# Proving Authentication Property of Modified Needham-Schroeder Protocol with Logic of Events

M.H. Xiao, C.Y. Deng, C.L. Ma, K. Zhu, D.L. Cheng
School of Software, East China Jiaotong University
Nanchang, P.R. China

***Abstract*—Security protocols are the foundation of modern secure networked systems. Proving security properties of cryptographic protocols is a challenge problem. Model checking has proven useful for finding certain classes of errors in network security protocols, but it is based on bounded model or constraint solving, and provides little insight into why a protocol is correct. While theorem proving puts no bound on the size of the principal and requires no state space enumeration. We present novel proof rules and mechanisms for protocol actions and temporal reasoning to check security properties of cryptographic protocols using logic of events theory. The logic is an event-ordering which extended by seven special event classes *New, Send, Receive, Sign, Verify, Encrypt, and Decrypt,* and axioms *AxiomK, AxiomR, AxiomV, AxiomD, AxiomS, AxiomF, Disjointness* axioms and Flow relation. As a case study, our method is illustrated by showing the proof of the modified Needham-Schroeder protocol. Result shows the logic of events is feasible and general for analyzing cryptographic protocols.**

***Keywords-Formal method; Logic of Events; Authentication of cryptographic protocols***

## I. INTRODUCTION

Proving security properties of network protocols is a complex problem. Formal method is a powerful method to prove the security properties of cryptographic protocols, which mainly includes model checking and theorem proving two parts. Existing methods based on model-checking are useful for finding bugs, but do not guarantee protocol security for an unbounded number of sessions. Theorem proving can deal with infinite state space problem. While explicit reasoning about traces that containing honest principals' and attacker's actions using theorem-proving approaches requires considerable effort and expertise. Many scholars have done in-depth research on theorem proving method, and achieved remarkable achievement [1][2].

A recent development in formal security protocol analysis is the Logic of Events [3][4]. In 2003 Mark Bickford defined a logic of events [4] that justifies the extraction of correct distributed processes from constructive proofs that system specifications are achievable, and the extraction process in logic of events can be implemented in the context of construction type theory. The researchers have made great contribution to logic of events [5][6][7][8][9][10]. In this paper, we describe specific extension and mechanism to logic of events, we call authentication event logic. Authentication event logic guarantees that any well-typed protocol is robustly safe under attack while reasoning only about the actions of honest principals in the protocol. It puts no bound on the size of the principal and requires no state space enumeration and it is decidable.

The rest of the paper is organized as follows. Section 2 describes the notations and operational semantics for event of logic. Section 3 inference rules are given. Section 4 presents six signification axioms. Case study is discussed in section 5 with conclusions in section 6.

## II. THE NOTATIONS AND SEMANTICS

This section describes the notations and semantics for analyzing security properties of cryptographic protocols. First we give a few symbols and the meaning of them:

$\mathsf{B}$ : Booleans $T$ : type $a, b, \cdots$ : represent atoms

$t$ : term of type $T$ $E$ : one of the seven event classes

$s$ : ciphertext

$A, B, \cdots$ : represent identifiers $bs$ : basic sequence

 

TABLE I. THE BASIC NOTATIONS AND SEMANTIC OF LOGIC OF EVENTS.

| *e or event* | **The points where information is transferred** | *Atom* | **consist of basis element that cannot be generated** |
|---|---|---|---|
| $loc(e)$ or $Id$ | the agent at which event *e* occurs $Honest : Id \to \mathrm{B}$ | *Data* | with identifiers and atoms that can represent all the messages and plaintexts |
| $info(e)$ | the original information had a relationship with event *e* | $(t : T \parallel a)$ | term $t$ of type $T$ is independent of atom $a$ |
| $thr$ | thread is an ordered set of operations at single location | *k or Key* | represent unguessable keys as atoms, and public keys as identifiers, defined as $Key \equiv_{def} Id + Atom + Atom$ |
| $New(e)$ | the nonce chosen at event $e$ and has type $Atom$ | $Rcv(e)$ or $Send(e)$ | the message sent or received at event *e* |
| $Sign(e) = \langle x, A, s \rangle$ | Principal *A* signs plaintext *x* to produce signature *s* | $Verify(e') = \langle x, A, s \rangle$ | principal *B* successfully verifies that s is the signature of *A* on plaintext *x* |
| $Encrypty(e) = \langle x, k, c \rangle$ | principal $A$ encrypts plaintext $x$ with key $k$ to produce ciphertext $c$. | $Decrypty(e') = \langle x, k', c \rangle$ | Principal *B* successfully decrypts ciphertext $c$ using key $k'$ to produce plaintext $x$ |
| $<$ | is a valid and locally-finite transitive relation on *E* | $\mathrm{P}(e)$ | represent the property of event $e$ |
| $s \sim r$ | a weak match：*s* is a send and *r* is a receive own the same information | $s \mapsto r$ | $s \sim r$ and $S$ is causally before $r$ (a strong match) |
| $thr_1 \preceq thr_2$ | thread $thr_1$ is an initial segment of thread $thr_2$ | $thr_1 \simeq thr_2$ | defined as $thr_1$ 鞍$thr_2 \vee thr_2$ $thr_1$ |
| $thr_1 \overset{n}{\approx} thr_2$ | a strong matching conversation of length n, each pair $\langle m_1, m_2 \rangle$ satisfies $m_1 \mapsto m_2 \vee m_2 \mapsto m_1$ | $thr_1 \overset{n}{\sim} thr_2$ | a weak matching conversation of length n, each pair $\langle m_1, m_2 \rangle$ satisfies $m_1 \sim m_2 \vee m_2 \sim m_1$ |

## III. INFERENCE RULES

First we introduce an event-ordering and corresponding type $ProtocolAction$. Every event has a location, and there is a natural causal-ordering on the set of events, the ordering first considered by Lamport [11]. This allows us to define an event-ordering, a structure, $\langle E, loc, <, in\ fo \rangle$, in which the causal ordering $<$ is transitive relation on $E$ that is well-founded, and locally-finite(each event has only finitely many predecessors). Also, the events at a given location are totally ordered by $<$. The information, $in\ fo(e)$, associated with event $e$ is the message input to $loc(e)$ when the event occurred.

We describe protocols by classifying the events in the protocol. In authentication protocols there are send, receive, nonce, sign, verify, encrypt and decrypt events. Events in each class have associated information, and the type of this information depends on the class of the event. Corresponding with the seven event classes [12] [13] we defined a type $ProtocolAction$ of the allowed actions. The members of $ProtocolAction$ are in one of the seven sets are give out in figure 2.

$$New : EClass(Atom)$$

$$Send, Rcv : EClass(Data)$$

$$Encrypt, Decrypt : EClass(Data \times Key \times Atom)$$

$$Sign, Verify : EClass(Data \times Id \times Atom)$$

Fig. 1：Event classes of the authentication theroy

$\{\mathbf{new}(a) \mid a \in Atom\}$

$\{\mathbf{send}(x) \mid x \in Data\}$

$\{\mathbf{rcv}(x) \mid x \in Data\}$

$\{\mathbf{sign}(t) \mid t \in (Data \times Id \times Atom)\}$

$\{\mathbf{verify}(t) \mid t \in (Data \times Id \times Atom)\}$

$\{\mathbf{encrypt}(t) \mid t \in (Data \times Id \times Atom)\}$

$\{\mathbf{decrypt}(t) \mid t \in (Data \times Id \times Atom)\}$

Fig. 2: Type *protocolAction*

Then we present the inference rules

(1) For any $e_1, e_2 \in E(X)$ If $e_1 \xrightarrow{a} e_2$ then $e_1 \le e_2$ and $e_2$ has $a$

(2) For any $e_1, e_2 \in E(X)$ If $e_1 \xrightarrow{a} e_2$ then $release(a, e_1, e_2)$

(3) (unique nonces). If $e_1, e_2 \in E(New)$ and $New(e_1) = New(e_2)$ then $e_1 = e_2$

Definition 1: If for all $e' \in E(New)$ and any event $e$ : $\neg(e' \le e) \Rightarrow (val(e) \square New(e'))$

Definition 2 Principle（Agent or Thread） $A$ does not release nonce $m$ before $e$ then

$$\exists\, e' \in \mathrm{New}(m).\ loc(e') = A \ \wedge\ \forall e''.\ e' < e'' < e.\ e'' \notin \mathrm{E(Send)}$$

(4) Define protocol $Pr$ to be basic sequence $bs$ authenticates $n$ messages

$$\begin{aligned}
&Pr \models auth(bs, n) \equiv_{def} \ \forall A, B.\ \forall thr_1. \\
&(Honest(A) \wedge Honest(B) \wedge Pr(A) \wedge Pr(B) \\
&\quad \wedge A \ne B \wedge loc(thr_1) = A \wedge bs(A, B, thr_1)) \\
&\Rightarrow \exists thr_2.\ loc(thr_2) = B \wedge thr_1 \overset{n}{\approx} thr_2
\end{aligned}$$

(5) Predicates of **has** and **potentially has**

$$\text{Event } e \ \mathbf{has} \text{ atom } a \equiv (e \in E(X) \wedge \neg(X(e) : T \square a))$$

$$\begin{aligned}
&e \ \mathbf{potentially\ has}\ a \ \equiv_{rec} \ (e \ \mathbf{has}\ a) \\
&\quad \vee\ \exists\, e' :\ \mathrm{E}.\ (e' < e) \wedge (e' \ \mathbf{potentially\ has}\ a) \\
&\quad \wedge \begin{pmatrix} (e' \in Sign \ \wedge\ e \ \mathbf{has}\ Sign(e')) \\ \vee\ (e' \in Encrypt \wedge\ e \ \mathbf{has}\ Encrypt(e')) \end{pmatrix}
\end{aligned}$$

(6)Temporal reasoning in Authentication Event Logic

Property of events $\square$P and <> P are expressed respectively by:

$(\square \mathrm{P})(e) \equiv \forall e'.\ e \le_{loc} e' \Rightarrow \mathrm{P}(e')$

$<> \mathrm{P}(e) \equiv \exists e'.\ (e \le_{loc} e' \wedge \mathrm{P}(e'))$

$<> \mathrm{P} \equiv \lambda e.\ \exists e'.\ e \le_{loc} e' \wedge \mathrm{P}(e')$, it is suitable at Lamda Calculus.

## IV. THE AXIOMS

Our theory includes six axioms [13] that we will discuss in turn. Then we give out them as follows.

**(1)Key axiom**

There we define two functions $MatchingKeys : Key \to Key \to \mathrm{B}$ and $PrivKey : Id \to Atom$ that provide a relation on keys. The key axiom is the following $AxiomK$ :

$$\begin{aligned}
&AxiomK: \quad \forall A, B : Id.\ \forall k, k' : Key.\ \forall a : Atom \\
&MatchingKeys(k; k') \Leftrightarrow MatchingKeys(k'; k) \wedge \\
&MatchingKeys(Symm(a); k) \Leftrightarrow k = Symm(a) \wedge \\
&MatchingKeys(P\mathrm{r}ivKey(A); k) \Leftrightarrow k = A \wedge \\
&MatchingKeys(A; k) \Leftrightarrow k = P\mathrm{r}ivKey(A) \wedge \\
&P\mathrm{r}ivKey(A) = P\mathrm{r}ivKey(B) \Leftrightarrow A = B
\end{aligned}$$

**(2)Causal axioms**

Causal axioms include three axioms. They are relate events in classes $Rcv$, $Verify$ and $Decrypt$ to corresponding, causally earlier, events in classes $Send$, $Sign$ and $Encrypt$. $AxiomR$ and $AxiomV$ are say that a successful receive or verify event must be preceded by a matching sending or signing event with the same associated information.

$$\begin{aligned}
AxiomR:\ &\forall e : E(Rcv).\exists e' : E(Send). \\
&(e' < e) \wedge Rcv(e) = Send(e') \\
AxiomV:\ &\forall e : E(Verify).\exists e' : E(Sign). \\
&(e' < e) \wedge Verify(e) = Sign(e')
\end{aligned}$$

$AxiomD$ is similar to $AxiomR$ and $AxiomV$ except for that the key is a matching key instead of the same key.

$$\begin{aligned}
AxiomD:\ &\forall e : E(Decrypt).\exists e' : E(Encrypt). \\
&e' < e \wedge DEMatch(e, e') \\
DEMatch(e, e') \equiv_{def}\ &plaintext(e) = plaintext(e') \\
&\wedge ciphertext(e) = ciphertext(e') \\
&\wedge MatchingKeys(key(e); key(e'))
\end{aligned}$$

**(3) Disjointness axioms**

Two axioms describe disjointness assumptions. The first simple and clear says that an event in and only in one of the seven special classes. As follows, it is described concisely using a function

$$\begin{aligned}
ActionsDisjoint : \exists f : E \to \square.\forall e : E. \\
&(e \in E(New) \Rightarrow f(e) = 1) \wedge \\
&(e \in E(Send) \Rightarrow f(e) = 2) \wedge \\
&\ldots \wedge \ldots \\
&(e \in E(Decrypt) \Rightarrow f(e) = 7)
\end{aligned}$$

Because a signature may be an application of a cryptographic hash technology, so a ciphertext may be a signature of a plaintext. The second says that on condition the hash of a well-formed member of type $Data$ a signature will not be equal to a ciphertext.

$$
\begin{aligned}
&NonceCiphersAndKeyDisjoint: \\
&\forall n: E(New).\forall s: E(Sign).\forall e: E(Encrypt).\forall A: Id. \\
&New(n) \neq signature(s) \\
&\wedge New(n) \neq ciphertext(e) \\
&\wedge New(n) \neq Private(A) \\
&\wedge ciphertext(e) \neq Private(A) \\
&\wedge signature(s) \neq Private(A) \\
&\wedge signature(s) \neq ciphertext(e)
\end{aligned}
$$

**(4) Honest axiom**

Honest agents keep their private keys, so the signer of sign events must be honest, and decryption or encryption events that use the private key must occur at the honest agent. We named this axiom $AxiomS$

$$
\begin{aligned}
&AxiomS: \forall A: Id. \forall s: E(Sign). \\
&\quad \forall e: E(Encrypt). \forall d: E(Decrypt). \\
&Honest(A) \Rightarrow \{signer(s) = A \Rightarrow (loc(s) = A) \\
&\wedge key(e) = PrivateKey(A) \Rightarrow (loc(e) = A) \wedge key(d) = PrivateKey(A) \Rightarrow (loc(d) = A)\}
\end{aligned}
$$

**(5) Flow relation**

The actions in any of the seven special classes define as type $Act$. The relation ($e$ **has** $a$) is true when action e has atom a. There is an obvious conclusion as below:

$$
\begin{aligned}
&e \textbf{ has } a \equiv_{def} (e \in E(New) \wedge New(e) \textbf{ has } a) \\
&(e \in E(Send) \wedge Send(e) \textbf{ has } a) \vee (e \in E(Rcv) \wedge Rcv(e) \textbf{ has } a) \vee \ldots
\end{aligned}
$$

The flow relation $e_1 \xrightarrow{a} e_2$ means that atom a flows from action $e_1$ to action $e_2$. This can happen only in limited ways; either the actions $e_1$ and $e_2$ are at the same location, or there are intervening send and receive events that send atom a "in the clear", or atom a is in the plaintext of an encryption event, and the ciphertext flows to a matching decryption event. The formal, recursive definition of the flow relation is

$$
\begin{aligned}
&e_1 \xrightarrow{a} e_2 =_{rec} (e_1 \textbf{ has } a \wedge e_2 \textbf{ has } a \wedge e_1 \leq_{loc} e_2) \\
&\vee (\exists s: E(Send).\exists r: E(Rcv).e_1 \leq s < r \leq e_2) \wedge Send(s) = Rcv(r) \wedge e_1 \xrightarrow{a} s \wedge r \xrightarrow{a} e_2 \\
&\vee (\exists e: E(Encrypt).\exists d: E(Decrypt). \\
&\quad e_1 < e < d \leq e_2 \wedge DEMatch(d, e) \wedge key(d) \neq Symma(a) \wedge e_1 \xrightarrow{a} e \wedge e \xrightarrow{ciphertext} d \wedge d \xrightarrow{a} e_2)
\end{aligned}
$$

**(6) Nonce axiom**

We call the nonce axiom $AxiomF$, which include three parts. The first part is

$$
AxiomF_1: \forall e_1: E(New). \forall e_2: E. \quad e_2 \text{ has } New(e_1) \Rightarrow e_1 \xrightarrow{New(e_1)} e_2
$$

The two other parts of $AxiomF$ says that if an action has a signature or ciphertext we can only evolve that there is some sign or encrypt action with the same information, the flow relation holds:

$$
\begin{aligned}
&AxiomF_2: \forall e_1: E(Sign).\forall e_2: E. \\
&e_2 \text{ has } signature(e_1) \Rightarrow \\
&\exists e': E(Sign). Sign(e') = Sign(e_1) \\
&\qquad \wedge e' \xrightarrow{signature(e_1)} e_2
\end{aligned}
$$

$$
\begin{aligned}
&AxiomF_3: \forall e_1: E(Encrypt).\forall e_2: E. \\
&e_2 \text{ has } ciphertext(e_1) \Rightarrow \\
&\exists e': E(Encrypt).Encrypt(e') = Encrypt(e_1) \\
&\qquad \wedge e' \xrightarrow{ciphertext(e_1)} e_2
\end{aligned}
$$

## V. Application

We have adopted this method to analyze several cryptographic protocols. It suggests that this is a promising and feasible method to prove authentication property of cryptographic protocols. We focus on the reduced modified Needham-Schroeder protocol which increases the time stamp as an example.

$$
\begin{array}{lll}
Initiator & & Responder \\
I_1 \left\{ \begin{array}{l} \rightarrow new(m) \\ \rightarrow Encrypt(\langle\langle A, m, T_s\rangle, s\rangle) \\ Send(s) \end{array} \right. & & \\
 & & \left.\begin{array}{l} rcv(s) \\ Decrypt(s) \rightarrow \langle A, m\rangle \\ \xrightarrow{\qquad\qquad} new(n) \\ Encrypt(\langle\langle B, m, n\rangle, s'\rangle) \\ send(s') \end{array}\right\} R_1 \\
I_2 \{ rcv(s') & \xleftarrow{\qquad\qquad} & \\
I_3 \left\{ \begin{array}{l} Decrypt(\langle\langle B, m, n\rangle, s''\rangle) \rightarrow \langle m, n\rangle \\ Encrypt(\langle n, s''\rangle) \\ send(s'') \end{array} \right. & & \\
 & \xrightarrow{\qquad\qquad} rcv(s'')\} R_2 & \\
 & Decrypt(s'') \rightarrow n\} R_3 &
\end{array}
$$

Define NS to be $Protocol([I_1, I_2, I_3, R_1, R_2, R_3])$ for the basic sequence relations defined above, we only need to verify that $CR \models auth(I_3, 2) \wedge CR \models auth(R_3, 1)$ .Suppose $A$ and $B$ are both honest and obey NS protocol. Because we defined that the events at a given location are totally ordered by $<$, while in the modified NS protocol we increase the time stamp, according to the inferences and $AxiomD$, $AxiomF$ are easy to proof that the modified NS protocol is security.

## VI. Conclusion

We propose a novel method to prove authentication property of cryptographic protocols based on event logic. The structure of event ordering, $\langle E, loc, <, in\ fo\rangle$, satisfies that events at one location totally ordered [14]. The event ordering is extended by axioms and seven special event classes. We figure out types for the keys, nonces, and messages of the protocol and present novel proof rules and mechanisms for protocol actions and temporal reasoning. The proof of modified Needham-Schroeder protocol illustrates that our method is general to analyze authentication property of cryptographic protocols. We have constructed a platform to verify authentication property of cryptographic protocols. Future work we will attempt to add theorem proving to the verification platform. The fixed platform will achieve automatically to prove authentication property of cryptographic protocols.


## Acknowledgements

This work was supported by Natural Science Foundation

of China (61163005), Science and Technology College in Jiangxi Province Ground Project(KJLD13038), Natural Science Foundation of Jiangxi, China (2007GZS1884, 2010GZS0150, 20132BAB201033), State Key Laboratory for Novel Software Technology Open Project (KFKT2012B18) , Project of Innovation Funds for Postgraduates in Jiangxi Province(YC2013-S173)and East China Jiaotong University (YC2014-X007).

## REFERENCES

[1] Rubin A D, Honeyman P. Formal methods for the analysis of authentication protocols[R]. CITI Technical Report 93-7, Information Technology Division, University of Michigan, 1993

[2] Anupam Datta, Ante Derek, John C. Mitchell, and Arnab Roy. Protocol composition logic (pcl). Electron. Notes Theor. Comput. Sci. 172,pp.311–358,2007.

[3] S. F. Allen, M. Bickford, R. L. Constable, R. Eaton, C. Kreitz, L. Lorigo and E. Moran. Innovations in computational type theory using Nuprl. Journal of Applied Logic, pp.428-469, 2006.

[4] M. Bickford and R. L. Constable. A Logic of Events. Technical Report TR2003-1893, Cornell University, 2003.

[5] Chien, Hung-Yu, Tzong-Chen Wu, and MING-KUEI YEW. Provably Secure Gateway-Oriented Password-Based Authenticated Key Exchange Protocol Resistant to Password Guessing Attacks [J]. Journal of Information Science & Engineering ,29(2 ),pp.249-265,2013.

[6] Berg M. Formal verification of cryptographic security proofs. Faculty of Natural Sciences and Technology, Department of Computer Science, Saarland University [J]. 9. August. 2013.

[7] Constable R L. On Building Constructive Formal Theories of Computation Noting the Roles of Turing, Church, and Brouwer[C]//Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on. IEEE, pp.2-8,2012.

[8] Mark Bickford. Unguessable Atoms: A Logical Foundation for Security. N. Shankar and J. woodcock (Eds.): VSTTE 2008, LNCS 5295, pp. 30-53, 2008.

[9] Mark Bickford and Robert Constable. Formal Foundations of Computer Security, NATO Science for Peace and Security Series, D: Information & communication Security, Vol. 14, pp.29-52, 2008.

[10] Xiao M, Bickford M. Logic of Events for Proving Security Properties of Protocols[C]//Web Information Systems and Mining, 2009. WISM 2009. International Conference on. IEEE, pp.519-523 2009.

[11] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system, Comm ACM, 21(7),pp. 558-65,1978.

[12] Mark Bickford. Component Specification Using Event Classes. Lecture Notes in Computer Science, Vol 5582, 140-155,2009.

[13] M. Bickford and R. L. Constable. Automated proof of authentication protocols in a logic of events. in Proceedings of the 6th International Verification Workshop, pp. 13-30，2010.

[14] Bickford M, Constable R, Guaspari D. Generating event logics with higher-order processes as Realizers , Computer Sicence Technical Report, Cornell University [J]. 2010.