A Problem Solving Mechanism for Formal Analysis of Cryptographic Protocols

JIHONG HAN, YU ZHAO,YADI WANG, ZHIYONG ZHOU

Abstract: - The paper proposes the cryptographic protocol insecurity problem, gives an accurate formal specifications for cryptographic protocols, defines the security properties precisely and reasonably, presents some pivotal concepts and propositions in the deduction of the problem solving. The method has provable semantics which is reasonable and sound, and it is easy to realize automatic deductions.

Key-Words: - Cryptographic protocols, Formal method, Problem solving, Operational semantics, Secrecy, Authentication

1 Introduction

A cryptographic protocol is a precisely defined sequence of communication and computation steps that use cryptographic mechanism, its purpose is to ensure security properties in a hostile environment. Unfortunately many cryptographic protocols are found having security flaws, even in many years after they were first proposed. So, there has been an explosion of interest in formal analysis methods and automatic verification tools of these protocols. Researchers have used techniques from model checking, term rewriting, theorem proving and logic programming amongst others[1,2,3,4,5,6]. Much attention is paid to finding the existence of an attack, but few are concerned about how the attack is accomplished. In this paper, we present a new formal approach for analyzing cryptographic protocols. Our purpose is automatically detecting and presenting attacks when they exist, especially for the parallel session attacks which often be ruled out in most finite models.

An attack on the protocol is a particular instantiation of the protocol in which a malicious agent or an intruder can obtain information meant to be kept secret between other principals in the protocol. The problem of finding the attack sequence can be modeled as a protocol insecurity problem, and we can make the problem solved by AI problem solving theory because problem solving is just the process of looking for a certain sequence of actions which will achieve some stated goal.

2 The Cryptographic Protocol Insecurity Problem

The cryptographic protocol insecurity problem can be defined as a triplet Π=<*InitStates*, *ActionsSet*, *Goals*> whose elements can be formulated as follow:

InitStates= $\{s_1 \cup s_2 \mid s_1 \in InitStates_P, s_2 \in InitStates_I\}$ is a set of initial states that express the knowledge each principal and the intruder have at the beginning of the protocol. $InitStates_P$ is the initial states set of protocol principals and $InitStates_I$ is intruder's.

$ActionsSet = ActionsSet_P \cup ActionsSet_I$ is a set of actions that express the behavior of each principal and intruder in the protocol running. $ActionsSet_P$ and $ActionsSet_I$ are action sets for protocol and intruder respectively.

Goals is a set of goals representing the insecure states in the cryptographic protocol.

We regard the process of cryptographic protocol analysis as a problem solving process, If the problem has a solution, namely a possible sequence of actions leading to a state in *Goals* can be found, then we say the cryptographic protocol is not secure, correspondingly the sequence of actions is just an attack scenario.

In the cryptographic protocol insecurity problem, we don't consider the path cost function, because all paths can be used to make the protocol fail.

2.1 Syntax

In our method, we use terms, knowledge, facts, states and rewriting rules to formulate cryptographic protocols, the syntax is given below:

Term::= AtomTerm| CompoundTerm

AtomTerm::= Var| Num| PName | Key| Nonce

Var::= a| b | k| i | r| m|n| x| y | z| authid|...

Num::= 0 | 1|2|...

PName::= A| B|C| S| T| I| Var | ...

Key::= $symK_{AB}|pubK_A|privK_A|...$

Nonce::= Num| Var

CompoundTerm ::= $\{Term\}_{Key}$

|[Term(,Term)]*

|Hash(Term)

Knowledge::= {Term(,Term)}*

Fact::= Await(PName,PName,Knowledge,Stepid,Sessionid)

|*Send(PName,PName,Term,Stepid,Sessionid)*
|*Receive(PName,PName,Term,Stepid,Sessionid)*
|*Shared(Term,PName, PName,...)*
|*Inverse(Key,Key)*| *Intrude(Term)*
|*New(PName,Nonce)*
|*Begin(PName,PName,authid(,Term) *)*
|*End(PName,PName,authid(,Term) *)*
|*Not Fact*
State::= {} , {Fact(,Fact)}*
Rewriting Rule ::= PreCond $\rightarrow$ *PostCond*
PreCond::= {} , {Fact(,Fact)}*
PostCond::= {} , {Fact(,Fact)}*

Here terms representing message of the protocol can be divided into atom terms and compound terms. Atom terms can be variables *Var,* numbers *Num,* principal name *PName*, keys and nonces. The key includes symmetrical shared keys and asymmetrical keys, e.g. $symK_{AB}$ is the secret key shared between principal A and B, $pubK_A$ is the public key of principal A, and $privK_A$ is the private key of principal A. Compound terms can be cipher $\{Term\}_{Key}$ generated by encrypting *Term* with *Key*, concatenation of terms [*Term(,Term)**] and hash value *Hash(Term).* Fact *Await* represents that a principal waits message from the other, for example, *Await(A,B,Knowledge,Stepid,Sessionid)* means that principal B who possesses knowledge *Knowledge* waits message from A in the protocol session indicated by *Sessionid* and step indicated by *Stepid.* Fact *Send* represents that a principal sends message to the other, e.g. *Send(A,B,Term,Stepid,Sessionid)* means that principal *A* sends message *Term* to *B* in the protocol session *Sessionid* and step *Stepid.* Fact *Receive* represents that a principal receives message from some one else, e.g. *Receive(A,B,Term,Stepid,Sessionid)* means that principal *A* receives message from B. *Shared (Term,PName, PName,...)* represents that *Term* is a secret shared by some principals. *Inverse(Key,Key)* represents two keys constitute a pair of public and private keys. *Intrude(Term)* represents that *Term* is known by intruder. *New(PName,Nonce)* represents that a principle generates a nonce. Fact *Begin* and *End* represents the start and response of the authentication event. States are multisets composed by facts, and rewriting rules can be used to describe the actions in cryptographic protocol running.

2.2 Representation of protocol itself

The protocol is represented by initial states set $InitStates_P$ and action set $ActionsSet_P$. We can illustrate the formalizing method on the following Needham-Schroeder public key protocol:

Step1. $A \rightarrow B : \{A,N_A\}pubK_B$
Step2. $B \rightarrow A : \{N_A,N_B\}pubK_A$
Step3. $A \rightarrow B : \{N_B\}pubK_B$

Each step in the protocol can be regarded as a series of actions on certain conditions. For the Step 1, we have the following rules:

$Rule_1=\{Await(i,i,\{i,r,pubK_i,privK_i,pubK_r\},0,sid)\} \rightarrow \{New(i,n_i),$
$Await(r,i,\{i,r,n_i,pubK_i,privK_i,pubK_r\},2,sid),$
$Send(i,r,\{[i,n_i]\}_{pubKr},1,sid),$
$Begin(i,r,auth\_n_i,n_i)\}$

It represents that the protocol initiator i sends message $\{[i,n_i]\}_{pubKr}$ to responder r, where n_i is a nonce generated by i, $Begin(i,r,auth\_n_i,n_i)$ indicates that principal i initiates an authentication event with principal r.

$Rule_2=\{Await(i,r,\{r,i,pubK_r,privK_r,pubK_i\},1,sid),$
$Send(i,r,\{[i,n_i]\}_{pubKr},1,sid)\} \rightarrow$
$\{Await(i,r,\{r,i,pubK_r,privK_r,pubK_i\},1,sid),$
$Receive(r,i,\{[i,n_i]\}_{pubKr},1,sid)\}$

It represents that the protocol responder r receives message $\{[i,n_i]\}_{pubKr}$ from initiator i.

In the same way, we can obtain rules for the Step 2, and Step 3. Put all these rules together, we have the action set $ActionsSet_P$.

The initial state of protocol $InitStates_P$ contains facts such as each principal knows his own name, public key and private key, he also know names and public keys of other agents who he wish to communicate with.

2.3 Representation of the intruder

The intruder can be formalized by initial states set $InitStates_I$ and action set $ActionsSet_I$. We still take the Needham-Schroeder public key as an example. The state of the intruder can be described by knowledge he possess at this moment, for public key protocols, the intruder's knowledge at the beginning of protocol running include the identity, public key, private key of its own, and the identities and public keys of protocol principals, we can obtain

$InitStates_I=$
$\{ Intrude(I),Intrude(pubK_I),Intrude(privK_I),$
$Intrude(A),Intrude(B),Intrude(C),$
$Intrude(pubK_A),Intrude(pubK_B),Intrude(pubK_C)\}$

The abilities of the intruder can be characterized by rules. for instance, Rule

$\{Send(a,b,m,stepid,sid)\}$
$\rightarrow \{Send(a,b,m,stepid,sid),Intrude(m)\}$

represents the message forwarding, and rule

$\{Intrude(m),Intrude(k)\}$
$\rightarrow \{Intrude(m),Intrude(k),Intrude(\{m\}_k)\}$

represents message encrypting, etc. All the rules compose the $ActionsSet_I$.

2.4 Representation of the goals

The goals of the security problem for cryptographic protocols are insecure states which we want to search. Our interest mainly focuses on the secrecy and authentication of protocols. The goal violating the security properties can be defined as below:

$SecrecyViolation=\{Intrude(\mathrm{m}),\ Not\ Shared(m,I,A),\ Not\ Shared(m,I,B)\ldots\}$

it means that the intruder has obtained the secret message m which he should not have.

When analyzing the protocol's authentication property, we adopt the Gavin Lowe's idea[7]. If the protocol running can reach a state containing the fact $End(a,b,authid(,Term)^*)$ but no corresponding fact $Begin(a,b,authid(,Term)^*)$, we say it violate the authentication property.

3 The Operational Semantic

When we solve the cryptographic protocol insecurity problem, two operations are mainly involved, they are rule application and rule unification.

3.1 Rule application

Definition 1(rule application) Let $R=PreCond\rightarrow PostCond$ be a first order rewriting rule, s be a state, if there exits a substitutionΘ, satisfying $\Theta PreCond\sqsubseteq s$, that is, $\forall fact\in PreCond,\ \Theta fact\in s$ holds, then we say rule R can be applied to state s. The new state s' generated by the rule application can be defined as:

$s'=[R]_\Theta(s)=(s\backslash\Theta PreCond)\cup\Theta PostCond$
$=\{f\,|\,f\in s\wedge f\notin\Theta PreCond\vee f\in\Theta PostCond\}$

When Θ is de-emphasis, $[R]_\Theta(s)$ can be simplified as $[R](s)$.

Definition 2 (fact space) The fact space FS is the set of all facts in the security problem for cryptographic protocols Π.

Definition 3 (state space) The state space SS is the set of all states in the security problem for cryptographic protocols Π.

For any state s in the state space SS, if $\exists f, f'\in$ s, and f=not f', the we say s is insignificant. All insignificant states constitute the set Ω.

Definition 4 (reachability) Given a state $s\in SS$, s is reachable if there exists a sequence of action $R_1,R_2,\ldots,R_n\in ActionsSet$ and substitutions $\Theta_1,\Theta_2,\ldots,\Theta_n$, $n>0$, satisfying $s=[R_n]_{\Theta n}(\ldots([R_2]_{\Theta 2}([R_1]_{\Theta 1}(s_0)))\ldots)$, $s_0\in InitStates$. $R_1,R_2,\ldots,R_n$ is called the path from s_0 to s. s is n-step reachable if it is reachable and n is the minimum integer.

Definition 5 (solvability) For a security problem Π=<$InitStates$, $ActionsSet$, $Goals$>, if there exists state $g\in Goals$, and g is reachable, then Π can be solved, and the paths are called solutions for problem Π.

Definition 6 (solution space) The solution space $Solutions$ is the set of all solutions for problem Π.

3.2 Rule unification

Definition 7 (rule increment) For rule $R=PreCond\rightarrow PostCond$ and fact set $F'\sqsubseteq FS$, if for any substitution Θ, $\Theta PreCond\cup F'\neq\Omega$ and $\Theta PostCond\cup F'\neq\Omega$ hold, then the rule $R'=PreCond\cup F'\rightarrow PostCond\cup F'$ is defined as increment rule of R, and F' is the increment of form R to R'. Also we can write $R'=R+F'$.

The increment relation is self-inverse and transferable, that is, each rule is the increment rule of itself, if R_1 is the increment rule of R_2, R_2 is the increment rule of R_3, then R_1 is the increment rule of R_3.

Proposition 1 If rule R_1 is the increment rule of R_2, and rule R_2 can be applied to state $s\in SS$, then rule R_1 can be applied to state s too, and $[R_1](s)=[R_2](s)$.

The Proposition can be proved according to Definition 7.

Definition 8 (rule unification) Given rules $R_1=PreCond_1\rightarrow PostCond_1$ and $R_2=PreCond_2\rightarrow PostCond_2$, if R_1 has increment rule R_1+F', $PreCond_1\cup F'$ and $PreCond_2$ are unifiable, and the unify of $\{PreCond_1\cup F', PreCond_2\}$is Θ, then rules R_1 and R_2 are unifiable, and the result of rule unification has the form of

$R_1\circ R_2=\Theta PreCond_1\cup\Theta F'\rightarrow\Theta PostCond_2$

Using rule unification through whole $ActionSet$, we can construct a more refined action set $ActionsSet^E$:

(1) Choose any rule R from $ActionsSet$, if $R\notin ActionsSet^E$, then add R into $ActionsSet^E$.

(2) For any rule R' in the $ActionSet^E$, if R' and R are unifiable, then add the resultant rule into $ActionSet^E$, go back to (1).

(3) When reaching a fixpoint, the process terminate.

With the $ActionsSet^E$, a new problem Π^E =<$InitStates$, $ActionSet^E$, $Goals$> can be constructed, where $InitStates$ and $Goals$ are identical with the corresponding elements in the original problem Π. Solving problem Π^E is less complicated than Solving Π because $ActionsSet^E$ contains less rules than $ActionSet$. If we can prove that problem

Π^E has solution iff Π has solution and two solutions are the same, complicated problem can be solved by transforming to a easier one.

Definition 9 (solution equivalence) Let *P* and *P'* be security problems, when *P* can be solved if and only if *P'* can be solved and their solutions have some partial order relations, *P* and *P'* have the equivalent solution.

Lemma 1 Assume *R* and *R'* are unifiable rules, the resultant rule is $R^\circ R'$. rule *R* can be applied to state $s \in SS$ if and only if $R^\circ R'$ can be applied to *s,* and $[R']([R](s))=[R^\circ R'](s)$.

Prove Let $R=PreCond_1 \rightarrow PostCond_1$, $R'= PreCond_2 \rightarrow PostCond_2$, $R^\circ R'=\Theta PreCond_1 \cup \Theta F' \rightarrow \Theta PostCond_2$.

Sufficiency If $R^\circ R'$ can be applied to *s,* according to definition 3, there exists a substitution θ, satisfying $\theta(\Theta PreCond_1 \cup \Theta F') \in s$, so $\theta\Theta PreCond_1 \in s$, that means rule *R* can be applied to state *s*.

Necessity If rule *R* can be applied to state *s,* then according to proposition 3, $Rule+F'$ can be applied to state *s* too, that is, there exists θ' , satisfying $\theta'(PreCond_1 \cup F') \in s$. Let $\theta''=\theta' \cdot \Theta^{-1}$, then $\theta''(\Theta PreCond_1 \cup \Theta F') =\theta'\Theta^{-1}\Theta(PreCond_1 \cup F') =\theta'(PreCond_1 \cup F') \in s$, that means rule $R^\circ R'$ can be applied to state *s*.

$[R']([R](s))=[R']([R+F'](s))$
$=[R']((State \backslash \theta'(PreCond_1 \cup F')) \cup \theta'(PostCond_1 \cup F'))$
$=(((State \backslash \theta'(PreCond_1 \cup F')) \cup \theta'(PostCond_1 \cup F')) \backslash \theta' PreCond_2) \cup \theta' PostCond_2$
$=(((State \backslash \theta'\Theta^{-1}\Theta(PreCond_1 \cup F')) \cup \theta'\Theta^{-1}\Theta(PostCond_1 \cup F')) \backslash \theta'\Theta^{-1}\Theta PreCond_2) \cup \theta'\Theta^{-1}\Theta PostCond_2)$
$=(((State \backslash \theta'\Theta^{-1}\Theta(PreCond_1 \cup F')) \cup \theta'\Theta^{-1}\Theta PostCond_2)$
$=[Rule^\circ Rule'](s)$ □

According to definition 5, definition 9 and lemma 1, we can draw a conclusion that problems Π and Π^E are solution equivalent. So we can refine our problem through rule unification.

Applying our problem solving method, we can find an attack sequence corresponding to following attack[8]:

(1) $A \rightarrow^1 I : \{A,N_A\} pubK_I$
(2) $I(A) \rightarrow^2 B : \{A,N_A\} pubK_B$
(3) $B \rightarrow^2 I(A) : \{N_A,N_B\} pubK_A$
(4) $I \rightarrow^1 A : \{N_A,N_B\} pubK_A$
(5) $A \rightarrow^1 I : \{N_B\} pubK_I$
(6) $I(A) \rightarrow^2 B : \{N_B\} pubK_B$

where the superscript of $\rightarrow$ is the identifier of protocol sessions.

4 Conclusion

Based on problem solving theory, we propose a model for cryptographic protocols. The model can precisely formulate cryptographic protocols and their security properties, it has reasonable and provable semantics. Combined the advantage of model checking and theorem proving, we believe that this method can provide new possibilities to analyze cryptographic protocols, and the security analysis of cryptographic protocols based on this model is reasonable and efficient.Direction for our furure work include efficient solving algorithm and automatic tool which can handle non-termination phenomena.

References:

[1] W.Marrero,E.Clarke, and S.Jha. Model checking for security protocols.*Technical Report CMU-CS-97-139*,School of Computer Science, Carnegie Mellon University, 1997.

[2] Mitchell, J.C. Finite-state analysis of security protocols, in A.J.Hu & M.Y.Vardi, eds, '*Computer Aided Verification (CAV-98): 10th International Conference*', Vol. 1427 of LNCS, Springer, 1998, pp. 71-76.

[3] Paulson, L. C. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security* 6, 1-2, pp. 85–128, 1998.

[4] C. Sprenger, M. Backes, D. Basin, B. Pfitzmann, and M. Waidner. , Cryptographically sound theorem proving. in *Proceedings of 19th IEEE CSFW,* 2006.

[5] C.Meadows, The NRL protocol analyzer: an overview. *Journal of Logic Programming*, 26(2), pp 113-131, 1996.

[6]B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. *In Procceedings of 14th Computer Security Foundations Workshop (CSFW'01)*, IEEE, 2001, pp. 82－96.

[7] Lowe G. A Hierarchy of Authentication Specifications. In *Proceedings of the 10th Computer Security Foundations Workshop(CSFW'97)*, Rockport, Massachusetts, 1997.

[8] G.Lown. Breaking and fixing the Needham-Schroeder public-kry protocol using FDR. In *Proceedings of TACAS, volume 1055 of Lecture Notes in Computer Science*, Springer-Verlag,1996. pp 147-166.