05 Jul 2005

Proving Secure Properties of Cryptographic Protocols with Knowledge Based Approach

Xiaochun Cheng

Xiaoqi Ma

Maggie Xiaoyan Cheng
Missouri University of Science and Technology, chengm@mst.edu

Scott C.-H. Huang

Recommended Citation

X. Cheng et al., "Proving Secure Properties of Cryptographic Protocols with Knowledge Based Approach," *Proceedings of the IEEE International Performance, Computing, and Communications Conference, 2005*, Institute of Electrical and Electronics Engineers (IEEE), Jul 2005.
The definitive version is available at https://doi.org/10.1109/PCCC.2005.1460503

Proving Secure Properties of Cryptographic Protocols with Knowledge Based Approach

Xiaochun Cheng
Dept of Computer Science
University of Reading
Reading, England
x.cheng@rdg.ac.uk

Xiaoqi Ma
Dept of Computer Science
University of Reading
Reading, England
xiaoqi.ma@rdg.ac.uk

Maggie Cheng
Dept of Computer Science
University of Missouri
Rolla, MO
chengm@umr.edu

Scott C.-H. Huang
Dept of Computer Science
Univ. of Minnesota
Minneapolis, MN
huang@cs.umn.edu

Abstract

Cryptographic protocols have been widely used to protect communications over insecure network environments. Existing cryptographic protocols usually contain flaws. To analyze these protocols and find potential flaws in them, the secure properties of them need be studied in depth. This paper attempts to provide a new framework to analyze and prove the secure properties in these protocols. A number of predicates and action functions are used to model the network communication environment. Domain rules are given to describe the transitions of principals' knowledge and belief states. An example of public key authentication protocols has been studied and analysed.

1 Introduction

Cryptographic algorithms have been widely used to protect security of information going through networks, especially insecure networks environment, of which the Internet is the best example. Although there are a large number of "good" cryptographic algorithms that are very difficult to break, mere cryptography sometimes cannot completely protect anything if it is not used in an appropriate way. To give guidelines to use cryptographic technology, a large number of security protocols have been used. These protocols do contribute greatly to information security.

Unfortunately, many cryptographic protocols are inherently incorrect, or at least cannot achieve the stated aim. Some protocols were considered perfect at first, and were found to contain flaws many years later [5, 9]. To evaluate and verify security protocols effectively, people always try to analyse the secure properties of these protocols.

In this paper, we propose a framework to analyse the secure properties of cryptographic protocols. Our framework is a knowledge-based logic inference system.

This paper first gives a summary of related work on cryptographic protocol analysis, and then gives a detailed description of our knowledge based framework. Thereafter, we take the famous Needham-Schroeder public key authentication protocol [9] as an example to prove two important authentication properties. The last section concludes the paper.

2 Related Work

Since Burrows *et al.* proposed their famous BAN logic [3], designing formal methods for analyzing cryptographic protocols has been a research highlight in recent years. Much research work has been conducted in this area and many good models and methods have been proposed. Generally speaking, all these methods can be broadly divided into two categories, namely state based methods and rule based methods.

State based methods model security protocols as finite state machines. They search the state space

0-7803-8991-3/05/$20.00 © 2005 IEEE

exhaustively to see whether all the reachable states are safe [10]. If some reachable state in a security protocol is proved to be unsafe, a flaw may be reported; otherwise, the protocol will be said to be correct and safe. State based methods are efficient and usually can find attacks quickly. Many attacks have been found by this kind of methods. However, it is difficult to effectively control the size of the state space; when the protocol is large, the state space will become surprisingly huge and it will be extremely time consuming, or even practically impossible, to search the whole space.

Famous state based methods include Lowe's CSP Method [5] and Meadows' NRL Protocol Analyzer [7, 8]. In Lowe's CSP Method, The Failure Divergences Refinement (FDR) Checker is a model checker for CSP, which is used to describe concurrent systems whose component processes interact with each other by communication. In Lowe's method, FDR takes two CSP processes, namely a specification and an implementation, as input, and tests whether the implementation refines the specification. It has been used to analyze many sorts of systems, including communication protocols and distributed databases. The NRL Protocol Analyzer developed by Meadows [7, 8] is a prototype special purpose verification tool, written in Prolog. As most other authentication protocol verification methods, the NRL Protocol Analyzer makes the assumption that principals communicate over a network controlled by a hostile intruder who can read, modify and destroy traffic, and also perform some operations, such as encryption, that are available to legitimate participants in the protocols. It has been developed for the analysis of cryptographic protocols that are used to authenticate principals and services and distribute keys in a network.

Rule based methods, also called belief logic methods in some literatures, formally express what a principal can infer from messages received [10]. With this approach, the protocols, the necessary assumptions and the goals of the protocols are formulated in formal logic. Some specific properties of the protocols can be proved by using the axioms and rules of the logic [4]. Since rule based methods do not have to search large state space, they can normally converge very quickly in less than one hundred steps for a medium sized protocol, which is much more efficient than state exploration methods, and they do have found many subtle flaws. However, some of them only consider single runs of the protocols and usually ignore the interleaving of two or more sessions. So there are a lot of flaws which cannot be found with these logics.

In the recent years, a number of rule based methods and belief logics have been developed. Among them, the BAN logic introduced by Burrows et al. [3] and Bolignano's model [2] are good representatives. In addition, a special kind of rule based models are also interesting. They construct proofs using specific rules. Paulson's inductive inference method [10] serves as a good example.

The BAN logic was proposed in 1989 and named after its three creators: Michael Burrows, Martin Abadi and Roger Needham [3]. Their aim to create this logic was to provide a theoretical tool for formally analyzing protocols for the authentication of principals in distributed computing systems. It is actually the first and probably the most well-known rule based method for verifying security protocols, which began the research of protocol formal verification. Bolignano's approach [2] is based on the use of general purpose formal methods. It is complementary with belief logic methods as it allows for a description of protocol, hypotheses and authentication properties at a finer level of precision and with more freedom. Paulson's inductive method [10] is also impressive. He has introduced several message analysis and synthesis operators, namely *parts*, *analz* and *synth*, which can be inductively defined. These operators can be used to describe what information a principal can get and produce from what he has known.

We also have proposed a framework [6] to verify cryptographic protocols, which is knowledge-based and forms the basis of this paper.

3 The knowledge based framework

Our framework is composed of a number of basic notations, predicates, action functions, assumptions and rules, which are described as follows.

3.1 Basic notations

Individuals taking part in network communications are called *principals*, denoted with uppercase letters. Conventionally, *A*, *B*, ..., stand for "honest" principals, while *I* stands for the *intruder*, or, in some literatures, the *adversary* or the *spy*, which are synonyms in our context.

Random numbers chosen by principals serve as *nonces* to identify protocol runs uniquely and avoid replay attack [10]. Nonces are normally denoted as N_a,

N_b, etc, where the subscripts imply the producers of the nonces.

Every principal has some keys. In public key cryptosystem, the principal A basically has a *public key* and a corresponding *private key*, which are denoted as K_a and K_a^{-1}, respectively. On the other hand, in symmetric key cryptosystem, two communicating principals A and B normally share a session key, denoted as K_{ab}.

A *message* is a piece of information sent from one principal to another. A message can consist of names of principals, keys, nonces, or the combination of them. Compound messages are bracketed using curly braces, such as $\{A, N_a\}$. A compound message consisting more than two components can be understood as nested compound message. For example, $\{M_1, M_2, M_3\}$ is the abbreviation of $\{\{M_1, M_2\}, M_3\}$. A message can be encrypted or signed with a certain key. An encrypted message is attached with the key as the subscript. For example, a message M encrypted with B's public key can be written as M_{K_b}. To gain clarity, we usually add curly brackets to the encrypted message, such as $\{M\}_{K_b}$.

3.2 Predicates

Several predicates are used in our framework to describe certain belief and knowledge states.

The predicate $Know(X, M)$ describes X's knowledge state about message M. If X knows that M is true, then the logical value of the predicate is true; if X knows that M is false, then its logical value is false; if X has no information about the truth of M, then its logical value is ignorant; if X has both information to conclude that M is true *and* false, then its logical value is inconsistent.

Similarly, the predicate $Auth(X, Y, M)$ describes X's authentication state about Y on message M, that is, whether message M is sent by Y to X and unmodified.

3.3 Action functions

A cryptographic protocol can be regarded as a series of message sending behaviours between different principals. We introduce action functions to describe this kind of behaviours. An action function $Send(X, Y, M)$ means that the principal X sends the message M to another principal Y.

Accordingly, an action function $Rcv(X, M)$ means that the principal X receives the message M from another.

3.4 Assumptions

Our inference system is based on some assumptions, which are widely accepted by most researchers in this field.

- In public cryptosystems, the public key of any principal is known to all other principals, while its private key is initially secret from others. Formally,

$$\forall X. \forall Y. (Know(X, K_Y))$$

$$\forall X. \forall Y. (X \neq Y \rightarrow \neg (Know(X, K_Y^{-1})))$$

- In symmetric cryptosystems, the session key shared between two principals is initially secret and unknown by all other principals (except the key server who knows all the keys in the whole system).

- The intruder always observes all messages sent through the network. He tries to use all the keys he knows to decrypt the messages on the network and send forged messages to others. He can also intercept messages sent from one principal to another. That is, the intruder has the "full" control over the network.

- There is only one intruder in the network. We always use I to denote the intruder.

- The intruder cannot read an encrypted message without the corresponding decryption key; i.e. secret keys are unguessable.

- An honest principal only read information addressed to him.

- A principal never sends messages to himself.

- Nonces are always different from each other.

3.5 Rules

A group of inference rules have been introduced into our framework to infer new knowledge from the old.

All these rules can be divided into the following categories:

(1) Encryption/Decryption rules

1.1 $Know(X, M) \wedge Know(X, K) \rightarrow Know(X, M_K)$

When a principal knows a message and a key, he can use this key to encrypt (sign) this message and get the encrypted (signed) message.

1.2 $Know(X, M_K) \wedge Know(X, K^{-1}) \rightarrow Know(X, M)$

When a principal knows a message encrypted (signed) with a key and the reverse of the key, he can use the reverse of the encryption (signature) key to get the original message. In public key cryptosystem, the public and private keys of a principal are reverses of each other. In symmetric cryptosystem, the reverse of a session key between two principals is itself (or a simple function of itself).

(2) Message combination/separation rules

2.1 $Know(X, M_1) \wedge Know(X, M_2) \rightarrow Know(X, \{M_1, M_2\})$

When a principal knows two messages, he can know the combination of them.

2.2 $Know(X, \{M_1, M_2\}) \rightarrow Know(X, M_1) \wedge Know(X, M_2)$

When a principal knows the combination of two messages, he can know them separately. These two rules can be used inductively to deal with compound messages consisting of more than two components.

(3) Message sending/receiving rules

3.1 $Send(X, Y, M) \rightarrow Rcv(Y, M)$

If a principal sends a message to another one, the object principal will eventually receive it.

3.2 $Send(X, Y, M) \rightarrow Rcv(I, M)$

As one of our assumptions describes, the intruder I can observe all information flowing over the network.

3.3 $Rcv(X, M) \rightarrow Know(X, M)$

After a principal receives a message, he will know it.

(4) Miscellaneous inference rules

4.1 $Send(X, Y, M) \rightarrow Know(X, M)$

If an honest principal X sends a message M to another principal Y, she must know the message M first.

4.2 $Rcv(X, M) \rightarrow (X \neq I) \rightarrow \exists Z. ((Z \neq X) \wedge Send(Z, X, M))$

If an honest principal X received a message M, then there must exist another principal who sent this message to him (according to one of the assumptions, X cannot send messages to himself).

4 Modelling Needham-Schroeder public key protocol

A cryptographic protocol can be modeled as a series of *send* action functions, each meaning that a principal sends a message to another one. Here we take the famous Needham-Schroeder public key authentication protocol as an example, which is often used to demonstrate the effectiveness of a protocol verification method and usually regarded as a standard "testbed".

The original Needham-Schroeder public key protocol [9] involves seven steps, four of which are concerning public key distribution procedures. In our model, all principals' public keys are open and known to the entire world. Therefore, the protocol can be simplified to only three steps [5] as follows.

1. $A \rightarrow B$: $\{N_a, A\}_{K_b}$
2. $B \rightarrow A$: $\{N_a, N_b\}_{K_a}$
3. $A \rightarrow B$: $\{N_b\}_{K_b}$

The aim of this protocol is to establish mutual authentication between an initiator A and a responder B. As its name indicates, the protocol uses public key cryptography.

At the beginning, A composes a fresh nonce N_a and sends it to B with her own name, encrypted with B's public key. After B receives this message, he decrypts it and then reads the nonce N_a and knows who is seeking to

communicate with him. B then sends back a message consisting of his own fresh nonce N_b as well as A's nonce N_a, encrypted with A's public key K_a to A. To respond B's message, A then returns B's nonce N_b to B.

This protocol can be easily modeled as three message sending actions.

NS-1: $Send(A, B, \{N_a, A\}_{K_b})$

NS-2: $Send(B, A, \{N_a, N_b\}_{K_a})$

NS-3: $Send(A, B, \{N_b\}_{K_b})$

5 Proving authentication properties

To analyze the Needham-Schroeder public key protocols, an important property which needs to be proved is authentication for the principal A, i.e. if A has used N_a to start a run and then receives message $\{N_a, N_b\}_{K_a}$ (from somebody), she will believe that another principal B really sent the message $\{N_a, N_b\}_{K_a}$ to her. Similarly, the authentication property also needs to be proved for B.

At the beginning, we need to state some assumptions useful for the proof: every principal knows all other's public key, but does not know any private key except that of himself.

(1)	$Know(A, K_b)$	Assumption
(2)	$Know(B, K_a)$	Assumption
(3)	$Know(I, K_a)$	Assumption
(4)	$Know(I, K_b)$	Assumption
(5)	$Know(A, K_i)$	Assumption
(6)	$Know(A, K_a^{-1})$	Assumption
(7)	$Know(B, K_b^{-1})$	Assumption
(8)	$Know(I, K_i^{-1})$	Assumption
(9)	$\neg Know(I, \{N_b\}_{K_b})$	Assumption

5.1 Authentication property for A

The authentication property for A can be formalized as a logic formula need to prove

$$Send(B, A, \{N_a, N_b\}_{K_a})$$

with two premises:

(10)	$Send(A, B, \{N_a, A\}_{K_b})$	Premise
(11)	$Rcv(A, \{N_a, N_b\}_{K_a})$	Premise

We start our inference from these two premises.

(12)	$Know(B, \{N_a, A\}_{K_b})$	[4.1] (10)
(13)	$Know(B, \{N_a, A\})$	[1.2] (7)(12)
(14)	$Know(B, N_a)$	[2.2] (13)
(15)	$\exists Z.((Z \neq A) \wedge Send(Z, A, \{N_a, N_b\}_{K_a}))$	[4.2] (11)

Besides A, we only need to consider two different principals in the system, namely B and the intruder I, according to the assumptions. Therefore there are only two possibilities for the formula (15): Z stands for B or I.

(16)	$Send(B, A, \{N_a, N_b\}_{K_a})$	I (15)
(17)	$Send(I, A, \{N_a, N_b\}_{K_a})$	I (15)

If formula (17) holds, then we get

(18)	$Know(I, \{N_a, N_b\}_{K_a})$	[4.1] (17)

For this formula, we still have two possibilities:

(19) $Know(I, N_a)$

(20) $Know(I, N_b)$

(21) $Know(I, K_a^{-1})$

which conflict with assumptions, and

(22) $Rcv(I, \{N_a, N_b\}_{K_a})$

from which we get

(23) $Send(B, I, \{N_a, N_b\}_{K_a}))$

which does not hold according to the protocol steps, or I intercept this information from the communication between A and B, implying

(24) $Send(B, A, \{N_a, N_b\}_{K_a})$

which is exactly the same as the formula (16).

Therefore, we can safely conclude that if A has used N_a to start a run and then receives message $\{N_a, N_b\}_{K_a}$, she will believe that another principal B really sent the message $\{N_a, N_b\}_{K_a}$ to her.

5.2 Authentication property for B

The authentication property for B states that if B receives $\{N_b\}_{K_c}$ (from somebody) and used N_b in step 2 of the protocol, then he can believe that A has sent $\{N_b\}_{K_c}$ to somebody. This property can be similarly formalised as:

$$\exists C.\ Send(A, C, \{N_b\}_{K_c})$$

with two premises:

(25)	$Send(B, A, \{N_a, N_b\}_{K_a})$	Premise
(26)	$Rcv(B, \{N_b\}_{K_b})$	Premise

From premises (25), we can get

(27)	$Rcv(A, \{N_a, N_b\}_{K_a})$	[3.1] (25)
(28)	$Know(A, \{N_a, N_b\}_{K_a})$	[3.3] (27)
(29)	$Know(A, \{N_a, N_b\})$	[1.2] (6)(28)
(30)	$Know(A, N_b)$	[2.2] (29)
(31)	$Know(A, \{N_b\}_{K_b})$	[1.1] (1)(30)

From premises (26), we can get

(32)	$\exists D.((D \neq B) \wedge Send(D, B, \{N_b\}_{K_b}))$	[4.2] (26)

According to the assumptions, there are only two possibilities for the formula (32): D stands for A or I.

(33)	$Send(A, B, \{N_b\}_{K_b})$	I (32)
(34)	$Send(I, B, \{N_b\}_{K_b})$.	I (32)

If the formula (33) holds, then the authentication property for B has been proved. If the formula (33) does not hold, the formula (34) will hold, and then we can get

(35)	$Know(I, \{N_b\}_{K_b})$	[4.1] (34)

There are still two possibilities for it:

(36) $Know(I, N_b)$

which conflict with assumptions, and

(37) $Rcv(I, \{N_b\}_{K_b})$

Since it is impossible for B or I itself (according to the assumptions) to send $\{N_b\}_{K_b}$ to I, the only possibility is that A sent this message to I, i.e.

(38) $Send(A, I, \{N_b\}_{K_b})$

Whichever of formula (33) or (38) holds, we can always conclude

$$\exists C.\ Send(A, C, \{N_b\}_{K_c})$$

having proved the authentication property for B.

It should be noticed that the authentication properties for A and B are not exactly symmetric to each other. Actually, the original Needham-Schroeder public key authentication is insecure. The intruder can break the protocol and impersonate A in the communication with B. Therefore, it cannot be guaranteed that B can believe A has sent $\{N_b\}_{K_c}$ to him if B receives $\{N_b\}_{K_c}$ and used N_b in step 2 of the protocol.

Actually, we can also use our framework to find the flaw in the protocol. In the above proof, formula (38) and (34) indicate the possibility that the intruder I impersonates A. In (38), A may send the secret information $\{N_b\}_{K_b}$ to I, who can in turn send it to B, impersonating A and befooling B. That is why we can only prove a "weaker" authentication property for B.

6 Conclusions and future work

In this paper, we have presented a new framework to prove secure properties for cryptographic protocols. The framework is a knowledge-based one. We have introduced notations, predicates, action functions and a number of rules to describe the knowledge and belief states of principals and the relationships among them. The rules in our framework give the conditions under which the knowledge and belief states can be changed,

and how they can change. We have presented the proving processes of the authentication properties for the simplified Needham-Schroeder public key authentication protocol. These examples show how to use our framework.

The examples indicate the effectiveness and efficiency of the framework. It can prove secure properties of cryptographic protocols correctly and efficiently: proving the authentication properties for *A* only needs 15 steps, more efficient than the inductive method, which needs at least six lemmas, each of which needs three or five steps. Although it is quite efficient to prove secure properties of small and medium sized cryptographic protocols, proving large protocols will still be tedious and time-consuming. Therefore, we need implement our framework with some proof tools to make protocol verification an automatic procedure.

Acknowledgements

Our researches have been supported by EC, EPSRC, the National Natural Science Foundation of China, and Hong Kong K C Wang Education Foundation.

References

[1] Giampaolo Bella. "Inductive Verification of Cryptographic Protocols". PhD Dissertation, University of Cambridge, 2000.

[2] Dominique Bolignano. "An Approach to the Formal Verification of Cryptographic Protocols". *Third ACM Conference on Computer and Communications Security*, pp 106-118. ACM Press, 1996.

[3] Michael Burrows, Martin Abadi and Roger Needham. "A Logic of Authentication". *ACM Transactions*, Computer Systems 8(1), 1990.

[4] Armin Liebl. "Authentication in Distributed Systems: A Bibliography". *Operating Systems Review*, 27(4):122-136, October 1993.

[5] Gavin Lowe. "Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR". *Tools and Algorithms for the Construction and Analysis of Systems*, Margaria and Steffen (eds.), volume 1055 of Lecture Notes in Computer Science, Springer Verlag, pages 147-166, 1996.

[6] Xiaoqi Ma and Xiaochun Cheng. "Verifying Cryptographic Protocols". *IEEE SMC UK-RI 3rd Conference on Intelligent Cybernetic Systems (ICS'04)*, pp 52-57, Londonderry, UK, September 2004.

[7] Catherine A. Meadows. "The NRL Protocol Analyzer: An Overview". *Journal of Logic Programming*, 26(2):113-131, February 1996.

[8] Catherine A. Meadows. "Analyzing the Needham-Schroeder Public-Key Protocol: A Comparison of Two Approaches". In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *Computer Security ESORICS 96*, LNCS 1146, pp 351-364. Springer, 1996.

[9] Roger Needham and Michael Schroeder, "Using encryption for authentication in large networks of computers", *Communications of the ACM*, Vol 21, No. 12, pp. 993-999, 1978.

[10] Lawrence C. Paulson. "Proving Properties of Security Protocols by Induction". *Proceedings of the 10th Computer Security Foundations Workshop*, pp 70-83, 1997.