Automated Design of Non-repudiation Security Protocols

Haifeng Xue[1], Huanguo Zhang[1, 2], Sihan Qing[3], Rongwei Yu[1]
1. School of Computer, Wuhan University, Wuhan 430072, Hubei, China
2. The State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, Hubei, China
3. Institute of Software, Chinese Academy of Sciences, Beijing 100080, China
xhf163com@163.com, liss@whu.edu.cn, qsihan@ercist.iscas.ac.cn

Abstract

This paper proposes an approach of automated design of non-repudiation security protocols from the abstract level that uses logic-based rules. The search strategy adopts the genetic algorithm which is a meta-heuristic search method. During the searching process, some counter measures against flaws of security protocols are added to assess the candidate protocols. Therefore candidate protocols can guarantee some security properties. In the past, security protocols are mostly designed manually which depends on the experiences and skills of experts. The automated design method of security protocol alleviates the burden of designing security protocols, and guides the designer to get a fast and better security protocols.

Keywords: Security protocol, BAN Logic, Genetic Algorithm

1. Introduction[1]

Security Protocols are widely used in network communication systems for authentication, key establishing, non-repudiation, fairness, etc. In the past 20 years, people paid much attentions on analysis and verification of security protocols by using many formal methods and automated tools[1]. Flows and leaks of security protocols have been found in endlessly cases. Security protocols are notoriously difficult to get right. The fundamental cause is that the security of protocol is un-decidable[2] in distributed systems. Security protocols are mostly designed manually which depends on the experiences and skills of designers. Little work has been carried out on automated design and synthesis of security protocols.

Gong and Syverson proposed the Fail-Stop protocols [3]. Perrig and Song's Automated Protocol Generator(APG) [4,5] adopted state checking strategy and strand space's theory to design the authentication protocols. Zhou[6] proposed an automated security protocol generator which uses logic-based rules and backward search. Clark and Jacob[7,8,9] automatically designed security protocols by using meta-heuristic search strategy to get the simplest authentication property. They almost did not consider any attacks to known flaws of security protocols. Even though the meta-heuristic search is beginning for designing security protocols, their works have provided us a new way to design security protocols. This paper bases on meta-heuristic search strategy by BAN logic[10].

This paper provides an automated design of security protocols which has some counter measures against some known flaws. Another function is that can design non-repudiation security protocols. The second part of the paper, belief logic and its inference rules are introduced; The third part proposes some initial rules for instructing security protocols; The non-repudiation is discussed in the forth part; the fifth provides the full genetic algorithm on designing security protocols; Experiments is showed in the sixth part; In the last part, we draw a conclusion about the paper.

2 Belief Logic and Its Inference Rules

2.1 Notion and Belief Logic

In 1989, Burrows, Abadi and Needham developed a logic (BAN Logic) [10] that could be used to reason about security protocol. Its best advantage that BAN logic has is its simplicity and high level abstract. The principals of protocol hold some information that is called the principal's beliefs.

The following is the basic notation of BAN logic.

Believes: The assertion $P| \equiv X$ means P *believes* the formula X. P may act as if X is true.

Sees: The assertion $P \lhd X$ means P *sees* X.

Once said: $P| \sim X$ means P once said X, i.e. P at some time sent a message including the statement X.

Jurisdiction: $P| \Rightarrow X$ means P *has jurisdiction over* X.

Fresh: #(X) means the formula X is fresh.

Key Goodness: $P \xleftrightarrow{K} Q$ means K is a good key for communication between P and Q.

Public Key: $\xrightarrow{K} P$ means K which is the public key K of a principal is owned by P.

Encryption: $\{X\}_k$ means the formula X is encrypted under the key K.

[1]Supported by the National High Technology Development 863 Program of China under Grant No. 2006AA01Z442;The National Natural Science Foundation of China under Grant No. 60673071, 60573042; The National Grand Fundamental Research 973 Program of China under Grant No. G1999035802

2.2 Inference Rules

When a principal receives a message, the receiver can update their beliefs according to the inference rules and beliefs owned. The following is the major inference rules.

Message Meaning Rules. The message meaning rules interpret how to derive new beliefs from the origin of messages.

$$\frac{P \mid\equiv P \xleftrightarrow{K} Q, P \lhd \{X\}_K}{P \mid\equiv Q \mid\sim X} \quad (1)$$

There is similarly rule for public keys.

$$\frac{P \mid\equiv \xrightarrow{K} Q, P \lhd \{X\}_{K^{-1}}}{P \mid\equiv Q \mid\sim X} \quad (2)$$

For using shared secrets

$$\frac{P \mid\equiv P \xleftrightarrow{Y} Q, P \lhd \langle X \rangle_Y}{P \mid\equiv Q \sim X} \quad (3)$$

Nonce Verification Rule. The nonce verification rule explains how a principal's view of a message changes when it determines that the message is part of the current protocol run.

$$\frac{P \mid\equiv \#(X), P \mid\equiv Q \sim X}{P \mid\equiv Q \mid\equiv X} \quad (4)$$

Jurisdiction Rule. The jurisdiction rule captures the notion that some principals are trusted to carry out certain tasks and make particular judgments.

$$\frac{P \mid\equiv Q \mid\equiv X, P \mid\equiv Q \mid\Rightarrow X}{P \mid\equiv X} \quad (5)$$

3 Initial Designing Rules

For the designing process simplicity, there are some rules exist intuitionally. Some of them are counter measures against known flaws of security protocols.

Rule 1. The sender and the receiver can not be the same in consecutive messages of protocols.

Rule 2. Every principal acts at least one time as sender and receiver.

Rule 3. In any concatenated message, there are no redundant message components, i.e., $\{N_a, N_a\}$.

Rule 4. We do not consider permutations of the message components of a concatenated message.

Rule 5. No initial keys are sent in a message. That is to say, it does not make sense to send a private key. We assume that every principal knows all principals' public keys.

Rule 6. Only encrypted messages can be sent. The unencrypted message contributes any belief to the receiver, which can only signal the receiver.

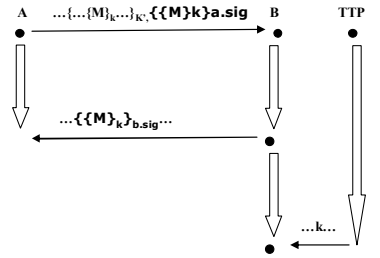
Rule 7. The messages' format can't be the same. This rule can prevent somehow type flaw and multi-protocols attacks.

The above rules are the start of using the automated approach. They are tenable intuitionally which can simplify our design process and decrease the state space considered.

4 Non-repudiation Protocols

Non-repudiation is an important property of e-commercial protocols, that highlights the receiver and sender can't repudiate themselves' action. The non-repudiation includes that the receiver can't repudiate his receiving messages. The Ref [11] lists some guidelines for non-repudiation protocols. A strong non-repudiation protocol is complicated greatly without employing trusted third party. Now we only consider the weak non-repudiation with trusted third party participating. The non-repudiation protocols common is two-step protocols, the first step, a principal sends/receives the encrypted evidence of message which is a signature generally, and the second step, another principal receives/sends the key to the encrypted evidence from TTP. Therefore, the design procedure of non-repudiation is that collects the evident and key in different messages.

We lists the model of non-repudiation protocols as the graph 1



Graph 1: The non-repudiation protocols model

5. Genetic Algorithm on Designing Security Protocol

There are some main problems accompany with the genetic algorithm to design security protocols, such as encoding, characterization of design space and fitness function, etc.

5.1. Protocol Representation

A protocol defines a sequence of sending and receiving action of the participating parties. Messages are defined by the following grammar.

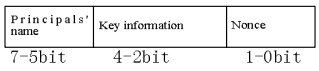
Message ::= Atomic | Encrypted | Concatenated
Atomic ::= PrincipalName | Nonce | Key
Encrypted ::= (Message, key)
Key ::= PublicKey | PrivateKey | SymmetricKey
Concatenated ::= MessagesList
MessageList ::=Message | Message, MessageList

The un-encrypted messages may be used as signals to notify encrypted messages to be sent, but they do not contribute to any principals' beliefs. In this paper, we consider at most two level nested encrypted messages. The messages have the form like the following: $\{A, Na, \{B, Nb\}_{k_b}\}_{k_a}$ or $\{A, Na\}_{k_a}$

5.2. The encoding of messages and beliefs

Because message's length is variable, now we only consider two-level-nest encrypted message with principal name and nonce.

For one-level-nest encrypted message, an unsigned char is used to encode the typical message $\{A, Na\}_{k_a}$.



For a three parts protocol: *A, B, S*, there are P_3^2 =6 combinations between them. There will be $2n+\frac{n(n-1)}{2}$ keys for a three parts protocol. The part of nonce denotes its owner who can identify the nonce whether it is correct or not.

For a two-level-nest encrypted messaged, such as $\{A, Na, \{B, Nb\}_{k_b}\}_{k_a}$, it is extended similarly as the above.

The goal belief is composed of the three operators from the set $\{|\equiv, \Rightarrow, \sim\}$. "$|\equiv$" must be the first operator. The belief's encoding is more complex than the message. A typical belief, such as $A|\equiv B\sim\{B,Na\}_{kab}$, and $A|\equiv B|\equiv S$, can be iterated deeply, which be held by principal *A*. Therefore every principal maintain a list component of beliefs. For $A|\equiv B\sim\{B,Na\}_{kab,}$ principal A only needs to stores the component "$B\sim\{B,Na\}_{kab}$", which can ignore the prefix "$A|\equiv$"

5.3. Initial and sendable beliefs

Initial beliefs may involve 1, 2, or 3 operators from the set $\{|\equiv, \Rightarrow, \sim\}$. Now a definition of a set of atomic assertions is given.

Definition: *SIMPLE* is a set of atomic assertions including nonces (e.g. Na), the goodness of keys (e.g. $A\xleftrightarrow{Kab}B$), or the freshness of such assertions (e.g. #(Na), #($A\xleftrightarrow{Kab}B$)).

The initial beliefs may take the following forms:

1) $P|\equiv X$, where P is a principal and $X\in$ *SIMPLE*. For example, $A|\equiv Na$, $S|\equiv A\xleftrightarrow{Kab}B$.
2) $P|\equiv Q|\Rightarrow X$, where P and Q are principals and $X\in$ *SIMPLE*.
3) $P|\equiv Q|\Rightarrow R$ op X where op$\in\{|\equiv, \Rightarrow, \sim\}$, P, Q and R are principals and $X \in$ *SIMPLE*. For example $A|\equiv S|\Rightarrow B|\sim Nb$.

Sendable beliefs are assertions in *SIMPLE*. They can get from the above rules. Thus, Na, $S|\equiv A\xleftrightarrow{Kab}B$ are both sendable beliefs but $A|\sim B|\sim Nb$ is not.

5.4. Genetic algorithm

The Ref[12] introduces the genetic algorithm. Now, a gene string can be represented as the following format:

vs	*vr*	*vk*	*vb1*	*vb2*	*vb3*	*vb4*

Every protocol message is represented by the gene string. *vs, vr, vk* respectively denote the sender, receiver and the key used to encrypted this message, and a series of *vb* indices that reference beliefs currently held by the sending principal. The number of principals and beliefs are specified as *N* and *T*. the table's value needs modular reduction. For example *vs*=*vs* mod *N*, *vr*=*vr* mod *N*, *vb2*=*vb*2 mod *T*, *vk*=*vk* mod (2*N*+*N*(*N*-1)/2).

Assume a protocol consists of M messages, each of which consists of B beliefs. Firstly we should initialize the belief state of each principal involved in this protocol with its initial assumptions. Then for each message in this protocol, we follow the steps below.

1) Determine the sender, receiver and the key under which the current message is encrypted. If this key is an appropriate one for communication between the sender and the receiver, then proceeds with the rest of the current message, else ignores this message and proceeds to the next messages.

2) Decode each of the B beliefs corresponding to the current message. For instance, the first belief in the message is vb1 mod T, where the sender currently holds *T* sendable beliefs.

3) Update the receiver's beliefs vector by applying these inference rules (1)-(5).

4) Record the number of required goals achieved after this message has been analyzed. At the same time, collect the evidence of non-repudiation.

5.5. The Fitness Function and Parameters

The fitness function is very important to a genetic algorithm which determines the search efficiency and effectiveness. The fitness tells how good a candidate one is. We use the fitness function for a protocol of form

$$\sum_{i=1}^{M} w_i * g_i$$

The w_i are weightings and g_i is the number of required goals achieved after *message i*.

The mainly parameters in genetic algorithm is the crossover probabilities, mutation probabilities. The crossover probabilities take 0.4 as the value. The mutation probabilities is 0.01.

6. Experiments

We give an initial assumption and the goals of a three-part non-repudiation protocol. The selected final results after searching by the genetic algorithm is shown as the below.

Initial Assumptions:

A$|\equiv \xrightarrow{Ka} A$ (Ka is the public key of principal A)
A$|\equiv \xrightarrow{Ka'} A$ (Ka' is the private key of principal A)
B$|\equiv \xrightarrow{Kb} B$ (Kb is the public key of principal B)
B$|\equiv \xrightarrow{Kb'} B$ (Kb' is the private key of principal B)
S$|\equiv \xrightarrow{Ks} S$ (Ks is the public key of principal S)
S$|\equiv \xrightarrow{Ks'} S$ (Ks' is the private key of principal S)
A$|\equiv$Na A$|\equiv$#(Na) B$|\equiv$Nb B$|\equiv$#(Nb) S$|\equiv$Ns S$|\equiv$#(Ns)
S$|\equiv \xrightarrow{Ka} A$ B$|\equiv \xrightarrow{Ka} A$

$A|\equiv \xrightarrow{Kb} B$ $S|\equiv \xrightarrow{Kb} B$

$A|\equiv \xrightarrow{Ks} S$ $B|\equiv \xrightarrow{Ks} S$

$A|\equiv S|\Rightarrow A\xleftarrow{Kab}B$ $B|\equiv S|\Rightarrow A\xleftarrow{Kab}B$

$A|\equiv \{Na, A\}$

The Goal:

$A|\equiv B|\equiv A\xleftarrow{Kab}B$ $B|\equiv A|\equiv A\xleftarrow{Kab}B$

$A|\equiv B\lhd \{Na, A\}$ $B|\equiv A\sim\{Na, A\}$

The candidate protocol:

1) $A\rightarrow B: \{Na, A\}_{Kab,} \{\{Na, A\}_{ka}\}_{Ka',}$
2) $B\rightarrow A: \{\{Na, A\}_{Kab}\}_{kb'}$
3) $A\rightarrow S: \{Kab, \{Kab\}_{ka'}\}_{ks}$
4) $S\rightarrow B: \{Kab\}_{kb,} \{Kab\}_{ks'}$
5) $S\rightarrow A: \{Kab\}_{ks'}$

7. Conclusion and Future Work

Automated design of security protocols by genetic algorithm is an effective method. Encoding of messages and beliefs is an important factor of achieving successful case. However, there are some shortcomings about the method. Firstly, the number of messages and beliefs of the wanted protocol is limited in advance. Secondly, protocol leaks are considered little during design procedure. The generated protocols can only hit the goals that are defined beforehand. The last, the initial assumptions are given manually, which probably be some strong assumptions far away the practice.

In the future, the more strategies against some known protocol's leaks should be added in the design procedure. Some specification on the initial assumptions should be more automated. This paper gives some tries about automated design of security protocol by genetic algorithm. How to Design more complicated and complex security protocol is still under investigation.

References:

[1] Rui Xue, Deng-guo Feng, The Approaches and Technolog ies for Forma l Ver if ica tion of Secur ity Protocols, Chinese Journal of Computers, Vol. 29, No.1, Jan 2006, pp.1-20

[2] I. Cervesato, N. A. Durgin et. Al. A Meta-notation for Protocol Analysis, In: Proceedings of the 12th IEEE Computer Security Foundation Workshop, Mordano, Italy, 1999, pp.55-72

[3] L Gong, P Syverson. Fail-Stop Protocols: An Approach to Designing Secure Protocols, In: Proceedings of DCCA-5 Fifth International Working Conference On Dependable Computer For Critical Applications, Oakland, IEEE Computer Society Press, 1998, pp.79-100

[4] D. Song, A. Perrig, A First Step on Automated Protocol Generation, in: Proceedings of Network and Distributed System Security Symposium. 2000-02

[5] D. Song, A. Perrig, Looking for Diamonds in The Dessert—Automated Security Protocol Generation for Three-Party Authentication and Key Agreement, in: Proceedings of the 13th Computer Security Foundations Workshop, IEEE Computer Society, 2000

[6] Hongbin Zhou, Simon N. Foley, Fast Automatic Synthesis of Security Protocols Using Backward Search, FMSE'03, Oct, 2003, Washington DC, USA

[7] J. Clark, J. Jacob, Protocols Are Programs too: the Meta-heuristic Search for Security Protocols, Information and Software Technology, 2001, pp. 891-904

[8] Hao Chen, J. Clark and J. Jacob, Automated Design of Security Protocols, Computer Intelligence, Vol.20, No3, Blackwell Publishing, USA, 2004

[9] J. Clark, J. Jacob, Searching for a Solution: Engineering Tradeoffs and the Evolution of Provably Secure Protocols. In: Proceedings 2000 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society, 2000, pp.82-95

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

[11] Panagiotis Louridsa. Some Guidelines for Non-repudiation Protocols. ACM SIGCOMM Computer Communication Review, 30(5), October 2000, pp.29-38

[12] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison- Wesley, Reading, MA, 1989