}$ message. Imagine mapping H (every reduction is a mapping) that $H(K_1) = H(K_2) = K_3$. The encrypted message after mapping is $H(\{m\}_{K_2}) = \{m\}_{K_3}$, and because the principal A knows that $H(K_1) = K_3$, the principal is able to decrypt the message! Every reduction has to keep same properties without losing some attacks or introducing false attacks.

4.2 Intruder and knowledge

First attacks were done by using old, not fresh, data from previous run of the protocol. For modeling this situation we have to run two consecutive instances of the protocol, during the run of the first instance an intruder is collecting information, the attack is done in the second instance. Necessarily we need to check messages twice. The most used reduction technique is to include some information into the intruder's start knowledge (also called stalled knowledge). Usually we include there all parts of all messages sent in protocol, setting all nonces and timestamps as old.

We can reduce also some parallelism here (role confusion), including parts of messages sent when principals' roles are interchanged. This reduction does not lead the analysis to only one run of protocol, it only simplifies the beginning of the analysis. It reduces parallelism only up to some level. It does not cover a situation when an intruder uses in communication between principals A, B information from instance of the same protocol between principals B, C.

4.3 Asymmetric keys

When trusted center (server) is used in protocol, communication with this center is usually done symmetrically by secret keys. In formalism, these keys can be modeled as stand alone values and server knows all values, or we can define one function with parameter principal and server knows this function and every principal knows his own function value. This could help in protocols with more principals communicating with the same center, it makes analysis a bit easier when analyzing communication with center, e.g., role confusion.

4.4 Nonces

Another reduction technique is encapsulating all fresh values. Standard behavior of analysis is to simulate real situations, so every principal has some fresh values, nonces. All principals need to be capable of generating, collecting and managing own fresh values. Encapsulation gives an easier management, generating new fresh values, deleting values within last occurrence in the protocol order. Encapsulation is done by the internal server (special type of principal) of formalism.

4.5 Yahalom protocol

Using these reduction techniques in analysis of Yahalom protocol [2] we can get the following sizes of state space (needed to find an attack):

external server, without stale knowledge	222.104
internal server, without stale knowledge	12.977
external server, with stale knowledge	1.024
internal server, with stale knowledge	249

5. Conclusions and future work

We have shown that parallelism in a protocol can be formally analysed using modal checkers. Using reduction techniques and their combinations, the state space problem is significantly reduced, so we can analyse more complex protocols or complex systems consisting of different protocols.

Still this is only the first part of proving security, there are also another problems, such as type flaw attacks, denial of service attacks, analysis of which has just started.

REFERENCES

[1] BRACKIN, S.—MEADOWS, C.—MILLEN, J.: *CAPSL interface for the NRL protocol analyzer*, in: IEEE Symposium on Application-Specific Systems and Software Engineering Technology, (ASSET '99), IEEE Computer Society, 1999, pp. 64–73.

[2] BROADFOOT, P. J.: *Data independence in the model checking of security protocols*, Doctor of Philosophy Thesis, University of Oxford, UK, 2001.

[3] BURROWS, M.—ABADI, M.—NEEDHAM, R.: *A logic of authentication*, in: Proc. of the twelfth ACM symposium on Operating systems principles, 1989, pp. 1–13.

[4] CANETTI, R.: *Security and composition of multi-party cryptographic protocols*, J. Cryptology **13** (2000), 143–202.

[5] DODIS, Y.—MICALI, S.: *Parallel reducibility for information-theoretically secure computation*, Lecture Notes in Comput. Sci. **1880** (2000), 74–92.

[6] LI GONG, NEEDHAM, R.—YAHALOM, R.: *Reasoning about belief in cryptographic protocols*, in: Proceedings of the 1990 IEEE Comput. Soc. Symposium on Research in Security and Privacy, 1990, pp. 234–148.

[7] GRITZALIS, S.—SPINELLIS, D.: *Cryptographic protocols over open distributed systems: A taxanomy of flaws and related protocol analysis tools*, 16th International Conference on Computer Safety, Reliability and Security, SAFECOMP'97, Springer-Verlag, Berlin, 1997, pp.123–137.

[8] JENČUŠOVÁ, E.—JIRÁSEK, J.: *Formal methods of analysis of security protocols*, Tatra Mt. Math. Publ. **25** (2002), 1–10.

[9] KRIVOŠ-BELLUŠ, R.: *Formal analysis of security of IKE protocol*, Information Technologies—Applications and Theory, 2002, Institute of Informatics, Pavol Jozef Šafárik University, Košice, Slovakia, 2002, pp. 125–133. (In Slovak)

[10] MEADOWS, C. A.: *Analyzing the Needham-Schroeder public key protocol: A comparison of two approaches*, in: Proc. of the 4th Eropean Symp. on Research in Computer Security, Lecture Notes Comput. Sci. Tutor. 1146, 1996, pp. 351–364.

[11] SAUL, E.: *Facilitating the modelling and automated analysis of cryptographic protocols*, Master's thesis, Data Network Architectures Laboratory, Department of Computer Science, University of Cape Town, South Africa, 2001.

Received October 23, 2003

Institute of Computer Science
Faculty of Sciences
Pavol Jozef Šafárik University
Jesenná 5
SK–041 54 Košice
SLOVAKIA
E-mail: rkb@science.upjs.sk