IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 20, 2019, accepted May 5, 2019, date of publication May 8, 2019, date of current version May 20, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2915645*

# LoET-E: A Refined Theory for Proving Security Properties of Cryptographic Protocols

**JIAWEN SONG, MEIHUA XIAO, KE YANG, XIZHONG WANG, AND XIAOMEI ZHONG**
School of Software, East China Jiaotong University, Nanchang 330013, China

Corresponding author: Meihua Xiao (xiaomh@ecjtu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61163005 and Grant 61562026, in part by the Jiangxi Province Major Discipline Academic and Technical Leadership Funding Program under Grant 20172BCB22015, and in part by the Jiangxi Special Graduate Innovation Foundation under Grant YC2018-S261.

**ABSTRACT** Nowadays, more and more new cryptographic protocols are emerging, and the security analysis of emerging cryptographic protocols is increasingly important. The logic of events is an axiomatic method based on theorem proving, designed around message automation with actions for possible protocol steps; it figured out types of information transmitted in the protocols and also presented novel proof rules and mechanism. However, with the emergence of various cryptographic protocols, the logic of events lacks corresponding axioms and rules in the process of proving certain cryptographic protocols, so it needs a further extension. Based on the logical framework of protocol composition logic, this paper presents a refined theory of the logic of events called LoET-E, in which the novel rules about the freshness of nonces, the event attributes of messages, and the states of the predicate is presented; the concepts of *Fresh*, *Gen* and *FirstSend* is introduced; and the definition of *has* and the honesty axiom of LoET is extended. The refined theory can guarantee the correctness, integrity, and validity of the original axioms, ensure the consistency of event classes and basic sequences in the proof process, reduce the complexity and redundancy in the protocol analysis process, and most importantly, extend the provable range of cryptographic protocols.

**INDEX TERMS** Cryptographic protocols, logic of events, LoET-E, theorem proving, information security.

## I. INTRODUCTION

The security of cryptographic protocols plays a vital role in the information security field, but most existent cryptographic protocols have vulnerabilities and defects that have been discovered or not, therefore in some safety-critical areas, we need a set of cryptographic protocols which is rigorous and can truly implement the security properties it claims. Gradually, there is a need for a more rigorous method that is different from conventional security detection and provides a means of security to detect objects that need to be verified, and this more rigorous method is theorem proving.

The initial LoET (Logic of Events) [1], [2] is a logic that justifies the extraction of correct distributed processes from constructive proofs that system specifications are achievable [2], and the extraction process in LoET can be implemented in the context of construction type theory [3]. Later, Dr. Xiao et al [4], [5] described the specific extension and mechanism of LoET, and focused on the latest supplement to the logical framework, which can be used to prove the security attributes of the cryptographic protocols [3]. Now LoET is an axiomatic method based on theorem proving and this logic is designed around a message automation with actions for possible protocol steps [3], [6]. Moreover, this theorem ensures that any well-typed protocol is robustly safe under attack while reasoning only about the actions of honest principals in the protocols [3]. However, due to the continuous and rapid development of cryptographic protocols nowadays, the types of cryptographic protocols increases, and as a result, LoET lacks corresponding axioms and rules in the process of proving certain cryptographic protocols. In order to verify the security properties of new cryptographic protocols, LoET needs further extension.

PCL (Protocol Composition Logic) is a logic for proving security properties of network protocols [7] and is mainly used to verify the authentication and secrecy of the protocols under the public key encryption system. Similar to LoET, PCL is an axiomatic logic [8] and this logic can prove security

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

properties of cryptographic protocols under attack while reasoning only about the actions of honest parties [7]. Moreover, PCL has a longer history of application experience than LoET and it has done a lot of research on step-by-step automation of protocol actions. Particularly, the PCL composition theorems are useful in carrying out larger scale case studies [9]–[11], until now, there are still people who continue to optimize the PCL [12].

Although LoET and PCL are not identical in nature, there are some commonalities to some extent, they all share some same or different advantages and disadvantages. Starting from these similarities and differences, this paper extends LoET with some core ideas of PCL under the premise of retaining the event attributes of LoET.

The specific works include extending the definition of *has* in the flow relation axiom of the LoET, ensuring the correctness, integrity and validity of the original axioms; redefining the source of the signature accepted by the honest principal in LoET and extending the honesty axiom used to describe the behavior of honest principals; introducing the concept of *Fresh* and two derivative concepts *Gen*, *FirstSend* in LoET, ensuring the consistency of event classes and basic sequences in the proof process; according to these three new concepts, presenting a series of important rules about the freshness of nonces, the event attributes of messages and the states of predicate, which can reduce the complexity and redundancy in the protocol analysis process; and the last but not least, extending the provable range of cryptographic protocols.

The rest of the paper is organized as follows, section 2 describes the proof system of LoET and section 3 presents the analysis of LoET and PCL. The specific logical extension process is described in section 4. Section 5 gives an application of LoET-E and section 6 summarizes this paper and gives a prospect for future work.

## II. PROOF SYSTEM OF LOET

The proof system of LoET consists of three parts, namely the basic modeling theory, the axiom system and the formal definition of the protocol. Before giving an introduction, an explanation of the basic symbols of LoET is first given.

**TABLE 1. Explanation of basic symbols.**

| Symbol | Quantity |
|---|---|
| $ID$ | Generally refers to the principals involved in protocol |
| $Atom$ | Class of secret information |
| $Data$ | All messages and plaintext |
| $X$ | Members in $Data$ |
| $e$ | Event |
| $E$ | Event set |
| $n$ | A challenge number in $Nonce$ |
| $loc(e)$ | A function on $e$, represents the principal of $e$ |
| $key(e)$ | The secret key of the principal of $e$ |
| $New(e)$ | The challenge number in $e$ |
| $Send(e)$ | Messages sent in $e$ |
| $Rcv(e)$ | Messages received in $e$ |
| $bs$ | Basic sequences |
| $\leq$ | Local finite partial order |
| $bs$ | $e_1$ occurs before $e_2$ |

### A. BASIC MODELING THEORY

Basic modeling theory includes important formal components that are required when modeling with event logic.

#### 1) UNGUESSABLE ATOMS

In general, only three different types of values are needed to construct LoET in the formal analysis of the protocol, namely $B$, $Id$, and $Atom$ [4] classes. $Atom$ is a unguessable data value, which is generally used to indicate anything we wish to regard as unguessable, such as nonces, signatures, ciphertexts or the keys [13], [14]. Use $\neg(info(e)||a)$ to assert that the information associated with event $e$ contains the $Atom$ $a$, which defines the independence of the information. (1) indicates that the message $x$ of type $T$ is independent of $Atom$ $a$.

$$x : T||a \tag{1}$$

#### 2) EVENT ORDERINGS

From any formal model of distributed computing one can define the runs of a distributed system and can identify, within a run, those points where information is transferred [13].

By default, $e$ is used to refer to events, $info(e)$ is used to refer to information transmitted on $e$, and $info(e)$ is generally associated with event $e$. With these preconditions, we can define the structure of the event orderings [15] as (2).

$$< E, loc, <, info > \tag{2}$$

#### 3) EVENT CLASSES

By classifying each different event in the protocol authentication process, it can be found that the events have associated information, and the types of the information depends on the event classes [16]. Therefore, the event class can be defined, and the events are divided into seven categories: send, receive, new, encrypt, decrypt, sign and verify [17].

The cryptographic protocols exchanges messages containing data tuples such as nonces, signatures, and principals. We can define it as a binary tree $Data \equiv_{def} Tree\ (Id + Atom)$, and this expression is sufficient to represent all the information and plaintext we need, and the protocol action types can be defined as (3).

$$\begin{pmatrix} \{new\ (a)\ |a \in Atom\} \\ \{send\ (x)\ |x \in Data\} \\ \{rcv\ (x)\ |x \in Data\} \\ \{sign\ (t)\ |t \in (Data \times Id \times Atom)\} \\ \{verify\ (t)\ |t \in (Data \times Id \times Atom)\} \\ \{encrypt\ (t)\ |t \in (Data \times Id \times Atom)\} \\ \{decrypt\ (t)\ |t \in (Data \times Id \times Atom)\} \end{pmatrix} \tag{3}$$

### B. AXIOM SYSTEM

LoET axioms formally define the types of message and describe the proof rules of cryptographic protocols, which

is the core of LoET. LoET mainly includes six axioms: Key axiom, Causal axioms, Disjointness axioms, Honesty axiom, Flow relation and Nonce axiom.

#### 1) KEY AXIOM

Key axiom is as shown in (4), it means that the matching key is a symmetric relationship, the symmetric key only matches itself, and the private key assigned to the principals only matches the relative public key. Moreover, there are no two principals with the same private key.

$$\left(\begin{array}{c} AxiomK : \forall A, B : Id.\forall k, k'; Key.\forall a : Atom \\ MatchingKeys(k; k') \\ \Leftrightarrow MatchingKeys(k', k) \\ \wedge MatchingKeys(Symm(a); k) \\ \Leftrightarrow k = Symm(a) \\ \wedge MatchingKeys(PrivKey(A); k) \\ \Leftrightarrow k = A \wedge MatchingKeys(A; k) \\ \Leftrightarrow k = PrivKey(A) \\ \wedge PrivKey(A) = PrivKey(B) \Leftrightarrow A = B \end{array}\right) \tag{4}$$

#### 2) CAUSAL AXIOMS

Causal axioms are shown in (5), in which the receive axiom AxiomR is similar to the verification axiom AxiomV. The specific content is say that for any receive or verify event there must be a causally prior send or sign event with the same associated information [13].

$$\left(\begin{array}{c} AxiomR : \forall e : E(Rcv).\exists e' : E(Send). \\ (e' < e) \wedge Rcv(e) = Send(e') \\ AxiomV : \forall e : E(Verify).\exists e' : \\ E(Sign).(e' < e) \wedge Verify(e) = Sign(e') \\ AxiomD : \forall e : E(Decrypt). \\ \exists e' : E(Encrypt). \\ e' < e \wedge DEMatch(e, e') \equiv_{def} \\ plaintext(e) = plaintext(e') \\ \wedge ciphertext(e) = ciphertext(e') \\ \wedge MatchingKeys(key(e); key(e')) \end{array}\right) \tag{5}$$

AxiomD provides that if the encryption event is the preorder of the decryption events and matches each other, the encryption event will hold and decrypt the same information content, and the key can be matched.

#### 3) DISJOINTNESS AXIOMS

Disjointness axioms also contain two parts, one for the seven event classes and the other for for the challenge number, private key, signature, and ciphertext. The first disjointness axiom is shown in (6), it is stipulated that an event in one of the seven special classes is not in any of the other special classes [13].

$$\left(\begin{array}{c} ActionDisjoint : \exists f : E \to Z.\forall e : E. \\ (e \in E(New) \Rightarrow f(e) = 1) \\ \wedge (e \in E(Send) \Rightarrow f(e) = 2) \\ \wedge \ldots \wedge \ldots \\ (e \in E(Decrypt) \Rightarrow f(e) = 7) \end{array}\right) \tag{6}$$

The second as defined in (7), states that the nonces generated by a principal disjointness with the private key, signature, or ciphertext held by the same principal.

$$\left(\begin{array}{c} NonceCiphersAndDisjoint : \\ \forall n : E(New).\forall s : E(Sign).\forall e : E(Encrypt). \\ \forall A : Id.New(n) \neq signature(e) \\ \wedge New(n) \neq ciphertext(e) \\ \wedge New(n) \neq Private(A) \\ \wedge ciphertext(e) \neq Private(A) \\ \wedge signature(s) \neq Private(A) \\ \wedge signature(s) \neq ciphertext(e) \end{array}\right) \tag{7}$$

#### 4) HONESTY AXIOM

Since an honest principal does not release private key, so an event that uses the honest principal private key must occur on the same principal when a signature event occurs with an honest principal, as in (8), honesty axiom are generated according to this rule.

$$\begin{array}{c} AxiomS : \forall A : Id.\forall s : E(Sign).\forall e : \\ E(Encrypt).\forall d : E(Decrypt). \\ Honest(A) \Rightarrow \\ \left\{\begin{array}{c} signer(s) = A \Rightarrow (loc(s) = A) \\ \wedge key(e) = PrivateKey(A) \\ \Rightarrow (loc(e) = A) \\ \wedge key(d) = PrivateKey(A) \\ \Rightarrow (loc(d) = A) \end{array}\right\} \end{array} \tag{8}$$

#### 5) FLOW RELATION

The flow relation contains the causal orderings between the nonce events. Assuming that *Atom a* flows from $e_1$ to $e_2$, it can only occur in limited ways. A detailed description is given in (9).

$$\begin{array}{c} e_1 \rightarrow^a e_2 =_{rec} (e_1 hasa \wedge e_2 hasa \wedge e_1 \leq_{loc} e_2) \\ \vee \\ \left(\begin{array}{c} \exists s : E(Send).\exists r : E(Rcv).e_1 \leq s < r \leq e_2 \\ \wedge Send(s) = Rcv(r) \wedge e_1 \rightarrow^a s \wedge r \rightarrow^a e_2 \end{array}\right) \\ \vee \\ \left(\begin{array}{c} \exists e : E(Encrypt).\exists d : E(Decrypt). \\ e_1 \leq e < d \leq e_2 \wedge DEMatch(d, e) \wedge \\ key(d) \neq Symma(a) \wedge \\ e_1 \rightarrow^a e \wedge e \rightarrow^{ciphertext} d \wedge d \rightarrow^a e_2 \end{array}\right) \end{array} \tag{9}$$

#### 6) NONCE AXIOM

Nonce axiom is shown in (10), AxiomF1 is about the nature of the flow, and the nonces are associated with unique events. The other two axiom AxiomF2 and AxiomF3 assert a similar relation between signatures and ciphertexts and events with

both signatures and ciphertexts [13].

$$\left(\begin{array}{c} AxiomF_1 : \forall e_1 : E(New). \\ \forall e_2 : E.e_2\ has\ New(e_1) \\ \Rightarrow e_1 \rightarrow^{New(e_1)} e_2 \\ AxiomF_2 : \forall e_1 : E(Sign). \\ \forall e_2 : E.e_2\ has\ signnature(e_1) \\ \Rightarrow \exists e' : E(Sign). \\ Sign(e') = Sign(e_1) \wedge e' \rightarrow^{signature(e_1)} e_2 \\ AxiomF_2 : \forall e_1 : E(Encrypt). \\ \forall e_2 : E.e_2\ has\ ciphertext(e_1) \\ \Rightarrow \exists e' : E(Encrypt). \\ Encrypt(e') = Encrypt(e_1) \\ \wedge e' \rightarrow^{ciphertext(e_1)} e_2 \end{array}\right) \quad (10)$$

### C. FORMAL DEFINITION OF THE PROTOCOL

After defining seven actions and six axioms, it is necessary to formally define the authentication attributes that the protocols needs to satisfy, which will be described by the following definitions.

#### 1) THREADS

A thread is an ordered list of actions at single location, which can be defined in (11).

$$Thread \equiv_{def} \{thr : ActList | \quad \forall i : thr[i] <_{loc} thr[i+1]\} \quad (11)$$

Use (12) to define the weak matching relationship between the threads (the information sent is the same as the information received) and the strong matching relationship (exist a direct causal relationship between send and receive behavior).

$$\left(\begin{array}{c} s \sim r \equiv_{def} s \in E(Send) \wedge r \in E(Rcv) \\ \wedge Send(s) = Rcv(r) \\ s \mapsto r \equiv_{def} s \sim r \wedge s < r \end{array}\right) \quad (12)$$

#### 2) MATCHING CONVERSATIONS

If $thr_1$ and $thr_2$ both contain n messages at least, and the first $n$ messages of each thread are paired, then a matching session of length $n$ is formed. At different event location, if there is a strong matching conversation between the two threads of the protocol, then it is said to satisfy the strong authentication attribute [18]. Similarly, the concept of weak matching conversations can be obtained.

#### 3) BASIC SEQUENCES

A basic sequence is essentially a parameterized list of protocol actions [13]. A principal subject to a protocol consists of any number of threads, each of which is an instance of the basic sequences. It is generally allowed to define a base sequence of two or more principals. The basic sequence is a member of the type (13).

$$Basic \equiv_{def} Id \rightarrow Id \rightarrow Thread \rightarrow P \quad (13)$$

#### 4) AUTHENTICATION

If there is a principal $A$ executing an instance of the complete initiator sequence associated with principle $B$, and assuming $B$ is honest and also obeying the same protocol, then there should be an instance of responder $B$, some of the messages in the protocol will form a matching conversation and the specified message sent by $A$ will be received by $B$. Similarly, if $B$ executes an instance of a complete response sequence, there should be a matching conversation of the same length as $A$. In summary, the formal definition formula is obtained in (14) [18]:

$$\left(\begin{array}{c} pr \models auth(bs, n) \equiv_{def} \forall A, B. \forall thr_1. \\ (Honest(A) \wedge Honest(B) \wedge \Pr(A) \wedge \Pr(B) \\ \wedge A \neq B \wedge loc(thr_1) = A \wedge bs(A, B, thr_1)) \\ \Rightarrow \exists thr_2. loc(thr_2) = B \wedge thr_1 \overset{n}{\approx} thr_2 \end{array}\right) \quad (14)$$

## III. ANALYSIS OF LOET AND PCL

Next, we need to analyze the commonalities and differences between LoET and PCL, so as to facilitate the subsequent logical extension of LoET.

### A. COMMONALITIES BETWEEN LOET AND PCL

Through the descriptions in the previous chapters, we can find that the commonality between LoET and PCL is mainly concentrated in the following aspects:

#### 1) THE BASIC ATTRIBUTES OF BOTH THEORIES

In PCL, knowledge indicates what facts principals may know in the process of protocol execution [7]. In the concept of knowledge, formula $Has(X, x)$, $Fresh(X, t)$, $Gen(X, t)$ and $Contains(t_1, t_2)$ is defined. The specifically action formulas described in (15).

$$\left(\begin{array}{c} Q, R \models Has(A, m) \\ Q, R \models Fresh(A, t) \\ Q, R \models Gen(A, t) \\ Q, R \models Contains(t_1, t_2) \end{array}\right) \quad (15)$$

And the same in temporal ordering, there are two important logical formulas: $Start(X)$ and $FirstSend(P, t, t')$, they can be used to strengthen the authentication properties by imposing ordering between actions of different participants [7].

And we can realize both LoET and PCL belong to theorem proving, and both prove the security by proving the authentication of cryptographic protocols. Moreover, both LoET and PCL define the same nature of strong authentication and all based on matching sessions, get the final result through strong authentication.

#### 2) DEFINITION OF ACTIONS

The proof system of PCL includes syntax and semantics, in which syntax gives more important formally components, and semantics introduce some important predicates and action formulas. And in protocol system, action formulas is used to state actions performed by different threads, and can be divided into seven classes: send, receive, new, encrypt, decrypt, sign and verify. The specifically action formulas

described in (16).

$$\begin{pmatrix} Q, R \models Send(A, m) \\ Q, R \models Receive(A, m) \\ Q, R \models New(A, m) \\ Q, R \models Encrypt(A, ENC_K\{|m|\}) \\ Q, R \models Decrypt(A, ENC_K\{|m|\}) \\ Q, R \models Sign(A, SIG_K\{|m|\}) \\ Q, R \models Verify(A, SIG_K\{|m|\}) \end{pmatrix} \tag{16}$$

Clearly, both LoET and PCL are designed around seven actions (send, receive, new, encrypt, decrypt, sign and verify) and provide operations for basic protocol actions. However, it is classified as event class in LoET and action predicates in PCL.

#### 3) HONEST PRINCIPALS

In PCL, ''honest'' is defined as ''follows one or more roles of the protocol'', and the honesty rule is the most important part in the framework of PCL, because all the derivation processes are carried out around honest principals. The specific honesty rule is shown in (17).

$$\frac{Start(X)[]_X\phi \forall \rho \in Q. \forall P \in BS(\rho).\phi[P]_X\phi}{Honest(\bar{X}) \supset \phi} HON_Q \tag{17}$$

It can be seen from the above introduction that both LoET and PCL use a novel form of induction: ''honesty'', over basic sequences of actions performed by honest principals can be used to derive conclusions about arbitrary runs in the presence of adversary actions. Both of them can prove security properties of cryptographic protocols under attack while reason only about the actions of honest parties [7].

#### 4) AXIOM SYSTEM

The proof system of PCL includes a complete axiom system and proof rules that are closely related to the syntax and semantics mentioned above. In the following part, we will introduce three representative axioms, and in which we use $a$ to denote actions performed by different principals and a to denote the corresponding action predicates.

The first is about protocol actions, we can divide action axioms into two parts, the first part is shown in (18), which mainly describes the change of the protocol state before and after actions.

$$\begin{pmatrix} AA1 : T[a]_X a \\ AA2 : Start(X)[]_X \neg a(X) \\ AA3 : \neg Send(X, t)[b]_X \neg Send(X, t) \\ if\, \sigma Send(X, t) \neq \sigma b for all substitutions \sigma \\ AA4 : T[a; \ldots\ldots; b]_X a < b \end{pmatrix} \tag{18}$$

The second part is shown in (19), which mainly focuses on the ownership and freshness of nonces.

$$\begin{pmatrix} AN1 : New(X, x) \wedge New(Y, x) \supset X = Y \\ AN2 : T[newx]_X Has(Y, x) \supset (Y = X) \\ AN3 : T[newx]_X Fresh(X, x) \\ AN4 : Fresh(X, x) \supset Gen(X, x) \end{pmatrix} \tag{19}$$

Then there is the possession axiom, this axiom analyzes predicate *Has* and defines the attribution of knowledge, which be described as follows.

$$\begin{pmatrix} ORIG : New(X, x) \\ \supset Has(X, x) \\ REC : Receive(X, x) \\ \supset Has(X, x) \\ TUP : Has(X, x) \wedge Has(X, y) \\ \supset Has(X, (x, y)) \\ ENC : Has(X, x) \wedge Has(X, K) \\ \supset Has(X, ENC_K\{|x|\}) \\ PROJ : Has(X, (x, y)) \\ \supset Has(X, x) \wedge Has(X, y) \\ DEC : Has(X, ENC_K\{|x|\}) \\ \wedge Has(X, K) \supset Has(X, x) \end{pmatrix} \tag{20}$$

The last one is about encryption and signature, this axiom is aimed at specifying the unforgeability and uniqueness of the signature, which is shown in (21).

$$\begin{pmatrix} SEC : Honest(\bar{X}) \wedge \\ Decrypt(Y, ENC_{\bar{X}}\{|x|\}) \\ \supset (\bar{Y} = \bar{X}) \\ VER : Honest(\bar{X}) \wedge \\ Verify(Y, SIG_{\bar{X}}\{|x|\}) \wedge \bar{X} \neq \bar{Y} \\ \supset \exists X.Send(X, m) \\ \wedge Contains(m, SIG_{\bar{X}}\{|x|\}) \end{pmatrix} \tag{21}$$

From the above introduction, we can see that both LoET and PCL include axiom systems, which assert that after an action is performed, the indicated thread has performed that action, or state properties of cryptographic operations [7].

In addition to the above commonalities, there are some logical commonalities between LoET and PCL, such as descriptions of unknown information. It is described as *Atom* in LoET and knowledge in PCL, although the description is different, it is essentially one concept.

### B. DIFFERENCES BETWEEN LOET AND PCL

Analysis of the second chapter, we can find that the differences between LoET and PCL are mainly concentrated in the following aspects:

#### 1) THE BASIC ATTRIBUTES OF BOTH THEORIES

In PCL, modal formulas play an important role in protocol logic, and it usually means that after actions $P$ are executed in thread $X$, starting from a state where formula $\theta$ is true, formula $\theta$ is true about the resulting state of X [7]. The formula is as follows (22).

$$\theta[P]_X\varphi \tag{22}$$

First of all, from the perspective of basic theory, we can know that LoET is event-based and event-oriented, while PCL emphasizes changes in state, which also creates a difference in two logical natures. In other words, PCL uses a dynamic logic to express properties of protocols, and LoET express a protocol as one kind of constraint on event orderings, and an authentication property as another kind of constraint [13].

### 2) COMPARISON OF ADVANTAGES AND DISADVANTAGES

Firstly, in the verification of protocol security attributes, PCL can only describe part of the cryptographic protocols, and the cryptographic protocols with the data signatures cannot be discribed, while LoET can discribe the corresponding protocol. Similarly, from the perspective of the thread concept, PCL is not rigorous in the protocol interaction action modeling, and lacks definition of the preceding actions in a thread [8]. While LoET clearly defines the protocol formal modeling thread mechanism, and the atomic independence is used to regulate the event thread.

Moreover, LoET figure out types for the keys, nonces, and messages of protocols, while PCL lacks the necessary restrictions on the data type in the protocol. And PCL method still lacks the ability to describe the hash function, while LoET characterizes the hash function when dealing with encryption actions.

While LoET has many advantages, it uses the flow relation and the nonce axiom to prove the event orderings, instead of definition of the freshness of nonces. So the process of proof is cumbersome and the consistency of event classes and basic sequences is difficult to define.

On the contrary, PCL has sequencing rule, preservation axioms and axioms and rules for temporal orderings, which can make the proof process simple and effective. Among them sequencing rule gives us a way of sequentially composing two cords $P$ and $P'$ when post-condition of $P$, matches the pre-condition of $P'$, this is helpful in the reasoning of the protocols, as shown by (23).

$$\frac{\phi_1[P]_X\phi_2\phi_2[P']_X\phi_3}{\phi_1[PP']_X\phi_3} \tag{23}$$

Preservation axioms indicates that certain actions have their state values unchanged after some action predicates are executed.

$$\begin{pmatrix} P1 : Persist(X,t)[a]_X Persist(X,t) \\ for Persist \in \{Has, FirstSend, a, Gen\} \\ P2 : Fresh(X,t)[a]_X Fresh(X,t) \\ where t \notin a \end{pmatrix} \tag{24}$$

Axioms and rules for temporal orderings gives a method to judge the order of actions in different threads based on the definitions of *Fresh* and *FirstSend*.

$$\begin{pmatrix} FS1 : Fresh(X,t)[sendt']_X FirstSend(X,t,t') \\ where t \subseteq t' \\ FS2 : FirstSend(X,t,t') \wedge a(Y,t'') \\ \supset Send(X,t') < a(Y,t'') \\ where X \neq Y and t \subseteq t'' \end{pmatrix} \tag{25}$$

### 3) PROOF PROCESS

In PCL, after the matching session is proved according to the pre-condition and post-condition, the proof of the protocol only needs to consider two aspects: the proof of weak authentication property and the proof of strong authentication property. However, in addition to weak authentication and strong authentication property, the basic sequences of authentication protocols and bilateral proofs should be considered in LoET, see Figure 1 in detail.

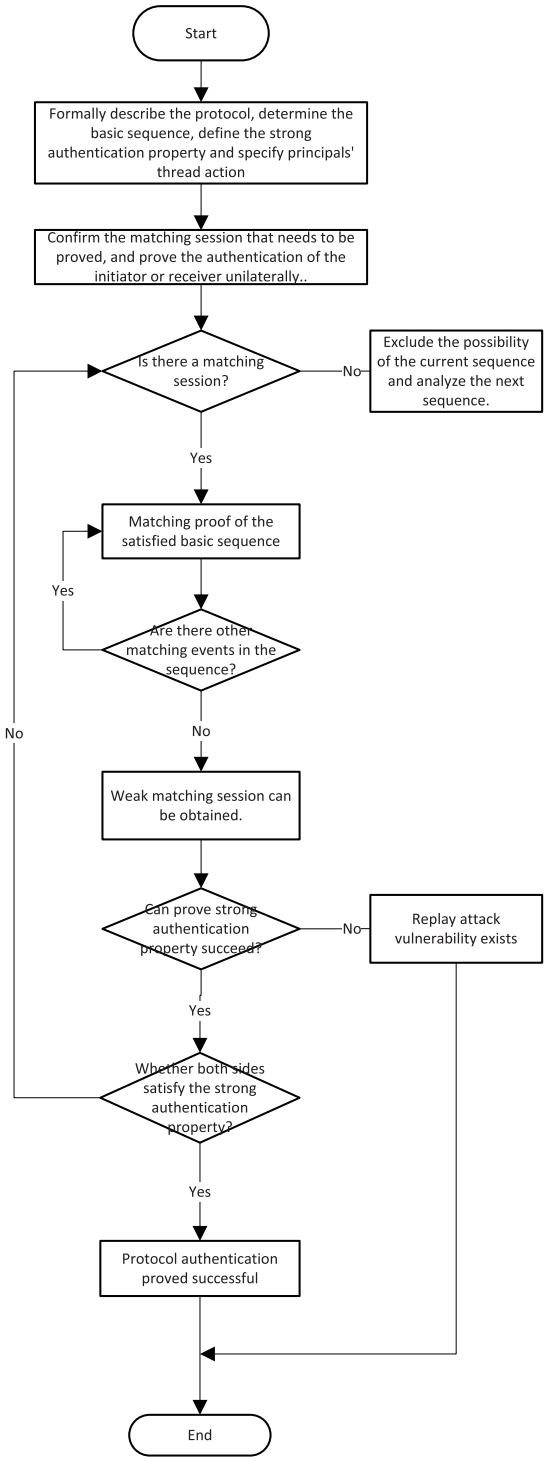



**FIGURE 1. Flowchart for using the LoET certification protocol for authentication.**

### 4) APPLICATION LAYER

PCL has a longer application experience history than LoET, and it has been continuously improving and perfecting in

this process, especially for the step-by-step automation of protocol actions. Although LoET is built on the premise of absorbing many basic ideas, but it has not yet implemented step-by-step automation. From this perspective, PCL has a great advantage over LoET.

In summary, from the above analysis, it can be found that although essentially one is based on events and the other emphasizes state, there are still exist accessible in the proof process of the protocol, so PCL can be used to extend event logic.

## IV. EXTENSION IN LOET-E

In this section, combined with the proof system of LoET and PCL, LoET-E is completely presented. Although it is necessary to extend LoET using a series of axiom rules of PCL based on the above analysis and judgment, it is more important to note that the original event attributes of LoET cannot be destroyed when extending.

### A. EXTENSION OF HAS

In LoET, the definition of *has* only appears in the flow relation, and is not comprehensive, it just describes the ownership of the overall *Atom* in a single operation which is shown in (26). In the following formula, it can be clearly seen that the definition of *has* is simply a description of the seven actions, and does not make a detailed distinction between the contents of the information.

$$\left(\begin{array}{c} e\ has\ a \equiv_{def} \\ (e \in E(New) \wedge New(e)\ has\ a) \\ \vee(e \in E(Send) \wedge Send(e)\ has\ a) \\ \vee(e \in E(Receive) \wedge Receive(e)\ has\ a) \\ \vee(e \in E(Encrypt) \wedge Encrypt(e)\ has\ a) \\ \vee(e \in E(Decrypt) \wedge Decrypt(e)\ has\ a) \\ \vee(e \in E(Sign) \wedge Sign(e)\ has\ a) \\ \vee(e \in E(Verify) \wedge Verify(e)\ has\ a) \end{array}\right) \tag{26}$$

But in PCL, predicate *Has* is used to model knowledge, so *Has* not only has a complete set of definitions in (17) but also has many corresponding axioms and rules that can be seen in (24). According to the definition of *Has* in PCL, We can add some rules to Has in LoET-E. As shown in (27).

$$\left(\begin{array}{c} RuleH1: \\ \forall event \in (E(New) \vee E(Send) \\ \vee E(Receive) \vee E(Encrypt) \\ \vee E(Decrypt) \\ \vee E(Sign) \vee E(Verify)). \\ ((e\ has\ a) \wedge (e\ has\ b) \Rightarrow e\ has\ (a,b)) \\ \vee(e\ has\ (a,b) \Rightarrow (e\ has\ a) \wedge (e\ has\ b)) \\ RuleH2: \\ \forall event: E(Encrypt) \vee E(Decrypt). \\ ((e\ has\ plaintext(e)) \wedge (e\ has\ K) \\ \Rightarrow e\ has\ ciphertext(e)) \\ \vee((e\ has\ ciphertext(e)) \\ \wedge(e\ has\ K) \Rightarrow e\ has\ plaintext(e)) \end{array}\right) \tag{27}$$

In Rule H1, the splitting and combination of groups are added on the basis of the original concept of *has*. Information items are defined in this rule, in other words, the concept of the tuple with information added. This rule aims to split and reconstruct the information itself, which provides great convenience in the process of sending and receiving information.

Rule H2 allows encryption and decryption of acquired information when the key is already held, making the information already available clearer.

With the enriched concept of *has* in LoET-E, not only the causal relationship between the nonce events in the flow relation can be verified more effectively, but also ensures the correctness and rigor of the proof process.

### B. EXTENSION OF HONEST PRINCIPALS

There is only one axiom in the description of honest principals' behavior in LoET, which is honesty axiom in (8). In this axiom, the hypothesis of an honest principals is described as the following behaviors: the private keys of the honest principals will not be released, and the sign events, encrypt events, and decrypt events associated with the private keys must occur on the honest principals.

However, just defining these is not enough, the honest principal executes the threads in the protocol in strict order, following the basic sequence of the protocol. The original axiom only considers the principal of the signature, and there must be a corresponding verify event for the sign event in the basic sequence, so the source of the signature in the verification step is ignored. And it needs to be extended in more detail in LoET-E.

$$\begin{array}{c} AxiomS: \forall A: Id. \forall s: E(Sign). \\ \forall e: E(Encrypt). \\ \forall d: E(Decrypt). \\ \forall v: E(Verify). Honest(A) \Rightarrow \\ \left\{\begin{array}{c} signer(s) = A \Rightarrow (loc(s) = A) \\ \wedge key(v) = Key(A) \wedge A \neq loc(v) \\ \Rightarrow ((loc(s) = A) \\ \wedge(\exists e: E(Send). loc(e) = A \\ \wedge ehassignature(e))) \\ \wedge key(e) = PrivateKey(A) \\ \Rightarrow (loc(e) = A) \\ \wedge key(d) = PrivateKey(A) \\ \Rightarrow (loc(d) = A) \end{array}\right\} \end{array} \tag{28}$$

As shown in (28), on the basis of the original honesty and axiom, the judgment of the source of the signature of the honest principal is added. By default, if the key used in the verify event is the public key of the honest principal $A$, then it can be determined that the owner of the current signature is the honest principal $A$, and there exist a send event in which the sender is $A$, and the information of the send event has the digital signature of honest principal $A$.

In addition, in the basic sequence, if an honest principal has some information in the preorder, then we believe that in the subsequent behavior, the honest principal always has this information and vice versa. For example, if in a protocol, principal $A$ receives the encrypted message containing $(a, b)$,

then $A$ needs to encrypt $b$ and sends it again. Assuming that principal $A$ is honest, and $A$ sends the encrypted information containing $b$, it can be inferred that $A$ has received the encrypted information containing $(a, b)$, so honest rules are introduced.

This part redefines the ownership of the private key and signature, extend the honesty axiom that describe the behavior of honest principals, reduce the complexity and redundancy in the protocol analysis process.

### C. INTRODUCTION OF FRESH AND EXTENSION OF RELATED RULES

In LoET, there is no definition of the freshness of nonces, but the flow relation and the Nonce axiom are used to prove the event orderings. The process of proof is cumbersome and the consistency of event classes and basic sequences is difficult to define. So in LoET-E, we need to introduce the concept of freshness and a series of related axioms to simplify the proof process and ensure the correctness of the proof process.

#### 1) FRESH OF NONCES

First, we need to introduce the concept of *Fresh*, suppose there is an *Atom* $a$, *Fresh* means that no principal other than the principal itself sees $a$ or any messages containing $a$, and in LoET, it will not be introduced in the form of seven event classes, but define its type as Boolean. *Fresh* can be used to refer to any information, but here we only use *Fresh* to describe nonces and judge the freshness of nonces. By the way, when the event does not own $a$, it also does not have freshness of $a$. And we can give the definition of Fresh in (29)

$$\left(\begin{array}{c} Definition\ of\ Fresh: \\ New(e_0) = a \Rightarrow Fresh(e_0, a) \\ \vee\forall A : Id.\exists e_1 : E(New).\forall e_2 : E_1.\exists e_3 : E_2 \\ loc(e_1) = loc(e_2) = loc(e_3) = A \\ \wedge e_1 < e_2 < e_3 \wedge E_2(e_3)\ has\ a \\ \wedge(\neg(Send(e_2)\ has\ a) \\ \vee\neg(e_2 = E(Send)) \\ \Rightarrow Fresh(e_3, a) \end{array}\right) \tag{29}$$

This definition is based on the rules of *has*, which states that if the event $e_0$ generates the nonce $a$, this event retains the freshness of the nonce, or if all the send events $e_2$ do not contain nonce after the occurrence of the new event $e_1$ and before the event $e_3$ containing the nonce, then we believe that the $e_3$ retains the freshness of the nonce.

After defining *Fresh*, two concepts *Gen* and *FirstSend* need to be introduced. *Gen* means that a principal generates a message, similar to *Fresh*, the message mentioned here usually refers to nonces and its type is also assigned as a Boolean value. We can use *Gen* to judge the generator of the nonce, and *Gen* can be associated with *Fresh*. For example, when a principal $A$ retains the freshness of the nonce, it can be determined that the nonce is generated by the principal $A$. A detailed definition of *Gen* is given in (30).

$$\left(\begin{array}{c} Definition\ of\ Gen: \\ if\ e : E(New) \Rightarrow Gen(e, a) \end{array}\right) \tag{30}$$

In (31), a definition about *Gen* is given, the freshness of nonces is used to determine whether the nonce is generated by the current principal. If there is an event $e : E(New)$, then $Gen(e, a)$ can be used directly to indicate that $a$ is generated on event $e$.

*FirstSend* means that the principal sends a message for the first time, and the message has not been sent by this principal or other principals before. Same as the above two, although the message mentioned above can represent any message actually, but it is used to represent nonce here. *FirstSend* can be used to determine if a nonce is sent for the first time. Similarly, the type of *FirstSend* is also a Boolean value, and it can also be associated with *Fresh* and *Gen*. For example, if a principal $A$ sends a nonce for the first time, it can be judged that the nonce is fresh and is generated by principal $A$. As shown in (31), it is the definition formula of *Firstsend*.

$$\left(\begin{array}{c} Definition\ of\ FirstSend: \\ \forall A : Id.\exists e_1 : E(New).\exists e_2 : E(Send). \\ loc(e_1) = loc(e_2) = A \wedge e_1 < e_2 \\ \wedge Fresh(e_2, a) \\ \wedge\neg(\exists Send(e)\ has\ a \wedge e_1 < e < e_2) \\ \Rightarrow FirstSend(e_2, a) \end{array}\right) \tag{31}$$

This formula in (31) means that after the principal $A$ generates a nonce, if one of the subsequent send event $e_2$ retains the fresh value of the nonce, then it can be determined that there is no other send event containing nonce during the period after $e_1$ occurs until $e_2$ occurs. And based on this definition, we can extend the previous *has* rules again.

$$\left(\begin{array}{c} RuleH3: \\ \forall A : Id.\exists e_1 : E(New).\exists e_2 : E(Send) \\ loc(e_1) = loc(e_2) = A\wedge \\ e_1 < e_2 \wedge \neg FirstSend(e_2, a) \\ \Rightarrow \exists B : Id.\exists e : E(Rcv).e\ has\ New(e_1) \end{array}\right) \tag{32}$$

In (32), the third rule about *has* is given. And the meaning is when a principal $A$ generates a nonce and finds that there is a send event on principal $A$ after the new event, and the send event is judged not the first time to send, it can be inferred that there is a principal $B$, and the information in one of his receive events exists a nonce generated by $A$.

Then the connection between *Fresh* and *Gen* can be defined in the language of LoET, as shown in (33), a rule about *Fresh* is given.

$$\left(\begin{array}{c} RuleF1: \\ \forall A : Id.\forall e_1 \notin E(New). \\ loc(e_1) = A \wedge Fresh(e_1, a) \\ \Rightarrow \exists e_2 : E(New). \\ e_2 < e_1 \wedge Gen(e_2, a) \end{array}\right) \tag{33}$$

In (33), a rule where *Fresh* and *Gen* linked together is given. If there is an event on principal $A$ that is not $New(a)$, and this event has freshness of nonce, then it can be inferred that there must exist a new event before this event to generate nonce $a$. Because $Gen(e, a)$ can be directly inferred from $New(a)$, so *New* is excluded from the rule RuleF1.

Next, start making rules based on the relationship between *FreshSend* and *Fresh*, as shown in the following formula (34).

$$\left(\begin{array}{c} RuleF2: \\ \forall A: Id.\exists e_1: E(New).\exists e_2: E(Send) \\ loc(e_1) = loc(e_2) = A \\ \wedge e_1 < e_2 \wedge \neg Fresh(e_2, a) \\ \Rightarrow \exists e_3: E(Send).loc(e_3) = A \\ \wedge e_1 < e_3 < e_2 \\ \wedge e_3\ has\ a \\ \wedge\neg(\exists e_4: E(Send).loc(e_4) = A \\ \wedge e_1 < e_4 < e_3 \wedge Fresh(e_4, a)) \\ \Rightarrow TFirstSend(e_3, a) \end{array}\right) \tag{34}$$

This rule RuleF2 means that after the principal $A$ generates a nonce, if one of the subsequent send event $e_2$ is not send $a$ for the first time (here does not make assumptions about whether $e_2$ has $a$, because the results are the same.), it can be judged that there is a send event $e_3$ on the principal $A$ to send the nonce for the first time, where $e_3$ occurs not only before $e_2$ but also after the event $e_1$ that generated the nonce. Moreover, during this period, there are no other send event containing freshness of the nonce, that is, the above mentioned *FirstSend*.

This part introduces the concept of *Fresh*, and derives two definitions of *Gen* and *FirstSend* based on the *Fresh* into LoET, presents two rules related to this three new concepts. In addition, the concept of *Fresh* is used to improve the rules for *has* in the previous section.

### 2) FRESH RELATED RULES

The first two rules related to *Fresh* have been presented in (33) and (34), and next we present the remaining rules.

First, we associate *Fresh* and *FirstSend* again to get the third rule, as shown in (35).

$$\left(\begin{array}{c} RuleF3: \\ \forall A: Id.\exists e_1: E(New). \\ \forall e_2: E(Send).\exists e_3: E(Send) \\ loc(e_1) = loc(e_2) = loc(e_3) = A \\ \wedge e_1 < e_2 < e_3 \\ \wedge Fresh(e_1, a) \wedge Send(e_2)||a \\ \wedge\neg Send(e_3)||a \\ \Rightarrow FirstSend(e_3, a) \end{array}\right) \tag{35}$$

This rule is similar to RuleF2, but is specified in two different directions. The rule in RuleF2 is that if there is no event containing freshness of nonce within the specified time period, then *FirstSend* is considered to be established. And in RuleF3, we take the independence of *Atom* as the point of penetration, and stipulates that if all the send events during this period before send event $e_3$ occurred and after new event $e_1$ occurred are independent of *Atom* $a$, then it is considered that $e_2$ is the first time to send *Atom* $a$.

Then we can infer the order between the events based on the above rules, specifically as shown in the formula (36).

$$\left(\begin{array}{c} RuleF4: \\ \forall A, B: Id.\exists e_1: E(Send).\forall e_2 = E. \\ loc(e_1) = A \wedge loc(e_2) = B \\ \wedge TFirstSend(e_1, a) \wedge \neg E(e_2)||a \\ \Rightarrow e_1 < e_2 \end{array}\right) \tag{36}$$

In this rule, the event orderings of two principals is specified. Suppose there is a send event on the principal $A$, and there is an arbitrary event on the principal $B$. If the *Atom* $a$ in the send event on principal $A$ is determined to be the first send, and the event on principal $B$ is not independent of *Atom* $a$, then inferring that the send event on principal $A$ occurs before the event on principal $B$ occurs. According to this rule, the process of reasoning strong matching sessions will become easier.

Finally, the two most important rules are presented. These two rules are used to justify the continuity of Boolean under different decision formulas.

The first one of the continuous rules is shown in (37), which stipulates the continuity of Boolean in *Fresh*.

$$\left(\begin{array}{c} RuleF5: \\ \forall A: Id.\exists e_1, e_3: E.\forall e_2 = E. \\ loc(e_1) = loc(e_2) = loc(e_3) = A \\ \wedge e_1 < e_2 < e_3 \\ \wedge Fresh(e_1, a) \wedge E(e_2)||a \\ \Rightarrow TFresh(e_3) \end{array}\right) \tag{37}$$

As is expressed in the (37), this formula specifies the consistency of the fresh value after an event. Suppose there are events $e_1$ and $e_3$ on principal $A$, for any event $e_2$ independent of *Atom* $a$ on principal $A$, if event $e_2$ occurs between event $e_1$ and event $e_3$, and event $e_1$ has freshness of *Atom* $a$, then event $e_3$ is still retains the freshness of *Atom* $a$.

Last but the most important, the second one of the continuous rules is shown in (38), and in this last formula, we specify the continuity of all newly defined expressions.

$$\begin{array}{c} RuleF6: \\ \left\{\begin{array}{c} \forall A: Id.\exists e_1: E.\forall e_2 = E. \\ \left(\begin{array}{c} loc(e_1) = loc(e_2) = A \\ \wedge FirstSend(e_1, a) \\ \wedge E(e_2) \wedge e_1 < e_2 \\ \Rightarrow TFreshSend(e_1, a) \end{array}\right) \\ \wedge \left(\begin{array}{c} loc(e_1) = loc(e_2) = A \\ \wedge Gen(e_1, a) \wedge E(e_2) \\ \wedge e_1 < e_2 \\ \Rightarrow TGen(e_1, a) \end{array}\right) \\ \wedge \left(\begin{array}{c} loc(e_1) = loc(e_2) = A \\ wedgee_1 has a \wedge E(e_2) \wedge e_1 < e_2 \\ \Rightarrow T(e_1 has a) \end{array}\right) \end{array}\right\} \end{array} \tag{38}$$

As shown in (38), all new definitions have been added to this rule. The first is about the continuity of *FreshSend*. Suppose there is event e1 on principal $A$, for any event $e_2$ on principal $A$, after the event $e_2$ occurs, the judgment of whether event $e_1$ is the first to send $a$ is still true. This means that no matter what happens after the event $e_1$, the Boolean value of *FirstSend* will not change on event $e_1$. Similarly, no matter what happens after the event $e_1$, the Boolean value of *Gen* and has will not change on event $e_1$. The continuous rules give the rules for the judgment of the continuity of the new definitions, further refining the new definition and ensuring the correctness, rigor and integrity of the subsequent proofs.

This part introduces the novel rules about the above three new concepts, associate three new concepts with independence of *Atom*, event orderings and other original concepts of LoET. And give enough rules to incorporate new concepts into LoET, thus the perfection of LoET-E can be shown more clearly.

## V. PROOF OF PROTOCOL

In this section, a simple example is used to verify the usability of LoET-E and to show a more concise proof process than LoET. Because this paper mainly focuses on the introduction of LoET-E, so next we will use LoET-E to formally prove the security properties of new cryptographic protocols. PUFs is a new hardware security primitive, and the research on PUFs is one of the emerging research focuses. For PUFs-based two-way authentication protocols [19], we can prove its security properties by using LoET-E.

First, we need to abstract the functions implemented by PUF. The realization of PUF security is mainly depends on the response of PUF and the nonces encoded by XOR, and then through the response of the authenticated party and the number of nonces encoded by XOR, the coded nonces is decoded and XOR is restored to the response of the authenticated party, and the second key is obtained.

The above functions are abstracted by using LoET-E, as shown in the following formula.

$$\begin{pmatrix} New(nonce_1), \\ Encrypt(<< K' >, K >), \\ Send(< B, nonce_1, << K' >, K >>), \\ Rcv(< B, nonce_1, << K' >, K >>), \\ Decrypt(<< K' >, K) \\ New(nonce2) \\ Encrypt(<< nonce_2 || nonce_1 >, K' >) \\ Send(<< nonce_2 || nonce_1 >, K' >) \\ Rcv(<< nonce_2 || nonce_1 >, K' >) \\ Decrypt(<< nonce_2 || nonce_1 >, K' >) \end{pmatrix}$$

By abstracting the functions of PUF, we can use LoET-E to describe the two-way authentication protocol based on PUF according to the abstracted functions. And we should describe this authentication protocol, and define $I_1, I_2, I_3, I_4, I_5$ as the basic sequences for initiator, $R_1, R_2, R_3, R_4$ are defined as the basic sequences for responder. The protocol description is shown in Figure 2.

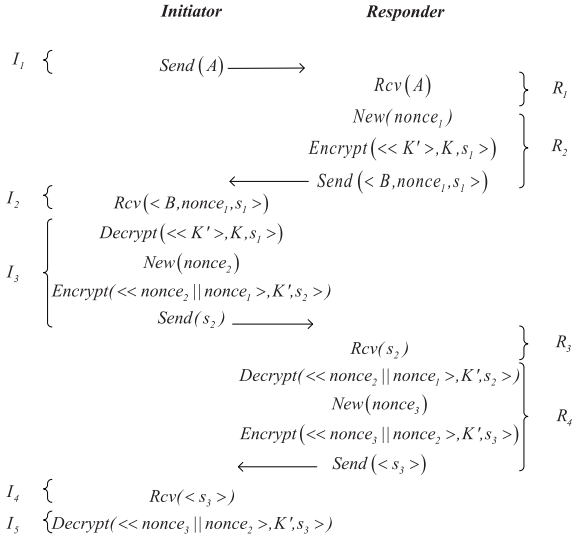



**FIGURE 2.** Description of interactive information in CR protocol.

$$\begin{aligned} &I_1 = Send(A) \\ &I_2 = Rcv(< B, nonce_1, s_1 >) \\ &I_3 = Decrypt(<< K' >, K, s_1 >), New(nonce_2), \\ &Encrypt(<< nonce_2 || nonce_1 >, K', s_2 >), \\ &Send(s_2) \\ &I_4 = Rcv(< s_3 >) \\ &I_5 = Decrypt(<< nonce_3 || nonce_2 >, K', s_3 >) \\ &R_1 = Rcv(A) \\ &R_2 = New(nonce_1), Encrypt(<< K' >, K, s_1 >) \\ &Send(< B, nonce_1, s_1 >), \\ &R_3 = Rcv(s_2) \\ &R_4 = Decrypt(<< nonce_2 || nonce1 >, K', s_2 >), \\ &New(nonce_3), \\ &Encrypt(<< nonce_3 || nonce_2 >, K', s_3 >), \\ &Send(< s_3 >) \end{aligned}$$

**FIGURE 3.** The basic sequence of protocol.

Then, this protocol is sorted through the concept of basic sequence in LoET-E, as shown in Figure 3.

By analyzing the basic sequences, this protocol is defined as $Protocol([I_1, I_2, I_3, I_4, I_5, R_1, R_2, R_3, R_4])$. It is known that the strong authentication properties that need to be verified in the proof of this protocol is $Nse \models auth(I_5, 4) \wedge Nse \models auth(R_4, 3)$. Here we only certify the authentication property for the initiator in this protocol, that is $Nse \models auth(I_5, 4)$.

First, we should prove this formula. Suppose *Initiator* $\neq$ *Responder* (they will be replaced by $A$ and $B$ later), and both principals follow this protocol. And according to the definition of the basic sequence, each thread is an instance of the basic sequence. Let $e_0 <_{loc} e_1 <_{loc} \ldots <_{loc} e_6$ be the events in $thr_1$, Because the principal of $thr_1$ is $A$, so the principal of the event is also $A$, and for some atoms $s_1, s_2, s_3, K', K, nonce_1, nonce_2, nonce_3$ have:

$$\left(\begin{array}{c} Rcv(e_0) =< B, nonce_1, s_1 \\ > \wedge Decrypt(e_1) =<< K' >, K, s_1 \\ > \wedge New(e_2) =< nonce_2 \\ > \wedge Encrypt(e_3) =<< nonce_2 || nonce_1 >, K', s_2 \\ > \wedge Send(e_4) =< s_2 \\ > \wedge Rcv(e_5) =< s_3 \\ > \wedge Decrypt(e_6) =<< nonce_3 || nonce_2 >, K', s_3 > \end{array}\right)$$

By AxiomV and the new AxiomsS, there is an event $e'$ such that:

$$e' < e_6 \wedge DEMatch(e_1, e') \wedge loc\left(e'\right) = B \vee loc\left(e'\right) = A$$

Because $B$ obeys the protocol, action $e'$ must be a member of the basic sequences of protocol. In the basic sequences, $I_2, I_3, I_4, I_5, R_2, R_3, R_4$ containing encrypt event, and in order to ensure the effectiveness of the event, the event class in which the initiator $A$ and responder $B$ participate must be bilateral or multiparty, as shown in the following formula (39).

$$\left(\begin{array}{c} \forall A, B : (A \neq B). \\ \forall e_1, e_2 : ((e_1 \in A, e_2 \in B) \wedge (e_1 < e_2)) \\ \vee (Send(e_1) = Rcv(e_2) \\ \vee (Sign(e_1) = Verify(e_2) \\ \vee (Decrypt(e_1) = Encrypt(e_2) \end{array}\right) \tag{39}$$

Therefore, $I_2, I_3, I_4, I_5$ can be ruled out, which means that $R_2, R_3, R_4$ have a basic sequence that may constitute a matching session with $I_5$.

Assuming $e'$ is an instance of $R_2$, and for some atoms $nonce_1', K_1', K_2, s_1'$, and location $C$, there are events $e_0', e_1', e_2', e_3'$ at location $B$ such that:

$$\left(\begin{array}{c} e_0' <_{loc} e_1' <_{loc} e_2' <_{loc} e_3' \\ \wedge Rcv(e_0') =< C \\ > \wedge New(e_1') =< nonce_1' \\ > \wedge Encrypt(e_2') =<< K_2' >, K_1', s_1' \\ > \wedge Send(e_3') =< B, nonce_1', s_1' > \end{array}\right)$$

Through the above formula, we can find that the encrypted event $e_2'$ in $R_2$ does not match the decrypted event $e_6$, so the possibility that $e'$ is an instance of $R_2$ is excluded, and similarly, $R_3$ can be excluded.

Assuming $e'$ is an instance of $R_4$, and for some atoms $s_2', s_3', nonce_1', nonce_2', nonce_3', K_2'$, and location $D$, there are events $e_0', e_1', e_2', e_3', e_4', e_5', e_6', e_7', e_8'$ at location $B$ such that:

$$\left(\begin{array}{c} e_0' <_{loc} e_1' <_{loc} e_2' <_{loc} e_3' <_{loc} e_4' <_{loc} \\ e_5' <_{loc} e_6' <_{loc} e_7' <_{loc} e_8' \\ \wedge Rcv(e_0') =< D > \wedge New(e_1') =< nonce_1' \\ > \wedge Encrypt(e_2') =<< K_2' >, K_1', s_1' \\ > \wedge Send(e_3') =< B, nonce_1', s_1' \\ > \wedge Rcv(e_4') =< s_2' \\ > \wedge Decrypt(e_5') =<< nonce_2' || nonce_1' >, K_2', s_2' \\ > \wedge New(e_6') =< nonce_3' \\ > \wedge Encrypt(e_7') =<< nonce_3' || nonce_2' >, K_2', s_3' \\ > \wedge Send(e_8') =< s_3' > \end{array}\right)$$

Through the above formula, the following can be found:

$$\left(\begin{array}{c} Encrypt(e') =<< nonce_3 || nonce_2 >, K', s_3 > \\ =<< nonce_3' || nonce_2' >, K_2', s_3' >= Encrypt(e_7') \end{array}\right)$$

Then we can see that there are $e_7' = e', D = A, nonce_3' = nonce_3, K_2' = K, nonce_2' = nonce_2, s_3' = s$, and we can draw:

$$\left(\begin{array}{c} e_0' <_{loc} e_1' <_{loc} e_2' <_{loc} e_3' <_{loc} e_4' <_{loc} \\ e_5' <_{loc} e_6' <_{loc} e_7' <_{loc} e_8' \\ \wedge Rcv(e_0') =< A > \wedge New(e_1') =< nonce_1 \\ > \wedge Encrypt(e_2') =<< K' >, K, s_1 \\ > \wedge Send(e_3') =< B, nonce_1, s_1 \\ > \wedge Rcv(e_4') =< s_2 \\ > \wedge Decrypt(e_5') =<< nonce_2 || nonce_1 >, K', s_2 \\ > \wedge New(e_6') =< nonce_3 \\ > \wedge Encrypt(e_7') =<< nonce_3 || nonce_2 >, K', s_3 \\ > \wedge Send(e_8') =< s_3 > \end{array}\right)$$

Similarly, we can prove that there is an event $e''$ as an instance of $I_3$ satisfies:

$$e'' < e_5' \wedge DEMatch(e_5', e'') \wedge loc\left(e''\right) = A \vee loc\left(e''\right) = B$$

And an event $e'''$ as an instance of $R_2$ satisfies:

$$e''' < e_3'' \wedge DEMatch(e_3'', e''') \wedge loc\left(e'''\right) = A \vee loc\left(e'''\right) = B$$

According to the above results, we can get:

$$\left(\begin{array}{c} Rcv(e_0) =< B, nonce_1, s_1 >= Send(e_3') \\ \wedge Send(e_4) =< s_2 >= Rcv(e_4') \\ \wedge Rcv(e_5) =< s_3 >= Send(e_8') \end{array}\right)$$

Now we have a weak matching conversation, then we need to assert temporal ordering between those two principals so that we can get a strong matching conversation, $e_3' < e_0,\ e_4 < e_4'$ *and* $e_8' < e_5$.

According to the definition of *Fresh*, the following formula can be obtained.

$$\left(\begin{array}{c} New(e_1') =< nonce_1 \\ >\Rightarrow Fresh(e_1', nonce_1) \end{array}\right) \wedge \left(\begin{array}{c} A : Id. \forall e_2 : E. \\ loc(e_1') = loc(e_2') = loc(e_3') = B \\ \wedge e_1' < e_2' < e_3' \wedge E(e_3') has nonce_1 \\ \wedge \neg(e_2' = E(Send)) \end{array}\right) \Rightarrow Fresh(e_3', nonce_1)$$

Then it can be proved that $nonce_1$ was sent for the first time in the event $e_3{}'$.

$$\left(\begin{array}{c} loc(e_1{}') = loc(e_3{}') = A \wedge \\ e_1 < e_3{}' \wedge Fresh(e_3{}', nonce_1) \wedge \\ \neg(\exists Send(e) has nonce_1 \wedge \\ e_1 < e < e_2) \end{array}\right) \Rightarrow FirstSend(e_3{}', nonce_1)$$

So that $nonce_1$ was sent for the first time in event $e_3{}'$, and all actions that contain $nonce_1$ must happen after event $e_3{}'$, including the event $Rcv(e_0) =< B, nonce_1, s_1 >$. Therefore, with the new honesty axiom, we have a new temporal ordering: $e_3{}' < e_0$.

Similarly, we can prove that $e_4 < e_4{}'$ and $e_8{}' < e_5$, and the strong authentication property obtained.

With the interaction of PUFs-based authentication protocol formally described by LoET-E, the basic sequences are constructed and the strong authentication property in protocol interaction process is verified.

If LoET is used to prove this new type protocol, since LoET does not have the judgment of the freshness of the challenge number, it cannot get results quickly and effectively in the process of proving strong authentication property.

## VI. FUTURE WORK

Although this paper present LoET-E successfully, there are still some follow-up work to be completed.

(1) LoET-E is currently only tested on some classic cryptographic protocols and small areas of the emerging cryptographic protocols, it has not been proven on most of the emerging cryptographic protocols. Subsequent work includes the use of LoET-E for instance analysis of more new cryptographic protocols.

(2) At present, both LoET-E and PCL can only prove the authentication attribute of cryptographic protocols, and cannot prove other attributes of cryptographic protocols. We can introduce new concepts and ideas to prove different security properties by learning how other formal methods can prove security properties. For example, a new formal method called Witness-Functions [20], [21] has recently emerged. When using this logic to prove the secrecy of protocols, the concept of tag is used [22], which we think is valuable for reference. From this perspective, the extended logic of event can be further studied to prove more properties of cryptographic protocols.

## VII. CONCLUSIONS

Based on the logical framework of PCL, this paper extends LoET from several different aspects and presents LoET-E. The main work is as follows:

(1) Combine the Possession Axioms to extend the definition of *has* in the flow relation axiom of LoET, ensuring the correctness, integrity and validity of the original axioms.

(2) Redefine the source of the signature accepted by the honest principal in LoET, take into account all possible situations in a formulaic manner, and extend the honesty axiom that describe the behavior of honest principals.

(3) Introduce the concept of *Fresh* and two derivative concepts *Gen*, *FirstSend* in LoET, ensuring the consistency of event classes and basic sequences in the proof process.

(4) According to these three new concepts, a series of important rules are presented to reduce the complexity and redundancy in the protocol analysis process and expand the scope of proof of the cryptographic protocols.

(5) By presenting LoET-E, the provable range of cryptographic protocols was extended.

## REFERENCES

[1] S. F. Allen *et al.*, "Innovations in computational type theory using Nuprl," *J. Appl. Logic*, vol. 4, no. 4, pp. 428–469, 2006.
[2] M. Bickford and R. L. Constable, "A logic of events," Dept. Comput. Sci., Cornell Univ., Ithaca, NY, USA, Tech. Rep., 2003.
[3] M. Xiao and M. Bickford, "Logic of events for proving security properties of protocols," in *Proc. IEEE Int. Conf. Web Inf. Syst. Mining (WISM)*, Nov. 2009, pp. 519–523.
[4] M. Bickford, "Unguessable atoms: A logical foundation for security," in *Proc. Work. Conf. Verified Softw., Theories, Tools, Exp.* Berlin, Germany: Springer, 2008, pp. 30–53.
[5] M. Bickford and R. Constable, "Formal foundations of computer security," *Nato Secur. Through Sci. D-Inf. Commun. Secur.*, vol. 14, pp. 29–52, Jan. 2008.
[6] M. Xiao, C. Ma, C. Deng, and K. Zhu, "A novel approach to automatic security protocol analysis based on authentication event logic," *Chin. J. Electron.*, vol. 24, no. 1, pp. 187–192, 2015.
[7] A. Datta, A. Derek, J. C. Mitchell, and A. Roy, "Protocol composition logic (PCL)," *Elect. Notes Theor. Comput. Sci.*, vol. 172, pp. 311–358, Apr. 2007.
[8] C. Cremers, "On the protocol composition logic PCL," in *Proc. ACM Symp. Inf., Comput. Commun. Secur.*, 2008, pp. 66–76.
[9] C. He, M. Sundararajan, A. Datta, A. Derek, and J. C. Mitchell, "A modular correctness proof of IEEE 802.11i and TLS," in *Proc. 12th ACM Conf. Comput. Commun. Secur.*, 2005, pp. 2–15.
[10] J. Szefer, T. Zhang, and R. B. Lee. (2018). "Practical and scalable security verification of secure architectures." [Online]. Available: https://arxiv.org/abs/1807.01854
[11] S. Biju and N. M. Shekokar, "Security approach on MQTT based smart home," in *Proc. IEEE Int. Conf. Power, Control, Signals Instrum. Eng. (ICPCSI)*, Sep. 2017, pp. 1106–1114.
[12] T. Feng, S. Han, X. Guo, and D. Ma, "A new method of formalizing anonymity based on protocol composition logic," *Secur. Commun. Netw.*, vol. 8, no. 6, pp. 1132–1140, 2015.
[13] M. Bickford and R. L. Constable, "Automated proof of authentication protocols in a logic of events," in *Proc. VERIFY IJCAR*, 2010, pp. 13–30.
[14] L. Yanan, X. Meihua, L. Wei, M. Yingtian, and Z. Xiaomei, "Security proof of wireless mesh network authentication protocol based on event logic," *Comput. Eng. Sci.*, vol. 39, no. 12, pp. 2236–2244, 2017.
[15] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, 1978.
[16] M. Bickford, "Component specification using event classes," in *Proc. Int. Symp. Compon.-Based Softw. Eng.* Berlin, Germany: Springer, 2009, pp. 140–155.
[17] L. Xinqian, X. Meihua, C. Daolei, M. Yingtian, and L. Wei, "Improvement of the safety of the needham-schroeder protocol based on event logic," *Comput. Eng. Sci.*, vol. 37, no. 10, pp. 1850–1855, 2015.
[18] X. Meihua, L. Xinqian, L. Yanan, C. Daolei, and M. Yingtian, "Proof of three-party network protocol security based on strong authentication theory," *Comput. Sci. Explor.*, vol. 10, no. 12, pp. 1701–1710, 2016.
[19] L. Dan, G. Limin, Y. Jun, Lihui, and S. Weijun, "A secure two-way authentication protocol based on SRAM PUF," *J. Cryptogr.*, vol. 4, no. 4, pp. 360–371, 2017.
[20] J. Fattahi, M. Mejri, and E. Pricop, "The theory of witness-functions," in *Recent Advances in Systems Safety and Security*. Cham, Switzerland: Springer, 2016.
[21] J. Fattahi, "Secrecy by witness functions," in *Proc. FMS@ Petri Nets*, 2014, pp. 34–52.

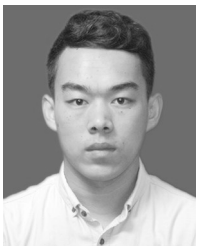
[22] J. Fattahi, "A theorem for secrecy in tagged protocols using the theory of witness-functions," in *Proc. IEEE Can. Conf. Elect. Comput. Eng. (CCECE)*, May 2018, pp. 1–6.

**JIAWEN SONG** received the B.S. degree in network engineering from Tongda College, Nanjing University of Posts and Telecommunications, in 2017. She is currently pursuing the M.S. degree with the School of Software, East China Jiaotong University. Her current research interests include information security and formal methods.

**MEIHUA XIAO** received the B.S. degree from the Department of Computer Engineering, University of Shanghai for Science and Technology (USST), in 1987, the M.S. degree in computer software and theory from Nanchang University, in 2001, and the Ph.D. degree in computer software and theory from the Institute of Software, Chinese Academy of Science, China, in 2007.

From 2008 to 2009, he was a Visiting Scholar with Cornell University. He is currently a Professor and the Dean of the School of Software, East China Jiaotong University. His research interests include formal methods and information security. Until now, he is the Director of the National College of Computer Basic Education Research Association, a member of the Chinese Computer Society Formal Methods Committee, a member of the China Electronics Association Information Theory Branch, and the Executive Director of the Jiangxi Computer Society.

**KE YANG** received the B.S. degree in electronic science and technology from the East China Institute of Technology, in 2016. He is currently pursuing the M.S. degree with the School of Software, East China Jiaotong University. His current research interests include information security and formal methods.



**XIZHONG WANG** received the B.S. degree in computer science and technology from the Hunan University of Science and Engineering, in 2017. He is currently pursuing the M.S. degree with the School of Software, East China Jiaotong University. His current research interests include information security and model checking.

**XIAOMEI ZHONG** received the B.S. degree in computational science and technology from the East China Institute of Technology, Jiangxi, China, in 2001, and the M.S. degree in earth exploration and information technology from the East China Institute of Technology, Jiangxi, China, in 2004. She is currently pursuing the Ph.D. degree in control science with East China Jiaotong University, Jiangxi, China.

Since 2004, she has been a Lecturer with the School of Software, East China Jiaotong University, Jiangxi, China. Her research interests include information security and formal methods.