Tatra Mt. Math. Publ. **25** (2002), 1–10

TATRA MOUNTAINS
Mathematical Publications

FORMAL METHODS OF ANALYSIS OF SECURITY PROTOCOLS

Eva Jenčušová — Jozef Jirásek

ABSTRACT. In time of internet attacks it is important to use security and cryptographic protocols to secure private data that are sent via Internet. But using of such protocol is not enough. To really secure our data we must know that the used protocol is really secure. For this purpose a lot of methods were designed. The article presents new DLA method (Database and Logic Abduction) that is based on combination of datalog and logic abduction theory. This method is used to find out whether a security or cryptographic protocol satisfies security properties.

1. Introduction

We live in time of computers and scientific and technological progress, but this time is also marked with many attacks on big Internet companies. So with the expansion of Internet, people start to protect information that they exchange through Internet. It was the beginning of security protocols and cryptographic algorithms in Internet community. A lot of work was done in this field and many different protocols were designed to protect data in packets. But designing a protocol is only one side of the problem; another one is verifying of designed protocol.

This article deals with verifying of security protocols. We present short preview of known methods that are used to verify protocols and we also want to present our DLA — method (Database and Logic Abduction) that is logic based and uses combination of top-down and bottom-up reasoning.

2000 Mathematics Subject Classification: 68N17, 94A60.
Keywords: security, cryptographic protocol, protocol analyzer, protocol attacks, datalog, abduction.

Supported by grant VEGA 1/7557/20.

2. Description of known models

At the beginning of 80s, in the computer world the first cryptographic key distribution protocol was published. In [2] Needham and Schroeder gave only intuitive argument for its correctness. Since their protocol (Needham–Schroeder) was published, a lot of techniques and methods have been designed to analyze security and cryptographic protocols.

2.1 BAN logic model

The BAN logic model for analyzing security protocols was published by Burrows, Abadi and Needham [1] in 1990. In this method it is hard to describe abilities of adversary or who adversary is. This method is based on logic. They present in their paper a specialized logic based on reasoning about belief. Formulae in BAN logic represent statements about the beliefs of agents that participate in a protocol.

The proof consists of statements. The statements look like:

"A believes that message M is fresh."

"A believes that B said M."

"A believes that B believes M" etc.

At the beginning of proof there is the set of initial beliefs and inference rules. The inference rules describe a protocol. The goal is to show that both participants of a protocol believe that a message M is sent from another participant and this message is "fresh". This protocol is idealized, it shows the belief of participants in the success of the protocol. In this method, the main point of formal method for analyzing protocols is missing. We can not inquire whether or not any adversary can learn some secret information from protocol communication.

2.2 Algebraic methods and NRL Protocol Analyzer

There are also another types of methods. Dolev and Yao designed the method that uses algebraic specification of a protocol. In their method they describe an adversary as legitimate agent that can participate in protocol. An adversary can initiate a protocol and receive protocol messages from other agents and can see any message in the network. They assume that the functions for encryption and decryption are unbreakable and one-way. They also assume that knowledge of a encrypted message does not uncover anything about the decrypted message.

Meadows presents in [3] NRL Protocol Analyzer — a tool for the verification of cryptographic protocols. Meadows tool is based on Dolev and Yao's work. The Analyzer is a communicating state machine that searches all possibilities to generate attacks on insecure protocols. In the case that the machine can reach only unreachable states, the Analyzer provides a security proof of a protocol. Also here an adversary can read all traffic, delete or modify any message in the traffic. The goal of this method is to determine whether or not a protocol is secure by specifying an insecure path (from insecure state to an initial state). The Analyzer works backwards from that state until it finds either an initial state (it means an attack) or an unreachable state.

2.3 Logic methods

The logic-based methods are well-suited for automation. Paulson [5] has designed a model that is used with Isabelle — ML-based theorem prover. This model is adversary-centric. It means that an adversary can read all messages and send new messages. An adversary can participate in protocol as a legitimate user. Protocol is defined as the set of possible utterances, then inductively defining the set of possible protocol traces. Protocol specification is the set of rules that extend protocol trace, including rules that describe what an adversary may send. Protocol analysis consists of proving properties (secrecy, regularity, uniqueness, etc.) of all possible traces.

Millen's Interrogator [6] is a Prolog-based search engine that searches for instantiations of a predicate specifying the adversary's knowledge of specified data via some sequence of messages to an insecure state. Protocol is specified as a sequence of transmit and receive actions with additional information about the initial state and message format.

Dexter in [4] describes the method that is based on ideas from logic programming. This method deals with relation *Knows* which represents what the adversary knows. That is, $Knows(X)$ is true iff the adversary knows a message X. Relation *Knows* initially contains agent identifiers and some data, for example keys that the adversary knows as a legitimate agent. This model is also adversary-centric, the adversary can receive, send, modify and waste messages. An analyzed protocol is modeled as a set of inference rules that copy the processing of protocol and adding new facts into adversary knowledge set. Dexter implemented his technique in the theorem prover written in Standard ML of New Jersey. The prover has on input a protocol specification and a goal message (the prover proves whether or not the adversary is able to learn this message). The prover generates on output a list of possible attacks. An attack is described as a sequence of messages that demonstrate how the adversary can learn the goal message.

3. Database model

We deal with a model based on logic, but our point of view is more common with databases. Our method DLA (=database and logic abduction) is different from others that mostly used top-down reasoning. In our model we combine top-down and bottom-up reasoning. In the first step of our method we use datalog evaluation (to show whether a direct attack is possible) and in the second step we apply logic abduction to state additional knowledge that must be used to perform an attack. The goal of our method is not only to prove that no attack is possible to be performed on an analyzed protocol (as others logic-based methods), but we also try to say where are possible weak points of this protocol. These weak points could lead to an attack.

3.1 DLA Environment

By every design, before designing of a new model, the environment of that model must be described. In the world of security and cryptographic protocols there are three important issues that describe environment:

1. Description of the adversary (intruder or attacker).
2. Defining of an attack.
3. Description of the protocol.

The adversary in our model has a complete control over the network. It may read, block, delay or alter any sent message. It also may create new messages. Of course, the adversary has all abilities of a legitimate participant. That means the adversary can also initiate legitimate communication. It knows identifiers of all legitimate participants, it may share a key with the key server. The goal of the adversary is to try to acquire illegitimately some privilege or information of a legitimate participant. Simply, the adversary tries to learn some information not intended for him (violating secrecy), or it tries to masquerade as another legitimate participant (violating authenticity).

In our model we focused on flaws that can arise in the design stage. It means we restrict our attention to fresh and oracle flaws from Carlsen taxonomy [11]. These types of attacks allow to the the adversary to achieve messages from one protocol evaluation and then adversary can perform the crypto-analysis or can simply use intercepted data by next protocol evaluation to try to masquerade or to learn the session key. We use following possibilities to try to find these types of flaws. We try to determine whether the adversary is able to produce all protocol messages generated by one of the legitimate participants (oracle flaws) and we try to show whether the adversary is able to learn the content of the encrypted message at the end of the protocol. We also try to find whether it is possible to do a denial of services attack, where one of the participants believes that he is sharing session key with the other one, but this doesn't share any session key with him. The adversary does not have any a priori knowledge of session-dependent information.

If we want to analyze any protocol we need to find a suitable language to express it in form that can be used for formal analysis. As a lot of others researchers, we use in our model the language of logic programming to express the evaluation of the protocol. In our model we describe the protocol as a set of inference rules that express the sending and receiving of messages and also the order of sending of the messages. We show that description of the protocol and the whole environment can be made also in a different language than the language of logic programming. In the next sections we shall present how it is possible to express the protocol and its environment in the language of database modeling.

3.2 DLA Language

First of all we describe the knowledge set and its attributes. The core of the whole model is the relation *Messages* that contains knowledge of all participants. *Messages* is a binary relation.

Messages (*Agent*: *Dagents*, *Word*: *Dmessages*)— it means that the agent *Agent* knows the message *Word*.

Dagents — it is a set that contains identifiers of all participants.

Dmessages =(*Dnonces* + *Dagents* + *Dkeys* + *Dtimestamps* + *Dwords*) it means that any message can consists of nonces, identifiers of participants, cryptographic keys, timestamps and arbitrary data. Arbitrary data can be data exchanged after the evaluation of the protocol. In the messages there can also be new words that are created by evaluation using functions as concatenation, projection, handshake, cryptographic hash and key encryption.

In the model there are also another relation schemes:

Keys (*Agent*: *Dagents*; *Key*: *Dkeys*)— it says that the participant *Agent* knows the key *Key* .

Nonces (*Agent*: *Dagents*; *Nonce*: *Dnonces*)— it says that the participant *Agent* knows the nonce *Nonce* .

Key (*Key*: *Dkeys*)— it says that the word *Key* is a key.

Nonce (*Nonce*: *Dnonces*)— it says that the word *Nonce* is a nonce.

Dagents = $\{a, b, s, i\}$ — it contains all possible values of agents (for Needham-Schroeder protocol).

Dkeys = $\{ka, kb, ki, kab$— where kx means a private key of the agent x and kxy means a session key for communication between agents x and y.

Dnonces $= \{m, n\}$ — where nonces m is used by the agent b in a session between agents b and a and the nonce n is used by the agent a in a session between agents a and b (for Needham–Schroeder protocol).

How do we perform an analysis with our method? First we describe the knowledge of participants as relations in the database model. Then we translate the protocol into the inference rules. These rules describe, how agents participating in protocol can learn new messages. In our model we have three types of rules — protocol independent cryptographic rules, protocol specification rules and relational rules. That is another difference from other models. Protocol independent cryptographic rules are rules that are for the most of protocols the same. This type of rules describes general cryptographic functions used in the security protocol as encryption, decryption, hash, handshake and operations with words as concatenation and projection. Protocol specification rules are used for description of an analyzed protocol. And the last group of rules is the group of the relational rules that are used to distinguish a type of words in the main relation. We use all types of rules to make the closure of the knowledge relation. In this moment we apply bottom-up reasoning. Rules in our method can look like the following:

$$\mathit{Messages}\,(X, M)\colon -\mathit{Messages}\,\big(X, e(M, K)\big)\,\&\mathit{Keys}\,(X, K)\,.$$

— it means that any agent X can learn the message M if he has the key K and has the encrypted message M with the key K.

We present here as an example the Needham–Schroeder protocol:

1. $a \to s\colon a, b, n$
2. $s \to a\colon \big\{n, b, kab, \{kab, a\}_{kb}\big\}_{ka}$
3. $a \to b\colon \{kab, a\}_{kb}$
4. $b \to a\colon \{m\}_{kab}$
5. $a \to b\colon \{m-1\}_{kab}$

6a. $a \to b\colon \{p\}_{kab}$

6b. $b \to a\colon \{q\}_{kab}$

These exchanged messages we transform into following inference rules:

Protocol specification rules in Needham–Schroeder protocol:

[1] $\mathit{Messages}\,\big(X, [a, b, n]\big)\colon\; - X = a \vee X = b \vee X = i \vee X = s\,.$

[2] $\mathit{Messages}\,\Big(X, e\big(\big[N, Z, kab, e\big([kab, Y], O\big)\big], L\big)\Big)\colon -\mathit{Messages}\,\big(s, [Y, Z, N]\big)\,.$

[3] $\mathit{Messages}\,(X, S)\colon -\mathit{Keys}\,(Z, L)\;\&\;\mathit{Messages}\;\big(Z, e\big([n, b, K, S], L\big)\big)\,.$

[4] $\mathit{Messages}\,\big(X, e(m, K)\big)\colon -\mathit{Keys}\,(Z, L)\;\&\;\mathit{Messages}\;\big(Z, e\big([K, U], L\big)\big)\,.$

[5] $\mathit{Messages}\,\big(X, e\big(f(M), K\big)\big)\colon -\mathit{Keys}\,(Z, K)\;\&\;\mathit{Keys}\,(Z, L)\&$

$$\&\mathit{Messages}\,\big(Z, e\big([n, b, K, S], L\big)\big)\;\&\;\mathit{Messages}\,\big(Z, e(M, K)\big)\,.$$

[6] $\mathit{Messages}\,(X, e(P, K)) : -\mathit{Keys}\,(Z, K)\ \&\ \mathit{Keys}\,(Z, L)\ \&$
$\&\ \mathit{Messages}\,(Z, e([n, b, K, S], L))\ \&\ \mathit{Messages}\,(Z, e(M, K))\,.$

[7] $\mathit{Messages}\,(X, e(Q, K)) : -\mathit{Keys}\,(Z, K)\ \&\ \mathit{Messages}\,(Z, e(f(m), K))\,\&$
$\&\ \mathit{Keys}\,(Z, L)\ \&\ \&\ \mathit{Messages}\,(Z, e([K, U], L))\,.$

Constants: a, b, i, s — from the domain *Dagents*
m, n — the session nonces between agents a, b from *Dnonces*
kab — the session key between agents a, b

Variables: X, Y, Z, U — from the domain *Dagents*
K, L, O — from the domain *Dkeys*
$P, Q, S, T, T1, T2$ — an arbitrary word (from the domain *Dmessages*)
M, N — from the domain *Dnonces*.

Protocol independent cryptographic rules used in Needham–Schroeder protocol:

[V.1] $\mathit{Messages}\,(X, T) : -\ \mathit{Messages}\,(X, e(T, L))\ \&\ \mathit{Keys}$ (X, L) — decryption.

[V.2] $\mathit{Messages}\,(X, e(T, L)) : -\ \mathit{Messages}\,(X, T)\ \&\ \mathit{Keys}\,(X, L)$ — encryption.

[V.3] $\mathit{Messages}\,(X, f(T)) : -\ \mathit{Messages}\,(X, T)$ — handshake.

[V.4] $\mathit{Messages}\,(X, T) : -\ \mathit{Messages}\,(X, f(T))\,.$

[V.5] $\mathit{Messages}\,(X, T1) : -\ \mathit{Messages}\,(X, [T1, T2])$ — projection.
$\mathit{Messages}\,(X, T2) : -\ \mathit{Messages}\,(X, [T1, T2])\,.$

[V.6] $\mathit{Messages}\,(X, [T1, T2]) : -\ \mathit{Messages}\,(X, T1)\ \&\ \mathit{Messages}\,(X, T2)$
— concatenation.

The relations *Nonce*, *Key* and *Agent* are unary relations that are not changed during the evaluation of protocol, because they only define the types of words in protocol messages. The relations *Nonces*, *Keys* and *Agents* are binary relations that describe some type of knowledge of participants and they can change during evaluation of protocol by using the following rules: Relational inference rules:

[R.1] $\mathit{Nonces}\,(X, N) : -\ \mathit{Messages}\,(X, N)\ \&\ \mathit{Nonce}\,(N)\,.$

[R.2] $\mathit{Keys}\,(X, K) : -\ \mathit{Messages}\,(X, K)\ \&\ \mathit{Key}\,(K)\,.$

[R.3] $\mathit{Agents}\,(X, A) : -\ \mathit{Messages}\,(X, A)\ \&\ \mathit{Agent}\,(A)\,.$

After this evaluations we have in relations all knowledge that can agents learn from traffic. At this point we can ask on this knowledge. The most interesting one is the adversary's knowledge. There are five important questions:

a) Does the adversary know the session key k between agents a and b?

b) Does the adversary know the secret information p sent by the agent a to agent b?

c) Does the adversary know the secret information q sent by the agent b to agent a?

d) Does the agent a know the secret information q sent by the agent b to him?

e) Does the agent b know the secret information p sent by the agent a to him?

These questions can be written in a different formalisms:

Question a) it means whether or not the agent i—adversary has (i.e., knows) key k that the server produced for secure communication between agents a and b.

1. From the database point of view:

   SELECT count(agent)

   FROM Messages

   WHERE agent="i" AND message="k"

2. From the relational algebraic point of view:

   $(i, k) \in Messages.$

There are two possible answers on every question in our model,— "Yes" if the ordered pair is in the relation, or "No". In formalism 1 the answer "Yes" means that the select returns at least one row (it depends on structure of table *Messages*) and in formalism 2 it means that the ordered pair (i, k) is a member of the relation *Messages*. If the answer is "No", it is useful to know in what cases it should be true. In this moment we use the logic abduction to determine conditions that are needed for the adversary to be able to learn the word in question. Abduction is an important form of reasoning allowing one to find explanations for certain observations. In our case, observations are questions on security properties of an analyzed protocol. On the well-known Needham–Schroeder protocol we shall show how in our case the abduction exactly works.

The background knowledge is an iterated relation *Messages* , and rules for abduction are the same rules used by iteration of the relation *Messages* . The abduction for question (i, k) is a part of the abduction for question (i, p), so in the following part, the evaluation of abduction for question (i, p) is shown.

The abduction for question $(i, p) \in Messages$

[V.1] $Messages\,(i, p){:} - Messages\,\big(i, e(p, L)\big)\ \&\ Keys\ (i, L).$

L from domain $Dkeys \to L$ must be from $\{kab, ka, kb, ki\}$:

$L = kab$

1. $Messages\,(i,p)\colon -Messages\,\big(i,e(p,kab)\big)\ \&\ Keys\,(i,kab)$.

 1.1?– $Messages\,\big(i,e(p,kab)\big)$.

 $\big(i,e(p,kab)\big)$ is in *Messages*, so the answer is *true*

 1.2?– $Keys\,(i,kab)$.

  [R.2] 1.2.1 $Keys\,(i,kab)\colon -\ Messages\,(i,kab)$.

  [V.5] 1.2.2 $Messages\,(i,kab)\colon -\ Messages\,\big(i,[kab,a]\big)$.

  [V.1] 1.2.3 $Messages\,\big(i,[kab,a]\big)\colon -$

   $Messages\,\big(i,e\big([kab,a],kb\big)\big)\ \&\ Keys\,(i,kb)$.

   1.2.3.1 $Messages\,\big(i,e\big([kab,a],kb\big)\big)$.

   $-\big(i,e\big([kab,a],kb\big)\big)$ is in *Messages*.

   1.2.3.2 $Keys\,(i,kb)$.

   —and this fact is possible to reach (for the adversary)

   only by crypt-analysis.

Result: $Messages\,(i,p) \Leftarrow Keys\,(i,kb)$. This result shows that the only possible way to attack the protocol is to obtain the key.

We understood the problem of analysis of security protocols as two-dimensional problem. One dimension is the method used by the analysis of protocols and the second dimension is the complexity of analyzed protocols. The analysis of protocols is focused especially on the key exchange protocols, because if we can make an attack on such a protocol, we can disable or learn all the following traffic.

We can distinguish two types of key exchange protocols: 1. with the key server that distributes the new keys 2. without a key server and without a key distribution (the participants exchange only a key material used for a key derivation)

The complexity of analysis of these groups is much different. The analysis of the first group of protocols is simple (in this group there are simple protocols as Needham–Schroeder). The complexity of the second group is not so simple. In such protocols we must include into analysis also the key derivation. In to this group there belong protocols like the Internet Key Exchange (IKE) protocol.

We tried our DLA method on both types of protocols. With our method we analyzed in [10] the Otway–Rees protocol, the Needham–Schroeder protocol and the Internet Key Exchange protocol. In all these cases we find all known weak points in these protocols. We also showed in [10] how we can use the results of an attacked protocol to modify such a protocol to be secure.

REFERENCES

[1] BURROWS, M.—ABADI, M.—NEEDHAM, R. M.: *A logic of authentication*, ACM TOCS Transactions in Computer Systems **8** (1990), 18–36.

[2] NEEDHAM, R. M.—SCHROEDER, M. D.: *Using encryption for authentication in large networks of computers*, Commun. ACM **21** (1978), 993–999.

[3] MEADOWS, C.: *The* NRL *Protocol Analyzer: an overview*, J. Log. Program (1996), 113–131.

[4] DEXTER, S.: *An Adversary-Centric Logic of Security an Authenticity*, PhD thesis, University of Michigan, 1998.

[5] PAULSON, C. L.: ML *for Working Programmer*, Cambridge University Press, New York, 1996.

[6] KEMMERER, R.—MEADOWS, C.—MILLEN, J.: *Three systems for cryptographic protocol analysis*, J. Cryptology (1994), 79–130.

[7] VOJTÁŠ, P.: *Tunable fuzzy logic programming for abduction under uncertainty*, in: Proceeding of the Workshop "Many valued logic for Computer Science Application" at ECAI'98, University of Brighton, 1998, p. 7.

[8] COMPTON, K. J.—DEXTER, S.: *Proofs Techniques for Cryptographic Protocols*, Workshop at ECAI'98.

[9] FERGUSON, N.—SCHNEIER, B.: *A Cryptographic evaluation of IP sec,* `http://www.counterpane.com` (1999).

[10] JENČUŠOVÁ, E.: *Security proofs of communication protocols*, PhD thesis, University of P. J. Šafárik, Košice, 2001.

[11] CARLSEN, U.: *Cryptographic protocol flaws: Know your enemy*, Computer Security Foundations Workshop VII, 1994.

Received February 14, 2002

Department of Computer Science
Faculty of Natural Sciences
University of P. J. Šafárik
Jesenná 5
SK–041 54 Košice
SLOVAKIA

E-mail: jencusova@yahoo.com
jirasek@kosice.upjs.sk