Automated Proof of Resistance of Denial of Service Attacks Using Event with Theorem Prover

Bo Meng
South-Center University for Nationalities/ School of Computer, Wuhan, China
Email: mengscuec@gmail.com

Wei Huang and Zimao Li
South-Center University for Nationalities/ School of Computer, Wuhan, China
Email: {hwaoding2002@yahoo.com.cn, lizm@sdu.edu.cn}

Abstract—**The huge damage of denial of service attacks in security protocols attracts researchers' attention and effort to analysis, verification and prevention of denial of service attacks. In order to model resistance of denial of service attacks, firstly, we extend applied pi calculus from both adversary context and processes aspects; secondly, the first computer-aided method of resistance of denial of service attacks based on event is proposed from the angle of state in security protocols by us; finally, the analysis using ProVerif indicates that JFK protocol is against of denial of service attacks but IEEE 802.11 i four-way handshake protocol is not, and simultaneously, a new denial of service attack is firstly detected, together with methods to prevent it in IEEE 802.11 i four-way handshake protocol.**

Index Terms—**Automatic Verification, Event, Symbolic Model, Availability**

I. Introduction

In virtue of the huge harm of denial of service attacks in security protocols and networks, people have paid serious attention to analysis, verification and prevention of denial of service attacks. For the sake of preventing denial of service attacks, the opening is to use the formal method to finish analysis and proof of resistance of denial of service attacks in security protocols and networks, which enhances the public's confidence in its security.

In symbolic model there exist two important formal models about resistance of denial of service attacks: Yu-Gligor model [1] and Meadows model [2]. Yu-Gligor model [1] is founded on user agreement. The idea of the model is based on access control policy. It can not tackle denial of service attacks that are performed before authentication between originator and responder in security protocols, for instance, SYN floods attacks. And also it does not get the help of the automated tools. The other is Meadows model [2] is founded on the fail-stop protocol. Researchers have paid much attention to it. Meadows deem that it can be applied by the modified automated tools, for example, NRL protocol analyzer. Tritilanunt et al. [3, 4] and Tritilanunt argue that the cost analysis has only related to the honest executions of the protocols in Meadows model. And simultaneously, they consider that he uses a coarse assess of computational cost with three ranks of cheap, medium and expensive, and it is difficult to assess operations in such a coarse measure in practice.

There are mainly two methods in symbolic model: theorem proving and model checking. Model checking is bases on constructing a finite state model of a system and then verifying that a claimed property holding in that model. In other words, the verification is implemented as an exhaustive state space search. In contrast to theorem proving, model checking is completely automatic and fast, but there is a hard problem that is state exposition problem. In theorem proving the system and its claimed properties are formalized as formulas in some mathematical logic. In comparison with model checking, theorem proving can tackle infinite state spaces. At the same time it can also with the support of automatic tools in proof, for example, ProVerif [5], Isabelle, HOL, ACL2, and PVS. The first order theorem prover ProVerif is a computer aided proof of security protocol verifier which use Horn clauses or applied pi calculus to model the security protocol. It can handle many different cryptographic primitives and can also tackle an unbounded number of sessions in security protocols and an unbounded message area. If ProVerif does not prove a claimed security property, then it detects an attack, in other words, a run trace of the security protocol fakes the claimed property. ProVerif has been applied on many protocols with good results [6-10,33-35].

Due to many denial of service attacks launched by protocol state, for example, the attacker sends many fake requests to the server, resulting in that the server exhausts its resources to process and maintain the malicious

Corresponding author: Bo Meng, School of Computer, South-Center University for Nationalities, Wuhan, China, 430074

The Project was Supported by the Special Fund for Basic Scientific Research of Central Colleges, South-Central University for Nationalities:CZY12017

requests state and can not accept requests from legitimate clients; the attacker alters the protocol state and makes it inconsistence, resulting the improper execution of the protocol, hence it is reasonable to model the resistance of denial of service attacks from the view of protocol state based on events, which is different from Yu-Gligor model and Meadows model.

In order to model the protocol state and resistance of denial of service attacks, from two aspects: one is the adversary context, the other is process expression, we extend applied pi calculus .Then using the event, the first computer-aided method of resistance of denial of service attacks is proposed from the angle of state in the paper. The contributions of us are summarized as follows:

❆ Review the formal models of resistance of denial of service attacks in security protocols, among which two formal frameworks are focused: Yu-Gligor model and Meadows's cost-based model. To our knowledge, until now it does not existing resistance of denial of service attacks analysis model in computational model.

❆ In order to model the protocol state and resistance of denial of service attacks using events, we extend applied pi calculus from two aspects: one is the adversary context, the other is process expression.

❆ Based on our proposed extended applied pi calculus, from the angle of protocol state, we further present for the first time a computed aided method of resistance of denial of service attacks based on event.

❆ We analyze the Resistance of denial of service attacks in JFK protocol [11] and IEEE 802.11 i four-way handshake protocol with ProVerif by our formal method. The results are that JFK protocol is resistance of denial of service attacks and IEEE 802.11 i four-way handshake protocol is not resistance of denial of service attacks. And simultaneously, a new denial of service attack in IEEE 802.11 i four-way handshake protocol is found by us, and the methods to prevent such denial of service attacks in IEEE 802.11 i four-way handshake protocol are proposed.

II. Related Work

In symbolic model Yu-Gligor model building on user agreement and Meadows model based on cost are mainly two formal frameworks in resistance of denial of service attacks. Resistance of denial of service attacks analysis based on the computational model does not existing to our knowledge.

The first works on formalize resistance of denial of service attacks were finished by Gligor [12, 13] based on maximum waiting time. For a maximum specified waiting time in operations, he uses availability to promise it. If a system is resistance of denial of service attacks then any requesting user will wait no more than maximum waiting time units of time before the service is granted.

Based on temporal logic, through introduction of user-agreement, Yu and Gligor [1] present a formal specification on resistance of denial of service attacks. The user-agreement specifications describe all the desired properties. They use their framework to analyze denial of services attacks in resource allocator in operating system and Dining philosophers' service. But it does not deal with the denial of service attacks that are performed before authentication between originator and responder in security protocols, for instance, SYN floods attacks. And also it does not get the help of the automated tools. Bacic and Kuchta [14] think that resource allocation plays an important role in addressing resistance of denial of service attacks. They introduce three reference monitor conditions: the first one is that it is tamper-proof; the second one is that it cannot be against from operation; the final one is that it can protects authorized access to resources. Through expressing the passage of time, Millen [15] extend Yu-Gligor model. And simultaneously He uses a resource allocation model to model resistance of denial of service attacks and thinks that a denial of service protection base is same to trusted computing base. Abadi and Lamport [16] introduce a similar idea. But Millen' model is based on a set-theoretic approach and passage of time.

Through the costs spending on computation implemented by the principles in security protocols, Meadows [2] introduces a formal framework of resistance of denial of service attacks. His formal model is based on fail-stop protocol that can provide a certain extent of security against attacks and will stop if the modified message is detected or the verification is failed. According to Meadows's framework when in a protocol run at which an adversary sends a message to launch a denial of service attack if the cost of generating the message is small with respect to his initial resources, while the cost of processing and verifying the message send by the attacker is relatively costly. If this relationship of costs between attacker and defender is not true during a protocol execution, then it is resistance of denial of service attacks. And simultaneously, he argues that it can get the help of the modified automated tools, for example, NRL protocol analyzer. He also gives an analysis of the station to station protocol and points out that it is not resistance of denial of service attacks. But Owning to the costs of generating a fake message is small than costs of processing and verifying it, so every protocol is not resistance of denial of service attacks. Hence Meadows model maybe not practical. Based on Meadows model, Ramachandran [17] points that JFK protocol is resistance of denial of service attacks under the condition that fake messages are handled in a good way. Smith et al. [18] also consider that JFK protocol is denial of service attacks in the presence of attackers who conceal their source IP addresses. This is because that availability of IP addresses makes the cost of revealing an address more expensive to an initiator. But we think that these arguments are worth discussing. Lafrance and Mullins [19] use admissible interference to detect denial of service attacks in security protocols. Using SPPA and Meadows framework, they introduce impassivity to detect whenever an attacker process may cause interference. Their model is suitable to formalize resource exhaustion denial of service attacks. They point out that 1kp electronic payment protocol is not resistance of denial of service attacks. Abadi et al. [6] formalize by hand denial of service attacks through observational equivalence relation and find JFK protocol is resistance of denial of service attacks. Tritilanunt et al. [3, 4] and Tritilanunt argue that the cost analysis has only related to the honest executions of the protocols in Meadows model. And simultaneously, they consider that he uses a coarse assess of computational cost with three ranks of cheap, medium and expensive, and it is difficult to assess operations in such a coarse measure in practice. So based on and model based on time and cost, they use the colored Petri nets to formalize the denial of service attacks and find that HIP protocol is not resistance of denial of service attacks in the conditions Type 3 adversary or Type 4 adversary. Zhou et al. [20] use strand spaces to analyze four-way handshakes protocol and find that it is not resistance of denial of services attacks. Groza and Minea[21] introduce several cost-based rules and use resource exhaustion to formalize denial of service attacks. They analyze station to station protocol and JFK protocol and point out that station to station protocol is not resistance of denial of service attacks and JFK protocol is.

Besides Yu-Gligor formal model and Meadows's cost-based formal model, according to prevent (p, c) policies, Amoroso [22] points that it is need a service model. Denial of service attacks policies are specified based on predicates that specify conditions. He analyzes resistance of denial of service attacks in system V/MLS with prevent (2, 2).

Cuppens and Saurel [23] formalize availability policy Based on modal logic and deontic logic by the four predicates expressions. It can make user to model availability properties, and to formally verify these properties by simulating its logical specification. Gabillon and Gallon [24] resemble the Cuppens and Saurel model. The difference is that they do not use an explicit waiting time policy. They model an availability policy as a special case of security policy. Cuppens et al. [25] use the Nomad model which can transform an insecure program into a secure program to specify availability requirements. They mainly concern the denial of service attacks in program.

Agha et al. [26] formalize denial of service attacks through PMAUDE and a sublogic of Continuous Stochastic Logic to express the degree of success of attack and use the statistical model-checking tool VESTA to analyze TCP three-way handshaking protocol and find it is not resistance of denial of services attacks. Mahimkar and Shmatikov [27] use the alternating time temporal logic to verify JFKr with the help of the model checker MOCHA and find that it is resistance of denial of service attacks.

Meng et al. [28] propose a formal method of resistance of denial of service attacks with ProVerif. The idea is similar to our current paper. But they use the standard channel in applied pi calculus and do not use the events. And simultaneously, it is hard to put into practice and requires the higher ability of the user in extended applied pi calculus and ProVerif.

III. Extended Applied Pi Calculus

Extended applied pi calculus is based on applied pi calculus [29] that is used to express concurrent processes and their interactions based on Dolev-Yao model. Applied pi calculus is an extension of the pi calculus that has the constructs for communication and concurrency from the pure pi calculus. It inherits the constructs for generating statically scoped new names and permits proof technology and a general development of syntax, operational semantics equivalence. At the same time there are several powerful automatic tool supported applied pi calculus, for example, ProVerif. Applied pi calculus with ProVerif has been used to analyze many complicated security protocols.

In order to model the protocol state and resistance of denial of service attacks, we extend the applied pi calculus from two aspects: one is the adversary context, the other is process expression. The semantic of extended applied pi calculus is same to the semantic of the applied pi calculus and can also get the help of ProVerif.

A. Adversary contexts

In applied pi calculus the adversary is in Dolev-Yao model. But in extended applied pi calculus, in term of abilities of attacker, the contexts of adversary are categorized into two contexts: one is ideal context, the other is real context. Real context is modeled as $\nu\tilde{n}.C\left[C\left[\overline{c}\langle u\rangle\right]\overline{u}\langle N\rangle.P\right],\nu\tilde{n}.C\left[C\left[c(u)\right]u(x).P\right]$, where $u\in\tilde{n},c\notin\tilde{n}$. In other words, in real context, if the sender wants to publish the message N, then firstly he must publish the channel name u in the public channel c, and then he publishes message N through the channel u; if the receiver wants to receive the message N, he must get channel name u in the public channel c, then he gets the message N from channel u. Intuitively, real context is insecure environments. The adversary in real context is in Dolev-Yao model.

Ideal context is modeled as $\nu\tilde{n}.C\left[\overline{u}\langle N\rangle.P\right],\nu\tilde{n}.C\left[u(x).P\right]$, where $u\in\tilde{n}$. In other words, the sender publishes message N through channel u directly; the receiver receives message N through channel u directly. The security of message N depends on the channel u that is a secure channel. Intuitively ideal context is secure environments. The attacker in ideal context can not overhear, intercept, and synthesize any message.

B. Plain process

In extended applied pi calculus, it consists of plain processes and extended processes. Plain processes in Fig.1 are constructed in a way that is consist to the way in the pi calculus, besides that messages can include terms and that names need not be merely channel names.

Here we only introduce the conditional construct in idea context and in real context. The description and explanation about other plain process express in Figure 1 can be found in reference [32]. The conditional construct $if\ M=N\ then\ P\ else\ Q$ executes that if M is equal to N, then executes P, otherwise executes Q in real context. The conditional construct $if\ M = N\ \ then\ P\ else\ C\left[existdos\left(M,N\right)\right].Q$ executes that if if M is equal to N, then executes P, otherwise runs $C\left[existdos\left(M,N\right)\right]$ in idea context.

$P,Q,R ::=$	plain processes
0	null process
$Q\vert P$	parallel composition
$!P$	replication
$\nu n.P$	name restriction
$if\ M = N\ \ then\ P\ else\ Q$	conditional in real context
$if\ M = N\ \ then\ P\ else\ C\left[existdos\left(M,N\right)\right].Q$	conditional in idea context
$\mathrm{in}\left(u,x\right).P$	message input
$out\left(u,N\right).P$	message output
$\mathrm{event}\left(M\right).P$	event

Figure 1. Plain process

Generally an event is used to mark the important steps of the security protocols in research, while it does not make any affect on behaviors of security protocols. It can be used to record the context of the sending or receiving message in security protocols. In extended applied pi calculus, event $\mathrm{event}\left(M\right)$ just outputs message M through a special channel. So event $\mathrm{event}\left(M\right)$ does not expose M to the attacker. Hence, the run of the process P after introducing events is the run of P without events, add the recording of event $\mathrm{event}\left(M\right)$. The process $\mathrm{event}\left(M\right).P$ firstly runs the event $\mathrm{event}\left(M\right)$, and then runs P.

In order to record the position where the denial of service attack is occurred, the event $existdos\left(M,N\right)$ is introduced. If the event $existdos\left(M,N\right)$ occurred, then the denial of service attacks can be launched by the message where the verification operation $\left(M,N\right)$ is processed.

C.Process context

$C ::=$	process context
$[\ ]$	null process context
$P\vert C$	parallel composition
$C\vert Q$	parallel composition
$!C$	replication
$\upsilon n.C$	name restriction
$if\ \ M = N\ then\ C\ else\ Q$	conditional
$if\ \ M = N\ then\ P\ else\ C$	conditional
$\mathrm{in}\left(u,x\right).C$	message input
$out\left(u,N\right).C$	message output

Figure 2. Process context

Here we only introduce the conditional process context. The description and explanation about other process context in Figure 2 can be found in reference [32].

The conditional construct $if\ \ M = N\ \ then\ C\ else\ Q$ executes that if M is equal to N, then runs process context C, and then C is a verified context. The conditional construct $if\ \ M = N\ \ then\ P\ else\ \ C$ executes that if M is not equal to N, runs C, and then C is not a verified context.

IV. Formalize Protocols and Definitions of Resistance of Denial of Service Attacks

❁ Definition 1: an annotated Alice-and-bob specification in protocols

An annotated Alice-and-bob specification in protocols consists of n statements of form $A \rightarrow B: R_1^i,\cdots,R_m^i \,\|\, M_i \,\|\, O_1^i,\cdots,O_k^i$, where $i \in \left[1,n\right]$, M_i denotes the ith message in protocol. This definition is borrowed from Meadows [2].

Protocol consists of n messages exchanged between two principles A and B. A statement $A \rightarrow B: R_1^i,\cdots,R_m^i \,\|\, M_i \,\|\, O_1^i,\cdots,O_k^i$ describes that firstly the sequences of operations $R_1^i,\cdots,R_m^i$ executed by principles A to generate a message M_i, and then it is sent to principle B, finally the sequence of operations $O_1^i,\cdots,O_k^i$ executed by principle B. $R_1^i,\cdots,R_m^i$ denotes the sequence of operations executed by principle A for generating M_i. $O_1^i,\cdots,O_k^i$ denotes the sequence of operations performed by principle B after receiving M_i and processing and verifying M_i.

Let $l = A \rightarrow B: R_1^l,\cdots,R_m^l \,\|\, M_l \,\|\, O_1^l,\cdots,O_k^l$ is an annotated Alice-and-bob specification in security protocols, $act_l\left(A\right)$ is a set of operations performed by principle A on l. $act_l\left(A\right)\left[R_1^l,\cdots,R_m^l,M_l\right]$ denotes that the set of the sequence of operations $R_1^l,\cdots,R_m^l$ preceding principle A sends message M_l to B. $act_l\left(B\right)\left[M_l,O_1^l,\cdots,O_k^l\right]$ denotes that the set of the sequence of operations $O_1^i,\cdots,O_k^i$ performed by principle B after receiving M_l. If any verification operations, for example, decryption, verification of digital signature, failed, and then the operation stops.

❁ Definition 2: authentication of message M_l

If the statement $l = A \rightarrow B: R_1^l,\cdots,R_m^l \,\|\, M_l \,\|\, O_1^l,\cdots,O_k^l$ carries out successfully then the fact that principle B receives message M_l from A exists; if principle B receives message M_l, but principle A does not perform the sequence of operations $act_l\left(A\right)\left[R_1^l,\cdots,R_m^l,M_l\right]$, then message M_l received by principle B is altered by the adversary; If message M_l received by principle B is altered by the adversary and B can find the fact that message M_l is modified , then message M_l received by principle B is authenticated.

❅ Definition 3: correspondence in operations

The relation between α and β is correspondence in operations if and only if message M_i in $act_i(A)\left[R_1^i,\cdots,R_m^i,M_i\right]$ is same to message M_j in $act_j(B)\left[M_j,O_1^j,\cdots,O_k^j\right]$, where $i,j\in[1,n]$, $\alpha\in act_i(A)\left[R_1^i,\cdots,R_m^i,M_i\right]$ and $\beta\in act_j(B)\left[M_j,O_1^j,\cdots,O_k^j\right]$. $act_i(A)\left[R_1^i,\cdots,R_m^i,M_i\right]$ denotes that the set of the sequence of operations for generating message M_i by principle A . $act_j(B)\left[M_j,O_1^j,\cdots,O_k^j\right]$ denotes that the set of the sequence of operations for processing and verifying message M_j by principle B .

❅ Definition 4: operation γ_1 casually precedes operation γ_2

P is an annotated Alice-and-bob specification in a protocol, S is a set of all operations in P . For any operation γ_1 and γ_2 in S , γ_1 casually proceeds γ_2 if and only if:

1. If $\gamma_1,\gamma_2\in act_i(A)\left[R_1^i,\cdots,R_m^i,M_i\right]$ or $\gamma_1,\gamma_2\in act_j(B)\left[M_j,O_1^j,\cdots,O_k^j\right],i,j\in[1,n]$, at the same time γ_1 occurred before γ_2 ;
2. If $\gamma_1\in act_i(A)\left[R_1^i,\cdots,R_m^i,M_i\right]$, $\gamma_2\in act_j(B)\left[M_j,O_1^j,\cdots,O_k^j\right]$, $i,j\in[1,n]$, at the same time the relation between γ_1 and γ_2 is correspondence.
3. There exists operations γ_3 , γ_3 casually precedes γ_2 , γ_1 casually precedes γ_3 .

$act_i(A)\left[R_1^i,\cdots,R_m^i,M_i\right]$ denotes that the set of the sequence of operations for generating the message M_i by principle A . $act_j(B)\left[M_j,O_1^j,\cdots,O_k^j\right]$ denotes that the set of the sequence of operations for processing and verifying the message M_j by principle B .

❅ Definition 5: set of association in message M_i and M_j .

Set of association ω between any message M_i and M_j in protocol P is intersection of set $\mho$ and set ψ : $\omega=\mho\cap\psi$,where $i,j\in[1,n]$, $i<j$, $\mho$ is set of data items in verification operations v in $act_j(B)\left[M_j,O_1^j,\cdots,O_k^j\right]$, ψ is the set of data items in message M_i in $act_j(B)\left[M_j,O_1^j,\cdots,O_k^j\right]$.

Intuitively, set of association ω describes the degree of association among messages in protocol. If ω is null, relations of messages in protocol P are independent and are not mutually associated, thus the protocol P is stateless. If ω is not null and includes many data items, relations between messages M_i and M_j are associated deeply.

❅ Definition 6: resistance of denial of service attacks in protocols

P is an annotated Alice-and-bob specification in a protocol, responder B is resistance of denial of service attacks if and only if set of association ω between any messages M_i and M_j in set *Recv(B)* has the fallowing conditions:

1. ω is null set $\varnothing$;
2. all data items in ω are authenticated.

Where *Recv(B)* is set where data items are in operations that are ordered in casually precedes in $act_j(B)\left[M_j,O_1^j,\cdots,O_k^j\right]$, where $i,j\in[1,n]$ and $i<j$. Intuitively, if any two messages M_i and M_j in protocol P are not associated each other, then contexts of processing messages M_i and M_j are independent, hence B is resistance of denial of service attacks; if any messages M_i and M_j in protocol P are associated, then contexts of processing and verifying messages M_i and M_j are not independent, hence B is resistance of denial of service attacks if and only if it has the above two conditions.

V. Automated Proof of Resistance of Denial of Service Attacks Based on Event

The protocol can be formalized as an annotated Alice-and-Bob specification with the extended applied pi calculus. We suppose that the protocol exchanges $2n$ messages between principle Alice and Bob in an execution. Principle Alice sends n messages M_i where $i\in[1,n]$ and receives n messages $M_i^{'}$ where $i\in[1,n]$. Principle Bob receives n messages M_i where $i\in[1,n]$ and sends n messages $M_i^{'}$ where $i\in[1,n]$. Protocol process $PP\equiv\nu\tilde{n}.(!Alice\,|\,!Bob)$ is a closed process and includes parallel composition of any processes *Alice* and processes *Bob* . In term of the extended applied pi calculus, initiator process *Alice* and responder process *Bob* can be reduced into some one process in Figure 3.

For the sake of using ProVerif to automatically prove resistance of denial of service attacks of Bob , any message M_i where $i\in[1,n-1]$ is formalized with extended applied pi calculus. If query event $existdos(M,N)$ is true, attacker can launch a denial of service attack through an attack of message M_i when the verification operation (M,N) is processing.

$$
\begin{array}{ll}
Alice, Bob(\rightarrow \cup \equiv)^* 0 & Alice, Bob(\rightarrow \cup \equiv)^* !P \\
Alice, Bob(\rightarrow \cup \equiv)^* \nu n.P & Alice, Bob(\rightarrow \cup \equiv)^* P \mid P' \\
Alice, Bob(\rightarrow \cup \equiv)^* c(x).P & Alice, Bob(\rightarrow \cup \equiv)^* \bar{c}\langle N\rangle.P \\
\multicolumn{2}{l}{Alice, Bob(\rightarrow \cup \equiv)^* if\ \ M=N\ then\ P\ else\ Q} \\
\multicolumn{2}{l}{Alice, Bob(\rightarrow \cup \equiv)^* if\ \ M=N\ then\ P\ else\ C\left[existdos(M,N)\right].Q \quad c\notin\tilde{n}}
\end{array}
$$

Figure 3. Process

The message M_i in a protocol is swapped and handled in real context in Figure 4. The messages $M_1, M_1^{'}, \cdots, M_{i-1}, M_{i-1}^{'}, M_i, M_{i+1}, M_{i+1}^{'}, \cdots, M_n, M_n^{'}$ are swapped and handled in ideal context. Protocol process PP is $PP \equiv \nu\tilde{n}.(!Alice_i \mid !Bob_i)$, c is public channel. c_j ($j\in[2,n-1]\cap j\neq i$) are private channels used by Bob to get messages M_j ($j\in[2,n-1]\cap j\neq i$). $Alice_i(\rightarrow\cup\equiv)^* C\left[\bar{c}\langle c_i\rangle\right]\bar{c}_i\langle M_i\rangle.Alice_{i+1}, c\notin\tilde{n}, c_i\in\tilde{n}$, $Bob_i(\rightarrow\cup\equiv)^* C\left[c(x)\right]x(m_i).Bob_{i+1}, c\notin\tilde{n}$, $Alice_j(\rightarrow\cup\equiv)^* C\left[\bar{c}_j\langle M_j\rangle\right]Alice_{j+1}$ $, c_j\in\tilde{n}, j\in[1,n]\cap j\neq i$, $Bob_j(\rightarrow\cup\equiv)^* C\left[c_j(m_j)\right]Bob_{j+1}, c_j\in\tilde{n}, j\in[1,n]\cap j\neq i$ .If query event $existdos(M,N)$ is true, and then the adversary can launch a denial of service attack by an attack of message M_i $i\in[2,n-1]$.

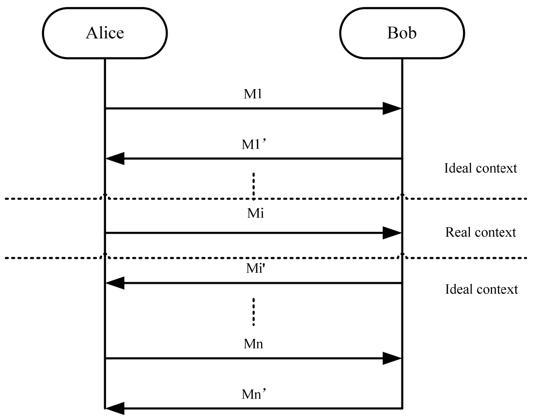


Figure 4. The model of messages M_i $i\in[1,n-1]$

❁ Theorem: resistance of denial of service attacks

Responder Bob in protocol process PP is resistance of denial of service attacks if and only if all query event $existdos(M,N)$ are not true in the formal model of all messages $M_i, i\in[1,n-1]$ received by principle Bob in PP . In other words, there does not exist processes P' , P'' and attacker process $Attacker$ to cause $(PP \mid Attacker)(\rightarrow\cup\equiv)^* existdos(M,N).P' \mid P''$, $c\notin\tilde{n}, S\in\tilde{n}$.

Proof:

If Bob has the ability of resistance of denial of service attacks, according to definition 6 of resistance of denial of service attacks, $\forall M_i, M_j \in Recv(Bob)$, $i,j\in[1,n]$, M_i casually precedes M_j ,verification operation v is $if\ \ M=N\ then\ P\ else\ C\left[existdos(M,N)\right].Q\in act(Bob)[M_j]$, then for the formal model of message M_i :

(1) $\omega=\varnothing$. In other words, the value of $M=N$ in the annotated Alice-and-bob specification in protocol is not related to message M_i , hence whatever M_i is exchanged in ideal context or real context, the value of $M=N$ is always true, protocol process $PP(\rightarrow\cup\equiv)^* P$.

(2) $\omega=\{\tilde{m}\}$, where $\tilde{m}$ is authenticated data items. In other words, attacker can not alter $\tilde{m}$, hence the value of $M=N$ is always true whatever M_i is exchanged in idea context or real context, protocol process $PP(\rightarrow\cup\equiv)^* P$.

Hence for the model of all messages $M_i, i\in[1,n-1]$ received by principle Bob , in PP all query event $existdos(M,N)$ are not true.

For the model of all messages $M_i, i\in[1,n-1]$ received by principle Bob , if in PP all query event $existdos(M,N)$ are not true, then we can conclude that there exist processes P' , P'' and attacker process $Attacker$ to cause $(PP \mid Attacker)(\rightarrow\cup\equiv)^* existdos(M,N).P' \mid P'', c\notin\tilde{n}$ .Hence attacker $Attacker$ can make that the value of $M=N$ is always false by altering the message which is casually precedes M_j and is associated to v . Then set of association ω has the data items $\tilde{m}'$ which are not authenticated, so responder Bob in protocol process PP is not resistance of denial of service attacks.

In term of the theorem in PP ,if query event $existdos(M,N)$ is true, attacker can construct a denial of service attack by altering the message M_i which make the receiver can not find without influence on other messages exchanges in protocol.

Hence people can use the extended applied pi calculus to model resistance of denial of service attacks in protocol, then in term of the proposed theorem, apply ProVerif to automatically verify and prove the resistance of denial of service attacks.

VI. CASE: JUST FAST KEYING PROTOCOL

JFK protocol is a key exchange protocol proposed to replace IKE as the standard key exchange protocol for the IPSec protocol suite. It claims that it has security, privacy, perfect forward secrecy and resistance of denial of service attacks.

JFK protocol includes two principals that play the roles of an initiator called Alice and a responder called Bob. This is consistent to the Alice-and-Bob specification. Alice and Bob want to establish a secure communication channel by establishing a shared secret key through Deffie-Hellman key exchange protocol. This shared secret key is used to generate session keys. Alice and Bob authenticated mutually by the shared secret key, at the same time Alice and Bob also have a agreement on various related communication parameters. Attackers are modeled in Dolev-Yao model, so they can eavesdrop, delete, and insert messages; at the same time they may also try to pretend to be Alice or Bob.

JFK protocol consists of two major versions, JFKr and JFKi. Their difference is that protection of identities private information. The objective of JFKr is to prevent the responder' identity from the active attacks and is to prevent initiator' identity against the passive attacks. JFKi prevents the initiator' identity from the active attacks. Alice-and-bob specification in simplified JFKr protocol is in Figure 5.

$$\begin{aligned}
&message1\text{:}\ \ Alice \to Bob : N_{Alice}, x_{Alice}\\
&message2\text{:}\ \ Bob \to Alice : N_{Alice}, N_{Bob}, x_{Bob}, g_{Bob}, t_{Bob}\\
&message3\text{:}\ \ Alice \to Bob : N_{Alice}, N_{Bob}, x_{Alice}, x_{Bob}, t_{Bob}, e_{Alice}, h_{Alice}\\
&message4\text{:}\ \ Bob \to Alice : e_{Bob}, h_{Bob}
\end{aligned}$$

Figure 5. JFKr protocol

where $x_{Alice} = g^{d_{Alice}}$ is the Diffie-Hellman exchange values of $Alice$; $x_{Bob} = g^{d_{Bob}}$ is the Diffie-Hellman exchange values of Bob , $t_{Bob} = H\{K_{Bob}\}(x_{Bob}, N_{Bob}, N_{Alice})$ is authenticator cookie used by Bob against denial of service attacks; $H\{\ \}(\ )$ is message authentication codes. K_{Bob} is Bob's secret hash key for authenticators t_{Bob} ; N_{Bob} is nonce for the session generated by Bob ; N_{Alice} is nonce for the session generated by $Alice$; $K_u = H\{{x_{Bob}}^{d_{Alice}}\}(N_{Alice}, N_{Bob}, u)$ is shared key obtained through Diffie-Hellman computation for $u = a, e, v$ that are three kinds of functions for keys: authentication, encryption and main session secret; $e_{Alice} = E\{K_e\}(ID_{Alice},\ ID'_{Bob},\ sa_{Alice},\ s_{Alice})$ is the ciphertext of payload messages and their MACs of Alice with shared key K_e ; $e_{Bob} = E\{K_e\}(ID_{Bob},\ sa_{Bob},\ s_{Bob})$ is the ciphertext payload messages and their MACs of Bob with shared key K_e ; $h_{Alice} = H\{K_a\}(i\ ,\ e_{Alice})$ is message authentication codes of $(i\ ,\ e_{Alice})$; $h_{Bob} = H\{K_a\}(r\ ,\ e_{Bob})$ is message authentication codes of $(r\ ,\ e_{Bob})$; i and r are two different constants used to distinguish initiator and responder MACs. $s_{Alice} = S\{K_-^{Alice}\}(N_{Alice},\ N_{Bob},\ x_{Alice},\ x_{Bob},\ g_{Bob})$ is signed nonce and exponentials with private key of Alice K_-^{Alice} , $s_{Bob} = S\{K_-^{Bob}\}(x_{Bob},\ N_{Bob},\ x_{Alice},\ N_{Alice})$ is signed nonce and exponentials with private key of $Bob\ K_-^{Bob}$.

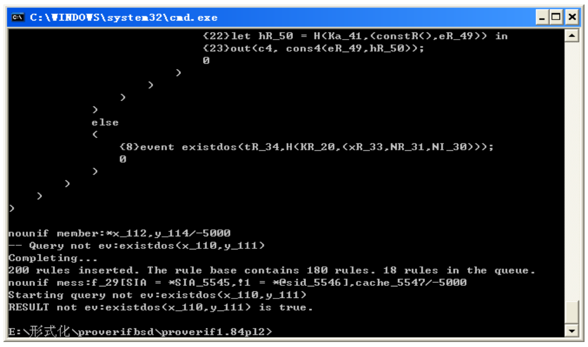
JFKr protocol consists of four messages. In other words, there are two pair messages in JFKr protocol. Message1 and message2 establishes a shared secret key between $Alice$ and Bob by Diffie-Hellman exchange protocol. Message3 and message 4 are used to authenticate $Alice$ and Bob mutually.

In message 1 $Alice$ generates random fresh nonce N_{Alice} for the session and Diffie-Hellman exchange values of $Alice$ $x_{Alice} = g^{d_{Alice}}$ and sends it to Bob . In message 2 Bob generates random fresh nonce N_{Bob} for the session, Diffie-Hellman exchange values of Bob $x_{Bob} = g^{d_{Bob}}$, authenticator cookie $t_{Bob} = H\{K_{Bob}\}(x_{Bob}, N_{Bob}, N_{Alice})$ and g_{Bob} in Diffie-Hellman exchange, and then sends message 2 to Alice. $(x_{Bob}, N_{Bob}, N_{Alice})$ is the context of Bob when Bob receives message1 and sends message 2 successfully. After receiving message 1 and message2 $Alice$ and Bob get the shared secret key $g^{d_{Alice}d_{Bob}}$. We can note that the Bob does not need to generate any state information at this step. This is meant to ensure that Bob is resistance of denial service attacks.

Message 3 and 4 include encrypted signatures of the nonce, exponentials, and other material. $Alice$ generates message 3 including authenticator cookies t_{Bob} and sends it to Bob . The authenticator cookie t_{Bob} is used by Bob to verify the authentication of the responded data and makes sure that $Alice$ has sent Message3 using the same address as in Message1. Bob checks authenticator cookie t_{Bob} .If the result is true, and then he generates message 4 including encrypted payload messages and their MACs e_{Bob} message authentication codes h_{Bob} , and sends it to $Alice$.

Based on the automated method of resistance of denial of service attacks, the message 1 received by responder Bob is formalized with the extended applied pi calculus, and then it is translated into the ProVerif inputs in the extended pi calculus. ProVerif has two formats: in the form of Horn clauses and applied pi calculus as input. The second one is a process in an extension of the pi calculus [30]. The two output of ProVerif is essentially identical.

Due to the space limitations, the only result of resistance of denial of service attacks in JFKr protocol is given in Figure 6. The codes of JFKr protocol in inputs of ProVerif is in Appendix A.



```
C:\WINDOWS\system32\cmd.exe
                                        {22}let hR_50 = H(Ka_41,(constR(),eR_49)) in
                                        {23}out(c4, cons4(eR_49,hR_50));
                                        0
                                   )
                              )
                         )
                    )
                    else
                    (
                         {8}event existdos(tR_34,H(KR_20,(xR_33,NR_31,NI_30)));
                         0
                    )
               )
          )
)

nounif member:*x_112,y_114/-5000
-- Query not ev:existdos(x_110,y_111)
Completing...
200 rules inserted. The rule base contains 180 rules. 18 rules in the queue.
nounif mess:f_29[SIA = *SIA_5545,!1 = *@sid_5546],cache_5547/-5000
Starting query not ev:existdos(x_110,y_111)
RESULT not ev:existdos(x_110,y_111) is true.

E:\形式化\proverifbsd\proverif1.84pl2>
```

Figure.6. The result of resistance of denial of service attacks in JFKr protocol

According to our definition 6, owning to that query event $existdos(x, y)$ is not true, so the result obtained is that JFKr protocol is resistance of denial of service attack.

According the formal model of messages $message1$, the operations on supplicant Bob process messages $message1$ and $message3$ is $Recv(Bob)\left\{\begin{array}{l} act(Bob)\left[M_1, O_1^1, ..., O_i^1, ..., O_n^1\right], \\ act(Bob)\left[M_3, R_1^3, ..., R_i^3, ..., R_k^3\right] \end{array}\right\}$, where v is the verification operation $t_{Bob} = H\{K_{Bob}\}(x_{Bob}, N_{Bob}, N_{Alice})$, $(x_{Bob}, N_{Bob}, N_{Alice}) \notin message_1$. So the set of association of $message1$ and $message3$ is null set. This is consistent with the output of public channels c in ProVerif. According to definition 6 supplicant Bob in JFKr protocol is resistance of denial of service attacks.

VII. CASE: 802.11 I FOUR-WAY HANDSHAKE PROTOCOL

Wireless Local Area Networks is very important in our digital society. IEEE 802.11 standard aims to provide the secure communication channel between principles in the 802.11i protocol. The purpose of the IEEE 802.11i is to improve the security aspect of IEEE 802.11. Besides including key management and establishment, it also includes improvements of encryption and authentication. There are three parties: the supplicant, the authenticator and the authentication server involved in the 802.11i protocol.

If a pre-shared key (PSK) is not pre-generated or cached, the supplicant and authentication server perform one of the mutual authentication protocols to generate the master session key (MSK) applied in the four-way handshake protocol.

If the MSK is transferred securely from authentication server to the authenticator, then four-way handshake protocol is executed between the supplicant and authenticator. Authenticator and supplicant firstly produce a secret key called the pair wise master key (PMK) based on MSK and then verify that the other party holds the same PMK in the handshaking. Finally both parties generate based on a pairwise transient key (PTK) used in the following session. PTK can also be produced based on the pre-shared key if the supplicant and supplicant are the same. No data is allowed to exchange before the handshake is finished successfully.

The four-way handshake does the four works: verifies PMK between the supplicant and authenticator; generates the temporal keys; authenticates the security parameters; introduces keying material to develop the group key handshake.

Alice-and-bob specification in simplified four-way handshake protocol is in Figure 7.

$$\boxed{\begin{array}{ll} message1: & Alice \to Bob: Anonce, m_1 \\ message2: & Bob \to Alice: Snonce, m_2, MIC_2 \\ message3: & Alice \to Bob: Anonce, m_3, MIC_3 \\ message4: & Bob \to Alice: m_4, MIC_4 \end{array}}$$

Figure 7. The simplified four-way handshake protocol

Where $Alice$ represents the authenticator; Bob represents supplicant; $Anonce$ and $Snonce$ are the random number generated by $Alice$ and Bob, respectively; $Anonce$ and $Snonce$ are used to generate the PTK that divided into three keys: Key Confirmation Key (KCK) is used by the EAPOL-key exchanges to implement data origin authenticity. Key Encryption Key (KEK) is used by the EAPOL-key exchanges to implement for confidentiality and Temporary Key (TK) is used by the data-confidentiality protocols. m_1, m_2, m_3, m_4 include the key replay counter $Replay\_Counter$. $MIC(\ )$ represents Message Integrity Code (MIC) function; $MIC_2 = MIC(Snonce, m_2)$, $MIC_3 = MIC(Anonce, m_3)$, $MIC_4 = MIC(m_4)$.

Four-way handshake protocol consists of four messages in Figure 7:

(1) Authenticator $Alice$ first sends message $message1$ to supplicant Bob. $message1$ includes random number $Anonce$ and some secret keying material.

(2) Supplicant Bob receives the message $message1$, then it checks key replay counter $Replay\_Counter$. If the result is true then it creates random number $Snonce$. He also applies the Pseudo Random Functions (PRF) to generate the PTK. PRF accepts $Anonce$, $Snonce$, MSK generated after authentication between supplicant and authentication server in IEEE 802.1X , the supplicant's MAC address and the authenticator's MAC address as its input.

Supplicant Bob produces $MIC_2 = MIC(Snonce, m_2)$, and then sends message $message2$ to authenticator $Alice$. The supplicant Bob also sends the security parameters used during the association and uses the KCK to verify the entire message.

(3) Authenticator $Alice$ receives message $message2$ and computes PTK. At the same time he also checks $MIC_2 = MIC\left(Snonce, m_2\right)$ in message $message2$ .If the result is true, then he generates message $message3$ and sends it to supplicant Bob , otherwise message $message2$ are discarded. Message $message3$ includes $MIC_3 = MIC\left(Anonce, m_3\right)$, $Anonce$ and other information. The entire message is made an authentication check, which allows the supplicant to verify that whether the information is valid or not.

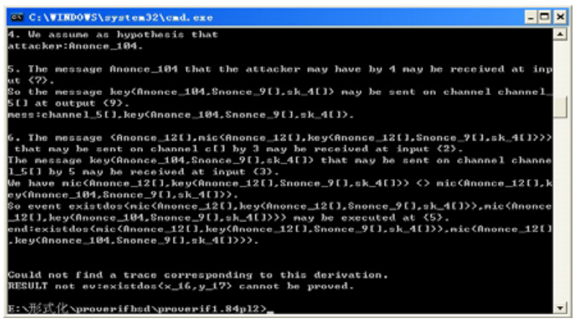
(4) Supplicant Bob receives message $message3$ and checks $MIC_3 = MIC\left(Anonce, m_3\right)$ .If the result is true then he generates message $message4$ and sends it to authenticator $Alice$ and configure PTK. Authenticator $Alice$ receives message $message4$ and checks $MIC_4 = MIC\left(m_4\right)$. If the result is true then he generates PTK. At this moment the temporal keys in the place are used by the data-confidentiality protocols.

Based on the proposed method of automatic proof of resistance of denial of service attacks, the extended applied pi calculus is used to formalize the model of $message1$ received by supplicant Bob . Then it is translated the input language of ProVerif. Due to the space limitation the results of analysis of ProVerif are given. ProVerif constructs two denials of service attacks: denial of service attack one in Fig. 9 and 10 and denial of service attack two in Fig.11 and 12. Denial of service attack two is detected by us with our proposed formal method.

According to the specification of four-way handshake protocol, the supplicant Bob verifies the message $message3$ based on PTK. The generation of PTK is based on $Anonce$ and $Snonce$. The generation of $Snonce$ is based on the verification of key replay counter $Replay\_Counter$ in message $message1$. So if the adversary fake or replay $Anonce$ or $Replay\_Counter$, then the verification of message $message3$ will be failed, hence the denial of service attack is launched.

Based on the formal model of message $message1$, the operations that supplicant Bob deal with message $message1$ and $message3$ are $Recv(Bob)\left\{\begin{matrix} act(Bob)\left[M_1, O_1^1, \ldots, O_i^1, \ldots, O_n^1\right], \\ act(Bob)\left[M_3, R_1^3, \ldots, R_i^3, \ldots, R_k^3\right] \end{matrix}\right\}$, where v is verification operation $MIC_3 = H\left\{KCK\right\}\left(Anonce^3, m_3\right)$. $KCK = \left\{Anonce^1, Snonce, MSK\right\}$, $M_1 = \left\{Anonce^1, m_1\right\}$, $M_3 = \left\{Anonce^3, m_3, MIC_3\right\}$ .So the set of association in message $message1$ and $message3$ is $\omega = \left\{Anonce^1\right\}$ .Owning to the $Anonce^1$ is not authenticated in four-way handshake protocol, we can get that four-way handshake protocol is not denial of service attack according to the definition 6. The codes of four-way handshake protocol in inputs of ProVerif are in Appendix B.

❁ Denial of service attack one in Figure 8 and 9. Figure 8 shows that the query event $existdos\left(x, y\right)$ can not be proved**.** ProVerif constructs denial of service attack one in Figure 9: in a run of the four-way handshake protocol, before the time supplicant Bob receives message $message3$ and after the time supplicant Bob message $message2$,the adversary impersonates authenticator $Alice$,then modifies the message $message1$ and constructs fake $Anonce'$ and $Replay\_Counter'$**,** and then generates message $message1$ and sends it to supplicant Bob . According to the specification on four-way handshake protocol supplicant Bob generates PTK' once more and update his cache, at the time supplicant Bob receives genius message $message3$, thus the verification of $MIC_3 = MIC\left(Anonce, m_3\right)$ with PTK', the result is false because PTK is different with the one in the authenticator. Thus this attack makes the PTK inconsistency. According to specification on four-way handshake protocol, authenticator $Alice$ will again send the message $message3$, but the verification of $MIC_3 = MIC\left(Anonce, m_3\right)$ is again false. After several times of this verification, authenticator $Alice$ and supplicant Bob need mutually authenticated again. Hence it is a denial of service attack.



```
4. We assume as hypothesis that
attacker:Anonce_104.

5. The message Anonce_104 that the attacker may have by 4 may be received at inp
ut <7>.
So the message key(Anonce_104,Snonce_9[],sk_4[]) may be sent on channel channel_
5[] at output <9>.
mess:channel_5[],key(Anonce_104,Snonce_9[],sk_4[]).

6. The message (Anonce_12[],mic(Anonce_12[],key(Anonce_12[],Snonce_9[],sk_4[])))
 that may be sent on channel c[] by 3 may be received at input <2>.
The message key(Anonce_104,Snonce_9[],sk_4[]) that may be sent on channel channe
l_5[] by 5 may be received at input <3>.
We have mic(Anonce_12[],key(Anonce_12[],Snonce_9[],sk_4[])) <> mic(Anonce_12[],k
ey(Anonce_104,Snonce_9[],sk_4[])).
So event existdos(mic(Anonce_12[],key(Anonce_12[],Snonce_9[],sk_4[])),mic(Anonce
_12[],key(Anonce_104,Snonce_9[],sk_4[]))) may be executed at <5>.
end:existdos(mic(Anonce_12[],key(Anonce_12[],Snonce_9[],sk_4[])),mic(Anonce_12[]
,key(Anonce_104,Snonce_9[],sk_4[]))).


Could not find a trace corresponding to this derivation.
RESULT not ev:existdos(x_16,y_17) cannot be proved.

E:\形式化\proverifbsd\proverif1.84pl2>_
```

Figure 8. The result of resistance of denial of service attack one in four-way handshakes protocol

$message1$: $Alice \rightarrow Bob: Anonce, m_1$

$message2$: $Bob \rightarrow Alice: Snonce, m_2, MIC_2$

$\boxed{message_1: adversary \rightarrow Bob: \left\{Anonce^1, m'_1\right\}}$

$message3$: $Alice \rightarrow Bob: Anonce, m_3, MIC_3$

$\boxed{message_3: Alice \rightarrow Bob: \left\{Anonce^3, m_3, MIC_3\right\}}$

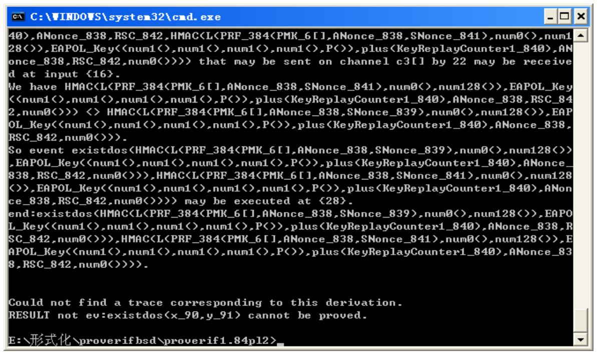
$message4$: $Bob \rightarrow Alice: m_4, MIC_4$

Figure 9. Denial of service attack one in four-way handshakes protocol

❆ Denial of service attack two in Figure 10 and 11. Figure 10 shows that the query event $existdos(x, y)$ can not be proved. ProVerif constructs denial of service attack one in Figure 11: in a run of the four-way handshake protocol, before the time supplicant Bob receives message $message3$ and after the time supplicant Bob message $message2$,the adversary impersonates authenticator $Alice$,then modifies the message $message1$ and constructs a fake $Replay\_Counter'$, and then generates message $message1$ and sends it to supplicant Bob . According to the specification on four-way handshake protocol supplicant Bob checks $Replay\_Counter'$, but does not check that if the $Anonce$ is replay or not, generates a new $Snonce$ and PTK' once more and update his cache, at the time supplicant Bob receives genius message $message3$, thus the verification of $MIC_3 = MIC(Anonce, m_3)$ with PTK', the result is false because PTK is different with the one in the authenticator. Thus this attack makes the PTK inconsistency. According to specification on four-way handshake protocol, authenticator $Alice$ will again send the message $message3$, but the verification of $MIC_3 = MIC(Anonce, m_3)$ is again false. After several times of this verification, authenticator $Alice$ and supplicant Bob need mutually authenticated again. Hence it is a denial of service attack.

For the sake of preventing denial of service attack one, He and Mitchell [31] argue that supplicant Bob stores the PTK and the TPTK (Temporary PTK) for each message $message1$. When it receives the message $message1$ it only updates TPTK. Only it receives the genius message $message3$ he updates PTK. But when the adversary sends many bogus messages $message1$, supplicant Bob need to store many TPTK, thus it makes a resource exhaustion of denial of service attack.

Based on the previous analysis, we can find the method that it makes the message $message1$ authenticated is to protect four-way handshake protocol against the two denial of service attacks. For example, sign messages $message1$ with private key of authenticator $Alice$, or encrypt the messages $message1$ with MSK, and so on. If people only protect it against denial of service two, then we need to let supplicant Bob to store the $Anonce$ in a time and check that whether $Anonce$ is replay or not before the next operation.



```
C:\WINDOWS\system32\cmd.exe
40),ANonce_838,RSC_842,HMAC(L(PRF_384(PMK_6[],ANonce_838,SNonce_841),num0(),num1
28()),EAPOL_Key((num1(),num1(),num1(),num1(),P()),plus(KeyReplayCounter1_840),AN
once_838,RSC_842,num0()))) that may be sent on channel c3[] by 22 may be receive
d at input {16}.
We have HMAC(L(PRF_384(PMK_6[],ANonce_838,SNonce_841),num0(),num128()),EAPOL_Key
((num1(),num1(),num1(),num1(),P()),plus(KeyReplayCounter1_840),ANonce_838,RSC_84
2,num0())) <> HMAC(L(PRF_384(PMK_6[],ANonce_838,SNonce_839),num0(),num128()),EAP
OL_Key((num1(),num1(),num1(),num1(),P()),plus(KeyReplayCounter1_840),ANonce_838,
RSC_842,num0())).
So event existdos(HMAC(L(PRF_384(PMK_6[],ANonce_838,SNonce_839),num0(),num128())
,EAPOL_Key((num1(),num1(),num1(),num1(),P()),plus(KeyReplayCounter1_840),ANonce_
838,RSC_842,num0())),HMAC(L(PRF_384(PMK_6[],ANonce_838,SNonce_841),num0(),num128
()),EAPOL_Key((num1(),num1(),num1(),num1(),P()),plus(KeyReplayCounter1_840),ANon
ce_838,RSC_842,num0()))) may be executed at {28}.
end:existdos(HMAC(L(PRF_384(PMK_6[],ANonce_838,SNonce_839),num0(),num128()),EAPO
L_Key((num1(),num1(),num1(),num1(),P()),plus(KeyReplayCounter1_840),ANonce_838,R
SC_842,num0())),HMAC(L(PRF_384(PMK_6[],ANonce_838,SNonce_841),num0(),num128()),E
APOL_Key((num1(),num1(),num1(),num1(),P()),plus(KeyReplayCounter1_840),ANonce_83
8,RSC_842,num0()))).


Could not find a trace corresponding to this derivation.
RESULT not ev:existdos(x_90,y_91) cannot be proved.

E:\形式化\proverifbsd\proverif1.84pl2>
```

Figure10. The result of resistance of denial of service attack two in four-way handshakes protocol

$message1$: $Alice \rightarrow Bob : Anonce, m_1$

$message2$: $Bob \rightarrow Alice : Snonce, m_2, MIC_2$

$\boxed{message_1 : adversary \rightarrow Bob : \{Anonce, m'_1\}}$

$message3$: $Alice \rightarrow Bob : Anonce, m_3, MIC_3$

$\boxed{message_3 : Alice \rightarrow Bob : \{Anonce, m_3, MIC_3\}}$

$message4$: $Bob \rightarrow Alice : m_4, MIC_4$

Figure 11. The new denial of service attack in four-way handshakes protocol

VIII. CONCLUSION

For the sake of preventing denial of service attacks, the opening stage is to give an analysis and proof of resistance of denial of service attacks in protocols, networks and distributed systems with formal method and enhance the confidence of people in its security. Many denial of service attacks are launched by protocol state. So far from Yu-Gligor model and Meadows model, resistance of denial of service attacks is formalized from the angle of protocol state. we further present for the first time a computed aided method of resistance of denial of service attacks based on event. We analyze the Resistance of denial of service attacks in JFK protocol [11] and IEEE 802.11 i four-way handshake protocol with ProVerif by our formal method. The results are that JFK protocol is resistance of denial of service attacks and IEEE 802.11 i four-way handshake protocol is not resistance of denial of service attacks. And simultaneously, a new denial of service attack in IEEE 802.11 i four-way handshake protocol is found by us, and the methods to prevent such denial of service attacks in IEEE 802.11 i four-way handshake protocol are proposed.

As future work, we plan to prove resistance of denial of service attacks in internet voting protocols. It would also be interesting to formalize resistance of denial of service attacks in computational model with CryptoVerif.

APPENDIX A RESPONDER BOB' FORMAL MODEL OF MESSAGE 1 IN JFKR PROTOCOL

```
(* JFKr *)
param redundantHypElim = beginOnly.
(* Exponential and Diffie-Hellman *)
data g/0.
fun exp/2.
equation exp(exp(g,y),z) = exp(exp(g,z),y).
(* Signature *)
fun S/2.
fun Pk/1.
data true/0.
fun V/3.
fun RecoverKey/1.
fun RecoverText/1.
equation V(S(k,v), Pk(k),v) = true.
equation RecoverKey(S(k,v)) = Pk(k). (* For the attacker *)
equation RecoverText(S(k,v)) = v.    (* For the attacker *)
(* Shared-key encryption *)
fun E/2.
fun D/2.
equation D(k,E(k,v)) = v.
(* Keyed hash function *)
fun H/2.
(* Sets *)
data consset/2.
data emptyset/0.
pred member/2.
clauses
member:x,consset(x,y);
member:x,y -> member:x,consset(z,y).
(* Tags *)
data tagE/0. data tagA/0. data tagV/0.
(* Constructors for JFK's formatted messages
Selectors are implicit when using "data" *)
data cons1/2. data cons2/5. data cons3/7. data cons4/2.
(* More constants *)
data constI/0. data constR/0. data saR/0.
(* Free names *)
private free dos,c2,c3,c4.
free c,c1. (* Public channel *)
free pub, getprinc, getexponential, grpinfoR,
channelSIAR1, channelSIAR2.
(* Denial of service for I. *)
query ev:existdos(x,y).
(*Initiator The process processI corresponds to I^A  in the figure.*)
let processI =
    !in(exponent, (dI, xI));
    !in(init, (IDRp, saI));    (* Init message *)
    new NI;
    out(c1, cons1(NI, xI));
    in(c2, cons2(=NI, NR, xR, grpinfoR, tR));
    let h = exp(xR, dI) in
    let Ka = H(h, (NI, NR, tagA)) in
    let Ke = H(h, (NI, NR, tagE)) in
    let Kv = H(h, (NI, NR, tagV)) in
    let sI = S(kAminus, (NI, NR, xI, xR, grpinfoR)) in
    let eI = E(Ke, (IDA, IDRp, saI, sI)) in
    let hI = H(Ka, (constI, eI)) in
    out(c3, cons3(NI, NR, xI, xR, tR, eI, hI));
    in(c4, cons4(eR, hR));
    if H(Ka, (constR, eR)) = hR then
    let (IDRl, saR, sR) = D(Ke, eR) in
    if V(sR, IDRl, (NI, NR, xI, xR)) = true then
    out(connect, (IDRl, IDRp, saI, saR, Kv)).
(*Responder The process processR corresponds to R^A  in the figure.*)
let processR =
     (
       (  !in(exponent,(dR,xR));(*R^A_1*)
          !in(c1, cons1(NI, xI));    new NR;
          let tR = H(KR, (xR, NR, NI)) in
          out(c2, cons2(NI, NR, xR, grpinfoR, tR))
       )
      |(*R^A_3*)
        new f;
       (
         out(f, emptyset)
       |
       (
          ! in(c3, cons3(NI,NR,xI,xR,tR,eI,hI));
    if tR = H(KR, (xR, NR, NI)) then
          (
          in(f, cache);
          (
            out(f, consset(tR, cache))
        |
           if member:tR,cache then 0 else
            new l;
            (
       (
             !
             in(exponent, (dR, =xR));
             out(l, dR)
             )
           |
             (
             in(l, dR);
             processR4
             )
            )
          )
          )
          else event existdos(tR, H(KR, (xR, NR, NI)))
        )
      )
    ).
let processR4 =
     let h = exp(xI,dR) in
     let Ka = H(h, (NI, NR, tagA)) in
     let Ke = H(h, (NI, NR, tagE)) in
     let Kv = H(h, (NI, NR, tagV)) in
     if H(Ka, (constI, eI)) = hI then
     let (IDIl, IDRp, saI, sI) = D(Ke,eI) in
     if V(sI, IDIl, (NI, NR, xI, xR, grpinfoR)) = true then
     out(accept, (IDIl, IDRp, saI, saR, Kv));
     (
     (
      let sR = S(kAminus, (NI, NR, xI, xR)) in
      let eR = E(Ke, (IDA, saR, sR)) in
      let hR = H(Ka, (constR, eR)) in
      out(c4, cons4(eR, hR));
      0
      )
     ).
(* Whole JFK system. *)
(* Standard version of the process *)
process
     new exponent;
     new KR;
(* private channel used to simulate the set C of honest principals *)
     new honestC;
     ( ! new d; let x = exp(g,d) in out(getexponential, x);
     ! out(exponent, (d,x)) )
     |
     !  new kAminus;
     let IDA = Pk(kAminus) in
     new connect; new accept; new init; new channelSIA;
     out(getprinc, (IDA, init, accept, connect, channelSIA));
     in(channelSIA, SIA);
     (
      (! out(honestC, IDA) )   (* IDA is in C *)
      | processI  | processR  )
```

APPENDIX B SUPPLICANT BOB' FORMAL MODEL OF MESSAGE 1 IN FOUR-WAY HANDSHAKES PROTOCOL

```
(*4-Way Handshake*)
(* the EAPOL-Key frame *)
data EAPOL_Key/5.
data num0/0.       (* the integer 0 *)
data num1/0.
data num128/0.   (* the integer 128 *)
data num256/0.     (* the integer 256 *)
data P/0.              (* the symbol for Pairwise *)
data cons/5.       (* Sets *)
(* Pseudo-random function producing 384 bits of output *)
fun PRF_384/3.
(* L(Str, F, L) From Str starting from the left,
extract bits F through F+L    1 *)
fun L/3.
(* Keyed-Hashing for Message Authentication *)
fun HMAC/2.
fun plus/1.             (* The function for add one *)
(* Shared-key encryption *)
fun enc/2.
fun dec/2.
equation dec(enc(x,y),y)=x.
(* channel *)
free c,c1.  (* Public channel *)
private free c2,c3,c4.  (* Private channel *)
private free S.
(* Denial of service for Supplicant. *)
query ev:existdos(x,y).
(* the process  authenticator corresponds to Initiator *)
let authenticator=
    (* Init message *)
    new ANonce;
    let KeyNonce1=ANonce in
    let KeyInfo1=cons(num0,num0,num1,num0,P) in
    new KeyReplayCounter1;
    let mess1=EAPOL_Key(KeyInfo1,KeyReplayCounter1,
KeyNonce1,num0,num0) in
    out(c1,mess1);   (* send the Message 1 *)
    in(c2,mess2);      (* receive the Message 2 *)
    let EAPOL_Key(KeyInfo2,KeyReplayCounter2,KeyNonce2,
KeyRSC2,KeyMIC2)=mess2 in
         if KeyReplayCounter1=KeyReplayCounter2 then
         let (S2,M2,A2,I2,K2)=KeyInfo2 in
         if K2=P then
         if M2=num1 then
         let PTK=PRF_384(PMK,ANonce,KeyNonce2) in
         let KCK=L(PTK,num0,num128) in
         let KEK=L(PTK,num128,num128) in
         let TK=L(PTK,num256,num128)    in
         let EAPOL2=EAPOL_Key(KeyInfo2,KeyReplayCounter2,
KeyNonce2,KeyRSC2,num0) in
         let MIC2=HMAC(KCK, EAPOL2) in
         if MIC2=KeyMIC2 then
         let KeyInfo3=(num1,num1,num1,num1,P) in
         let KeyReplayCounter3=plus(KeyReplayCounter2) in
         new RSC;
         let EAPOL3=EAPOL_Key(KeyInfo3,KeyReplayCounter3,
KeyNonce1,RSC,num0) in
         let MIC3=HMAC(KCK, EAPOL3) in
         let mess3=EAPOL_Key(KeyInfo3,KeyReplayCounter3,
KeyNonce1,RSC,MIC3) in
(* send the Message 3 *)
         out(c3,mess3);
in(c4,mess4);             (* receive the Message 4 *)
let EAPOL_Key(KeyInfo4,KeyReplayCounter4,KeyNonce4,
KeyRSC4,KeyMIC4)=mess4 in
if KeyReplayCounter3=KeyReplayCounter4 then
     let (S4,M4,A4,I4,K4)=KeyInfo4 in
     if S4=num1 then
     if M4=num1 then
     if A4=num0 then
     if I4=num0 then
     if K4=P then
     let EAPOL4=EAPOL_Key(KeyInfo4,KeyReplayCounter4,
KeyNonce4,KeyRSC4,num0) in
     let MIC4=HMAC(KCK, EAPOL4) in
     if MIC4=KeyMIC4 then  0.
(* the process  Supplicant corresponds to Responder *)
(* receive the Message 1 *)
let Supplicant=in(c1,mess1);
     let EAPOL_Key(KeyInfo1,KeyReplayCounter1,
KeyNonce1,KeyRSC1,KeyMIC1)=mess1 in
     let (S1,M1,A1,I1,K1)=KeyInfo1 in
     if A1=num1 then
     if K1=P then
     new SNonce;
     let PTK=PRF_384(PMK,KeyNonce1,SNonce) in
     let KCK=L(PTK,num0,num128) in
     let KEK=L(PTK,num128,num128) in
     let TK=L(PTK,num256,num128)    in
     let KeyInfo2=(num0,num1,num0,num0,P) in
     let EAPOL2=EAPOL_Key(KeyInfo2,KeyReplayCounter1,
SNonce,KeyRSC1,num0) in
     let MIC2=HMAC(KCK, EAPOL2) in
     let mess2=EAPOL_Key(KeyInfo2,KeyReplayCounter1,
SNonce,KeyRSC1,MIC2) in
     out(c2,mess2);          (* send the Message 2 *)
     in(c3,mess3);             (* receive the Message 3 *)
     let EAPOL_Key(KeyInfo3,KeyReplayCounter3,KeyNonce3,
KeyRSC3,KeyMIC3)=mess3 in
     if KeyNonce3=KeyNonce1 then
     let (S3,M3,A3,I3,K3)=KeyInfo3 in
     if S3=num1 then
     if M3=num1 then
     if A3=num1 then
     if I3=num1 then
     if K3=P then
     let EAPOL3=EAPOL_Key(KeyInfo3,KeyReplayCounter3,
KeyNonce3,KeyRSC3,num0) in
     let MIC3=HMAC(KCK, EAPOL3) in
     (* Verify the Message 3 *)
     if MIC3=KeyMIC3 then
     (
          let KeyInfo4=(num1,num1,num0,num0,P) in
          let EAPOL4=EAPOL_Key(KeyInfo4,KeyReplayCounter3,
num0,KeyRSC3,num0) in
          let MIC4=HMAC(KCK, EAPOL4) in
          let mess4=EAPOL_Key(KeyInfo4,KeyReplayCounter3,
num0,KeyRSC3,MIC4) in
          out(c4,mess4)               (* send the Message 4*)
            )
        else
 event existdos(MIC3,KeyMIC3).(* Denial of service for Message 1 *)
 (* Whole 4-Way Handshake system. *)
process new PMK;((!authenticator)|(!Supplicant))
```

ACKNOWLEDGMENT

This work was supported in part by a grant from: the Special Fund for Basic Scientific Research of Central Colleges, South-Central University for Nationalities: CZY12017; Natural Science Foundation of South-Center University for Nationalities: YZZ09008; Technology Innovation Foundation of SME in Wuhan: SZY11008; Natural Science Foundation of The State Ethnic Affairs Commission of PRC under the grants No: 10ZN09.

REFERENCES

[1] C.F.Yu, V.D.Gligor, "A Formal Specification and Verification Method for the Prevention of Denial of Service," IEEE Transactions on Software Engineering, vol.16, No.6, pp.581-592, 1990.

[2] C.Meadows, "A cost-based framework for analysis of denial of service networks," Journal of Computer Security, vol.9,No.1/2,pp.143-164,2001.

[3] S.Tritilanunt , C.Boyd , E.Foo, J.Manuel, and G.Nieto, " Cost-based and time-based analysis of DoS-resistance in HIP," In Proceedings of the thirtieth Australasian conference on Computer science, Darlinghurst, Australia, Australia, pp.191-200,2007.

[4] S.Tritilanunt, "Protocol engineering for protection against denial of service attacks," Doctor Thesis, Brisbane Australia .Queensland University of Technology.2009.

[5] B.Blanchet, "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules," In Proceedings of the14th IEEE Computer Security Foundations Workshop, Cape Breton, Nova Scotia, Canada, pp.82-96,2001.

[6] M.Abadi, B.Blanchet, C.Fournet, "Just Fast Keying in the Pi Calculus," ACM Transactions on Information and System Security, vol.10, No.3, pp.1-59, 2007.

[7] M.Backes, C.Hritcu, M.Maffei, "Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-Calculus," In Proceedings of the 2008 21st IEEE Computer Security Foundations Symposium, Carnegie Mellon University, Pittsburgh, USA, pp.195-209,2008.

[8] B.Meng, W.Huang, J.Qin, "Automatic Verification of Security Properties of Remote Internet Voting Protocol in Symbolic Model,"Information Technology Journal , vol.9,No.8,pp.1521-1556,2010.

[9] B.Meng, W.Huang, Z.M.Li, D.J.Wang, "Automatic Verification of Security Properties in Remote Internet Voting Protocol with Applied Pi Calculus," International Journal of Digital Content Technology and its Applications, vol. 4, No. 7, pp.88-107, 2010.

[10] B.Meng, "Refinement of mechanized proof of security properties of remote internet voting protocol in applied PI calculus with ProVerif," Information Technology Journal, vol.10, No.2, pp.293-334, 2011.

[11] W.Aiello, S.M.Bellovin, M.Blaze, R.Canetti, J.Ioannidis, A. D.Keromytis , O.Reingold, "Just fast keying: Key agreement in a hostile internet,".ACM Transactions on Information and System Security, Vol.7,No.2,pp.242-273,2004.

[12] V.D.Gligor, "A note on the denial-of-service problem," In Proceedings of IEEE Symposium on Security and Privacy, Washington, DC, USA, pp.139-149, 1983.

[13] V.D. Gligor, "A note on denial-of-service in operating systems,"IEEE Transactions on Software Engineering. vol.10, No.3, pp.320-324,1984.

[14] E.Bacic, M.Kuchta, "Considerations in the Preparation of a Set of Availability Criteria," In Proceedings of the Third Annual Canadian Computer Security Symposium, Ottawa, Canada, 283-292,1991.

[15] J.K.Millen, "A Resource Allocation Model for Denial of Service Protection," Journal of Computer Security, vol.2, No.2-3, pp. 89-106, 1993.

[16] M.Abadi, L.Lamport, "Composing specifications,".ACM Transactions on Programming Languages and Systems, vol.15, No.1, pp.73-132, 1993.

[17] V.Ramachandran, "Analyzing DoS-resistance of protocols using a cost-based framework," Technical Report DCS/TR-1239, Yale University.2002.

[18] J.Smith, J. M.Gonzalez-Nieto , C.Boyd, "Modeling denial of service attacks on JFK with Meadows's cost-based framework," In Proceedings of the 2006 Australasian workshops on Grid computing and e-research, Darlinghurst, Australia, Australia, 125-134,2006.

[19] S.Lafrance, J.Mullins, "Using Admissible Interference to Detect Denial of Service Vulnerabilities," In Proceedings of the Sixth International Workshop in Formal Methods, Paris.2003.

[20] S.Zhou., R.Jiang,X.Yang, "DoS attacks on security protocols of the formal analysis," Journal of Research Institute of China Electronics, vol.3,No.6, pp.592-598, 2008.

[21] B.Groza, M.Minea, "Formal Modelling and Automatic Detection of Resource Exhaustion Attacks," In Proceedings of 6th ACM Symposium on Information, Computer and Communications Security, Hong Kong, China, pp.326-333, 2011.

[22] E.Amoroso, "A policy model for denial of service," In Proceedings of Third IEEE Computer Security Foundations Workshop, Franconia, New Hampshire, USA,pp.110-114,1990.

[23] F.Cuppens, C.Saurel, "Towards a formalization of availability and denial of service," In Proceedings of Information Systems Technology Panel Symposium on Protecting Nato Information Systems in the 21st Century.1999.

[24] A.Gabillon, L.Gallon, "An Availability Model for Avionic Data Buses," In Proceedings of First International Workshop on Issues in Security and Petri Nets, Eindhoven, The Netherlands.2003.

[25] F.Cuppens, N.Cuppens-Boulahia,T.Ramard, "Availability Enforcement by Obligations and Aspects Identification,"In Proceedings of the First International Conference on Availability, Reliability and Security, Vienna University of Technology, Austria,pp.229-239,2006.

[26] G.Agha, M.Greenwald, C.A.Gunter, S.Khanna, J.Meseguer, K.Sen,P.Thati, "Formal Modeling and Analysis of DoS Using Probabilistic Rewrite Theories," In Proceedings of International Workshop on Foundations of Computer Security, Chicago, IL, pp.91-102, 2005.

[27] A.Mahimkar, V.Shmatikov, "Game-Based Analysis of Denial-of-Service Prevention Protocols," In Proceedings of the 18th IEEE workshop on Computer Security Foundations, Aix-en-Provence, France, pp.287-301, 2005.

[28] B.Meng, W.Huang, D.Wang, F.Shao, "Automatic proof of resistance of denial of service attacks in protocols," Journal on communications, vol.33, No.3, pp.112-121, 2012.

[29] M.Abadi, C.Fournet, "Mobile values, new names and secure communication,"In Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, United Kngdm, pp.104-115,2001.

[30] M.Abadi, B.Blanchet, "Analyzing security protocols with secrecy types and logic programs," Journal of the ACM, vol.52, No.1, pp.102-146, 2005.

[31] C.He, J. C. Mitchell, “Analysis of the 802.11i 4-Way Handshake,” In Proceedings of the Third ACM International Workshop on Wireless Security, Philadelphia, pp.43-50,2004.
[32] B.Meng, W.Wang, W. Chen. “Verification of Resistance of Denial of Service Attacks in Extended Applied Pi Calculus with ProVerif,” Journal of Computers, Vol 7, No 4, pp.890-899, 2012.
[33] L. Jiang, H. Chen, F. Deng, Q. Zhong, “A Security Evaluation Method Based on Threat Classification for Web Service,”Journal of Software, Vol 6, No 4, pp.595-603, 2011.
[34] B. Li, L.Zhao, J. Zhu, J. Wu, “A Policy-based Adaptive Web Services Security Framework,”Journal of Software, Vol 6, No 12 , pp.2456-2463, 2011.
[35] L.L.Jilani, I. Derbel, K.Bsaies, H.Nasreddine, A. Mili, “Reasoning About Quantitative Architectural Attributes,” Journal of Software, Vol 6, No 4, pp.574-583, 2011.

Bo Meng was born in 1974 in China. He received his M.S. degree in computer science and technology, Ph.D. degree in traffic information engineering and control from Wuhan University of Technology, at Wuhan, China, in 2000, 2003, respectively. From 2004 to 2006, he works in Wuhan University, China as Postdoctoral researcher in information security.

Currently he is an Associate Professor in school of computer, South-Center University for Nationalities, China. He has authored/coauthored over 50 papers in International/National journals and conferences. His current research interests include electronic commerce, Internet voting, and protocol security.

Wei Huang was born in 1983 in China. He received his BA degree in computer science and technology in 2007 from Donghu College of Wuhan University. at Wuhan in China.

Currently he is a postgraduate student in college of computer science, South-Central University For Nationalities, at Wuhan, China. His current research interests include information security, internet technology.

Zimao Li was born in 1974. He received his B.S. degree in Mathematics in 1996, M.Eng degree in Computer Science in 1999, both from Shandong University, P.R.China, and Ph.D degree in Computer Science from City University of Hong Kong, Hong Kong, in 2002.

He is currently an Associate Professor in the School of Computer Science, South-Center University for Nationalities, P.R. China. His research interests are design and analysis of algorithms, complexity theory and computational biology.