Verification of Security Protocols Using ProVerif

Prof Jayana C. Kaneriya, Prof Jalpa Khamar, Prof Avani Dadhania
Assistant Professor, Assistant Professor, Assistant Professor
Computer Engineering, LDRP-ITR,Gandhinagar, India

Abstract: **With the rise of the proliferation of the Internet and other open networks in the day-to-day life, the use of the security protocols therein has also been on rise and so is the need for devising newer security protocols. The security protocols are meant to provide secure communication. Before being deployed, the security protocols are required to be thoroughly tested to gain confidence in the same. However, as is known from the basic principles of Software Engineering, exhaustive testing is virtually impossible.**

1. Introduction

To secure Internet applications security protocols are developed. Security protocols are meant to provide secure communication. Communication channels may be exposed to attacks who may try to breach the security of the system by observing, modifying, delaying, redirecting or replaying the messages. So before being deployed, the security protocols are required to be verified. The formal verification of security protocols can state protocol is formally correct or not and thus free of errors such as inconsistency or ambiguity. For formal verification, we use ProVerif tool because it is a complete automated tool and can explore the complete state space of security protocols.

2. THEORETICAL BACKGROUND

2.1 Security Protocols

A security protocol is an abstract or concrete protocol that performs a security related function and applies cryptographic methods. Security protocols or cryptographic protocols are small distributed programs that ensure security properties in a hostile environment. Examples of properties that need to be ensured are secrecy and authentication [2].

2.2 Goals Of Security Protocols

A security protocol is designed to provide one or more security related services. These services can be confidentiality, authentication, integrity, access control and nonrepudiation.

2.3 Attacks On Security Protocols

The goal of an attacking a cryptosystem is to recover the key in use rather than simply to recover the plaintext of a single cipher text. There are two general approaches to attacking symmetric encryption scheme. One is cryptanalysis and other is brute force attack.

Other attacks are dictionary attacks, Denial of service attack [3], Virus, impersonation attack, release of message attack, traffic analysis, masquerade attack etc.

2.4 Formal Methods To Verify Security Protocols

Formal verification conducts exhaustive exploration of all possible behaviours. There are roughly two approaches to formal verification. These are theorem proving and model checking. Theorem Proving consists of using a formal version of mathematical reasoning about the system. Examples of theorem proving tools are HOL theorem prover, the ACL2 theorem prover, Isabelle theorem prover etc. Model checking consists of a systematically exhaustive exploration of the mathematical model.

3 . RELATED WORK

Model checking tools have advantages over theorem proving tools. There are different model checking tools i.e. Casper/FDR[5], NRL, Athena[4], LySatool[6], ProVerif[7], Avispa(OFMC)[8]. Comparison of some model checking tools is given in following table.

Tool Name	Publicly available	Falsification	Verification		Termination
			bounded	unbounded	
Casper /FDR	Yes	Yes	Yes	No	Yes
NRL	No	No	No	Yes	Yes
Athena(bounded)	No	Yes	Yes	No	Yes
Athena(unbounded)	No	Yes	No	Yes	No
LySatool	Yes	No	No	Yes	Yes
ProVerif	Yes	Yes	No	Yes	Yes
OFMC	Yes	Yes	Yes	No	Yes

4. Implementation and Analysis

4.1 Andrew Secure RPC Protocol

Using out and in commands sender process and receiver process communicate with each other. Using encryption function sender or receiver can encrypt any message. This function takes two arguments. One is plain text message to be encrypted and other is key for encryption i.e. encrypt (M, Kab). Using decryption function sender or receiver can decrypt any message. This function takes two arguments. One is message to be decrypted and other is key for decryption i.e. decrypt (M, Kab). Using let command different values of messages are compared with stored values i.e. let (a, b) = decrypt (M, Kab). In this message M contains two values and first value is stored in variable a and other value is compared with b.

4.2. Kao Chow Authentication Protocol

In this protocol sender and receiver process is same as RPC protocol. This protocol uses server for authentication. Server Process binds host name with key using let command i.e. let Kas = getkey (a) in and then server sends message to other process.

4.3 An New Symmetric Key Distribution Protocol Using Centralized Approach

Functions for calculation are defined below. Inv function is used to inverse the number which is calculated by cal function.

```
fun cal/1.
fun inv/1.
equation inv(cal(x))= x.
```

Sender Process and Receiver Process does same thing as previous protocols. Server Process binds host name with key using let command i.e. let Kas = getkey (a) in and then server sends message to other process. Here server process sends data to other processes simultaneously. This parallel execution of processes is done by | operator and this operator is placed between two processes i.e. out(C, M1) | out(C, M2).

4.4 An On Demand Key Establishment Protocol for MANETs

In our model, ProVerif uses following functions and equations for calculating and solving messages.

```
fun hash/1.
fun exp/2.
data g/0.
fun key/3.
equation exp(exp(g,y),z) = exp(exp(g,z),y).
```

ProVerif model for this protocol uses hash function for authentication. The function exp is used for exponential calculations. The function key is used to calculate final shared key. Sender and receiver process modelling is same as previous protocols.

4.5 An Authentication Protocol for Exchanging Encrypted Messages via an Authentication Server

This protocol is used for only authentication. Using out and in commands sender process and receiver process communicate with each other. Using encryption function sender or receiver can encrypt any message. This function takes two arguments. One is plain text message to be encrypted and other is publickey for encryptions i.e. encrypt (M, PubK). Using decryption function sender or receiver can decrypt any message. This function takes two arguments. One is message to be decrypted and other is privatekey for decryptions i.e. decrypt (M, PriK). Using let command different values of messages are compared with stored values i.e. let (a, =b) = decrypt(M, PriK). In this message M contains two values and first value is stored in variable a and other value is compared with b. Main process creates private public key pair for sender, receiver and server. The public keys of each of the components are distributed on public channels i.e. let publickey = pk (privatekey) in.

4.6 TinyPK: Securing Sensor Networks with Public Key Technology

Sender and receiver process does encryption and decryption using functions. To calculate checksum of any message it uses checksum function. Declaration is given below.

```
fun checksum/1.
```

5. Modelling Attacks in ProVerif

ProVerif uses Dolev –Yao model of an attacker. The active attacker can eavesdrop or intercept any message sent over the network and send them to protocol participants.

5.1 Reply attack

ProVerif can send any message to any participants which are sent in the current session. In replay attack message stored in the previous session is sent to participants in current session. So we model replay attack by which a process send previously stored messages to the participants.When we consider Andrew Secure RPC Protocol, the last message contains session key and nonce. These values are newly generated by receiver. So if any attacker has

these values from previous session then it can compromise the sender to use stale key. To implement replay attack one new process is added with sender and receiver.

```
let ProcessForReplayAttack =
        ! in(c,m);
        ! out(c,m).
```

We use correspondence assertion to prove Replay Attack. ProVerif implementation is shown below.

```
query ev:endA(x,y) ==> ev:beginA(x,y).
let processA =
        ….
        in(c,m4);
        let(kabnew,NB1)=decrypt(m4,kab) in
        event endA(kabnew,NB1);
        out(c,encrypt(SecretA,kabnew)).
 let processB =
        ….
        new NbNew; new kabnew;
        event beginA(kabnew,NbNew);
        out(c,encrypt((kabnew,NbNew),kab)).
```

5.2. Denning Sacco Attack

This attack is somewhat same as replay attack. But in replay attack, attacker cannot impersonate the legitimate participants. In Denning Sacco attack, attacker can fool the legitimate participants, means here authentication property is compromised. We model this attack in Kao Chow Authentication protocol. To implement denning sacco attack one new process is added with sender and receiver.

```
let ProcessForDenningSaccoAttack =
        ! in(c, (m1,m2));
        ! in(c,(m3,m4,nb));
        ! out(c, encrypt(nb, kab)).
```

We use correspondence assertion to prove Denning Sacco Attack. ProVerif implementation is shown below.

```
query ev:endeventforattack(x,y) ==> ev:begineventforattack(x,y).
query evinj:endeventforattack(x,y) ==> evinj:begineventforattack(x,y).
let processA =
        ….
        if na = decrypt(m4, kab) then
        out (c, encrypt(nb, kab));
        event begineventforattack(Na, nb);
        ….
let processB =
        ….
        out (c, (m1,encrypt (na, kab ), nb) );
        in (c, (m5 ) );
        if nb = decrypt(m5, kab) then
        event endeventforattack(na, nb);




        ….
```

6. METRICS FOR EVALUATION

6.1 Time

Application running time is one of the important factors. To measure time taken by our ProVerif model we design function Calculate_time in Java. This function uses system time to calculate time taken to verify different properties of security protocols. Following Table gives rules generated by ProVerif and its total running time to verify all queries for the protocol.

Security Protocol	Verification	
	Rules	Time(msec)
Andrew Secure RPC Protocol	39	11
Kao Chow Authentication Protocol	81	15
An New Symmetric Key Distribution Protocol	207	17
An On Demand Key Establishment Protocol for MANETs	196	16
TinyPK	60	11
An Authentication Protocol with an Authentication Server	244	32

6.2 Verification Result

To validate our approach experimentally, we implemented a series of cryptographic protocols and verified their security against demanding threat models. Following figures summarize our results for these protocols.

Following table concerns verification; it gives the number of queries and the kinds of security properties they express. A secrecy query requires that a password or key be protected. An authentication query requires that message exchange be authentic. Some queries can be verified even in the presence of attackers that control some corrupted principals, thereby getting access to their keys and passwords. Not all queries hold for all protocols; in fact some queries are designed to test the boundaries of the attacker model and are meant to fail during verification.

Security Protocol	Security Goals			
	Queries	Secrecy	Authentication	Attacker
Andrew Secure RPC Protocol	3	No	_	Yes
Kao Chow Authentication Protocol	4	_	No	Yes
An New Symmetric Key Protocol	5	Yes	Yes	No
On Demand Key Establishment Protocol	5	Yes	Yes	No
TinyPK	6	Yes	Yes	No
An Authentication Protocol with an Authentication Server	4	Yes	Yes	No

7. Conclusion

Applying formal methods to security protocols is an interesting, challenging, and current research area. There have been many significant verification tools for security protocols developed over the last twenty years. But much work needs to be done in the area of formal methods for specifying security properties, logics for reasoning about protocols. Security protocols used in different networks i.e. internet, wireless network, and mobile ad hoc networks are verified in this thesis. ProVerif is fully automated and efficient tool to verify security protocols. We verify security properties i.e. confidentiality, authentication in the

applied pi calculus using ProVerif tool. We model replay attack and denning sacco attack using ProVerif tool.

REFERENCES

[1] Juan Carlos Lopez Pimentel, Raul Monroy, Formal Support to Security Protocol Development: A Survey, 2008.

[2] A. Gordon, A. Jeffrey, Authenticity by typing for security protocols, Jounral of Computer Security, 2003

[3] Kevin J. Houle, George M. Weaver, Trends in Denial of Service Attack,Oct-2001.

[4] Chul-Wuk Jeon, Il-Gon Kim, Jin-Young Choi, Automatic Generation of the C# Code for Security Protocols Verified with Casper/FDR, 2005.

[5] Dawn Song , Sergey Berezin ,Adrian Perrig, Athena: A Novel Approach to Efficient Automatic Security Protocol Analysis,2000

[6] Mikael Buchholtz, User's Guide for the LySatool, April 18 2005

[7] Bruno Blanchet ProVerif Automatic Cryptographic Protocol Verifier User Manual, May 18, 2008

[8] AVISPA v1.1 User Manual, IST-2001-39252,June 30 2006

[9] R. H. Rahman, N. Nowsheen, M. A. Khan, An New Symmetric Key Distribution Protocol Using Centralized Approach, Asian Journal of Information Technology, 2007.

[10] Mounis Khatib, Khaled Masmoudi, Hossam Afifi, An on demand key establishment protocol for MANETs, In Proc. International conference on Advanced Information Networking and Applications, IEEE, 2006

[11] Ronald Watro, Derrick Kong, TinyPK: Securing Sensor Networks with Public Key Technology, 2004.