Soft Computing (2022) 26:841–852
https://doi.org/10.1007/s00500-021-06163-9

APPLICATION OF SOFT COMPUTING

Proving authentication property of PUF-based mutual authentication protocol based on logic of events

Jiawen Song[1] · Meihua Xiao[1] · Tong Zhang[1] · Haoyang Zhou[1]

Accepted: 9 July 2021 / Published online: 12 November 2021

Abstract
PUF (Physical unclonable function) is a new hardware security primitive, and the research on PUFs is one of the emerging research focuses. For PUF-based mutual authentication protocols, a method to abstract the security properties of hardware by using logic of events is proposed, and the application aspects of logic of events are extended to protocols based on hardware security. With the interaction of PUF-based mutual authentication protocol formally described by logic of events, the basic sequences are constructed and the strong authentication property in protocol interaction process is verified. Based on the logic of events, the freshness of nonces is defined, and the persist rule is proposed according to the concept of freshness, which ensures the consistency of the protocol state and behavior predicate in the proof process, and reduces the complexity and redundancy in the protocol analysis process. Under reasonable assumptions, the security of the protocol is proven, and the fact that logic of events applies to PUF-based mutual authentication protocols is shown.

Keywords Cryptographic protocols · Logic of events · Physical Unclonable Function · Theorem proving.

1 Introduction

The security of cryptographic protocols plays an important role in the field of information security, especially in the aspects of device identification, identity authentication, key generation and storage, which should be considered when choosing appropriate cryptographic protocols. In recent years, with the rapid development of information technology, mobile devices and embedded devices have begun to be widely used. However, there are some problems in intelligent mobile devices, such as poor computing power and resource constraints. These constraints lead to great obstacles in the application of traditional cryptographic protocols, which are almost impossible to achieve (Herder 2014). Therefore, in order to ensure the security of equipment, it is necessary to modify the implementation of traditional cryptographic protocols or use new mechanisms.

PUF (Physical Unclonable Function) is a promising new hardware security primitive which can be understood as a new "digital fingerprint" (ONeill 2016). Due to its unique physical characteristics, it can ensure the security of devices under relatively low power consumption (Barbareschi et al. 2018). PUFs can distinguish different semiconductors according to their physical structure during semiconductor fabrication. When supplied with challenge, it will produce unique and unpredictable response according to the internal physical changes (Barbareschi et al. 2018). Because the uniqueness of physical changes within each chip, even if the same challenge is supplied, no two devices will generate the same response, which gives the uniqueness and non-clonability to the results of generation. So PUFs can be used as the unique identifier of the semiconductor device (Ruhrmair et al. 2009), and research on PUF-based cryptographic protocols is of great important in both theory and practical.

Formal methods mean a combination of a mathematical or logical model of a system and its requirements, and the system and conditions are analyzed by effective procedure to determine whether the system satisfies the constraints (Meadows 2003). It is a more rigorous method that is different from conventional security detection, providing a security means to detect the objects that need to be verified, and helping cryptographic protocols to truly realize the security properties they claim. Theorem proving is significant method in formal methods, which uses mathematical language to describe the system and its properties, and then deduces and proves the

✉ Jiawen Song
songjw1995@gmail.com

[1] East China Jiaotong University, Nanchang, China

properties according to axiomatic systems to ensure that the desired results can be achieved (Yang 2018).

In summary, the contributions of the specific works are as follows.

(1) The freshness of nonces based on event structure, event classes and axiom cluster is defined.
(2) The persist rule is proposed according to the concept of freshness, which can ensure the consistency of the protocol state and behavior predicate in the proof process, and reduce the complexity and redundancy in the protocol analysis process.
(3) Hardware security properties are abstracted by using logic of events.
(4) The interaction of PUF-based mutual authentication protocol is formally described, the basic sequences are constructed, and the strong authentication property in protocol interaction process is proved.
(5) The logic of events applied to PUF-based mutual authentication protocols is shown.

The rest of this paper is organized as follows: Sect. 2 introduces the related work on PUF-based mutual authentication protocols and Sect. 3 describes the structure of PUF-based mutual authentication protocol. In Sect. 4, the introduction of logic of events is given and the persist rule is proposed to ensure the consistency of the protocol state and behavior predicate in the proof process. Section 5 gives the security proof of PUF-based mutual authentication protocol in detail, and Sect. 6 summarizes this paper and gives a prospect for future work.

2 Related work

Because of the vulnerabilities of most cryptographic protocols based on traditional encryption and the need for plenty of computing resources, so that for smart mobile devices, choosing a cheap and secure way to authenticate devices has gradually become a research focus. And the PUF-based authentication protocols have been widely concerned due to its security and cost.

In 2002, the concept of PUF was first formally proposed by Pappu (2002), and they showed that although the cost of the physical one-way function is low, it is difficult to duplicate, which is intrinsically tamper resistant. In 2007, Suh and Devadas (2007) demonstrated the feasibility of using PUFs for low-cost identity authentication and key generation, and proposed the PUF-based authentication method for the first time. Later, many cryptographic protocols used this authentication method for identity authentication.

In 2011, Meguerdichian and Potkonjak (2011) enabled ultra-low power security protocols for trusted devices by integrating PUFs directly into sensor hardware, including authentication and public key communication. In the same year, Brzuska et al. (2011) proposed that PUFs could be added to UC (Universal Composition) framework to model tokens and derive schemes with strong security guarantees. In 2012, van Dijk and Ruhrmair (2012) proved the usability of PUFs in cryptographic protocols and showed that in order to make PUFs a broadly applicable cryptographic tool, new hardware attributes were required. The research and application of PUFs has gradually changed from the basic security tasks to more complex encryption protocols. In 2013, Lee et al. (2013) proposed a mutual authentication scheme for wireless body area network based on PUFs. In 2014, Wachsmann and Sadeghi (2014) explored the design of trusted embedded systems based on PUFs, and discussed how to integrate PUF into lightweight device authentication. Since 2014, more and more scholars have joined the design of PUF-based authentication protocols, combining PUF with authentication schemes in certain lightweight devices or smart IoT devices. In 2015, Delvaux (2015) published valuable research results on PUF-based protocols, which not only summarized the recent progress in PUF-based authentication protocols, but also pointed out that future research should focus on developing a PUF with truly powerful encryption properties. In 2019, Chatterjee et al. (2019) proposed a formal analysis tool for evaluation of PUFs by observing the correlation-spectra of the PUF instances under test. In 2020, Focardi and Luccio (2020) presented the first mechanized symbolic model for PUFs that allows for precisely reasoning about their security with respect to a variegate set of attackers. They formally proved security properties and specified the capabilities required by the attacker to break them.

The research on PUF-based authentication protocols has undergone numerous improvements and innovations since it was proposed. Whether it is a simpler authentication protocol based on PUF or a PUFs integrated cryptographic protocol based on UC framework, it reflects the development trend of PUF-based protocols in different periods. However, in the process of security analysis of PUF-based protocols, most people use cryptanalysis directly, and there are only a handful of cases using formal methods to analyze PUF-based protocols, most of which use BAN logic. Although the definition of BAN logic rules is relatively mature and widely studied, BAN logic relies on initial assumptions, which may lead to slight discrepancies between the analysis results and the original protocol design (Yanan 2018).

3 PUF-based authentication protocols

3.1 Authentication

Consider an honest agent A obeying a protocol. If A performs an instance of the full initiator sequence with parameter B,

then provided that B is honest and also obeys the protocol, there should be an instance of the responder sequence at B that forms a matching conversation for the first two messages in the protocol (we can not be sure that the third message, sent by A, will be received by B). Similarly, if B performs an instance of the full responder sequence, then there should be a matching conversation with A of length three. This authentication process can effectively prevent the third party from pretending to be any party of the communication parties from destroying the communication, that is, ensuring that the communication parties are honest subjects.

In the registration phase, the trusted third party with PUF devices stores the corresponding CRPs (challenge-response pairs) in a secure database by executing each device, and registers the IDs of all devices in the database for future verification operations. And in the authentication phase, when a trusted third party needs to authenticate the device, it selects the corresponding challenge from the database and remotely acquires the PUF response from the device. If the difference between the response provided by the device and the response recorded previously is limited to the pre-set threshold, the device will be authenticated successfully, and the CRP will delete from the database to prevent replay attacks.

3.2 PUF-based authentication method

The PUF-based authentication method was first proposed by Suh and Devadas (2007), which is intended to prove the feasibility of using PUF for low-cost authentication and cryptographic key generation. They provide a low-cost mutual authentication between principals, and this scheme can be divided into register phase and authentication phase.

Since most PUF-based authentication protocols are based on this authentication method proposed by Suh and Devadas (2007), this paper takes the authentication protocol based on this authentication framework as an example, to analyze the security of its verification phase. The principals of PUF-based mutual authentication protocols are divided into the authenticator and the authenticated party, in which the initiator holds the response of the PUF and the responder holds the equipment of the assembled PUF, and the default is that both principals hold the pre-agreed symmetric key. The interaction of the main authentication information is shown in Fig. 1.

The process of PUF-based mutual authentication protocols is as follows.

(1) First, the initiator sends an authentication request to the responder.
(2) The responder generates $nonce_1$ and obtains the PUF's response R', calculates the HD (helper data, generated by XOR of the coded $nonce$ and R) and encrypts it with the symmetric key K, then sends the ID of the responder, the encrypted ciphertext and $nonce_1$ to the initiator.

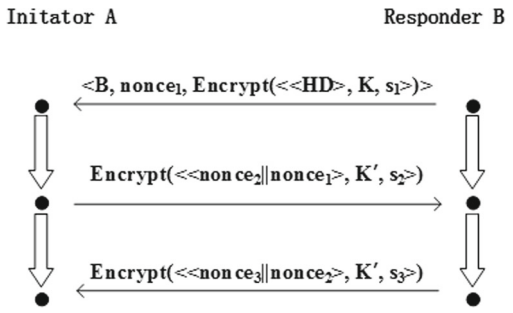

Fig. 1 Description of main interactive information in mutual authentication

(3) The initiator decrypts the ciphertext to get HD based on ID of the responder, and uses the previously agreed secret key K. Then uses R already owned to restore the current PUF's response R', and object on R' hash to get the secret key K'. After that, the initiator generates $nonce_2$, uses K' to encrypt the information $< nonce_2||nonce_1 >$ and sends the ciphertext to the responder.
(4) The responder calculates the hash value of R' and gets the secret key K', decrypts the received ciphertext and determines whether $nonce_1$ is the same as the one it sent before. If it is the same, the identity authentication succeeds; otherwise, the authentication fails.
(5) The responder generates $nonce_3$ and sends the information $< nonce_3||nonce_2 >$ to the initiator after encrypting with K'.
(6) The initiator uses K' to decrypt and determine whether $nonce_2$ is the same as the one it sent before. If it is the same, the identity authentication succeeds; otherwise, the authentication fails.

4 Logic of events

Logic of events (Bickford and Constable 2003; Xiao and Bickford 2009) is one of the formal methods, which describes the protocols and algorithm of distributed systems. The basic framework of logic of events mainly includes the theoretical basis, theoretical system (including logical axiom and inference) and formal description system (Bickford and Constable 2010; Meihua et al. 2015). Among them, the theoretical basis is used to model the initial encryption system. The theoretical system provides a set of complete inference system for the proof of cryptographic protocols. The formal description system describes and defines the thread sequence and matching session of the protocols.

4.1 Theoretical basis

The basic part of logic of events is the most basic theory in the process of proving the security properties of cryptographic

Table 1 BASIC SYMBOLS AND SEMANTICS

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 $Nonce$
$Nonce$	Challenge number set
$loc(e)$	A function on e, represents the principal of e
$key(e)$	The secret key of the principal of e
Has	Logical inclusion
bs	Basic sequences
$\leq$	Local finite partial order

protocols, which is used to model the initial encryption system. The basic symbols and semantics are given in Table 1.

In the formal analysis of cryptographic protocols, three types of values are needed to build the logic of events: B, Id and $Atom$ (Bickford 2008). B is used to judge whether a proposition is true or false, Id is used to distinguish the different principals in cryptographic protocols, and $Atom$ is an "unpredictable" data value. Generally, $Atom$ is used to represent anything that cannot be guessed, such as the nonce in cryptographic protocols, the ciphertext in the process of protocol transmission and the private key held by the principal itself.

Formula (1) is used to define the independence of the information, which assert that the information associated with event e contains $Atom\ a$.

$$\neg\,(info\,(e)\,||a) \tag{1}$$

In order to identify the point of information transmission, which is the event (Bickford and Constable 2010) in operation, we need to define the structure of the event sequence as formula (2).

$$< E, loc, <, \inf o > \tag{2}$$

By classifying different events in the protocol interaction process, the information associated with the events can be found, which means that the type of information depends on the event classes. Therefore, the events are defined and divided into seven categories: send, receive, new, encrypt, decrypt, sign and verify (Bickford and Constable 2010). The cryptographic protocols contain the messages of data tuples such as nonces, signatures and principals. We can define it as a binary tree data as shown in formula (3).

$$Data \equiv_{def} Tree\,(Id + Atom) \tag{3}$$

Formula (3) can represent all the information and plaintext we need; then, the protocol action type can be defined as (4).

$$\begin{pmatrix} \{\text{new (a)} \mid \text{a} \in \text{Atom}\} \\ \{\text{send (x)} \mid \text{x} \in \text{Data}\} \\ \{\text{rcv (x)} \mid \text{x} \in \text{Data}\} \\ \{\text{sign (t)} \mid \text{t} \in (\text{Data} \times \text{Id} \times \text{Atom})\} \\ \{\text{verify (t)} \mid \text{t} \in (\text{Data} \times \text{Id} \times \text{Atom})\} \\ \{\text{encrypt (t)} \mid \text{t} \in (\text{Data} \times \text{Id} \times \text{Atom})\} \\ \{\text{decrypt (t)} \mid \text{t} \in (\text{Data} \times \text{Id} \times \text{Atom})\} \end{pmatrix} \tag{4}$$

4.2 Theoretical system

It is the core of proving the security of cryptographic protocols to define the message types and describe the rules of cryptographic protocols. In this section, we will also give the definition of freshness and propose the persist rule.

(1) Key axiom
Key axiom is as shown in (5), it means that the symmetric key can only match itself, while the private key assigned to the principal only matches the relative public key. In the interaction process of protocols, there are no two principals (in logic of events, this is defined as identifiers) will have the same private key.

$$\left(\begin{array}{c} AxiomK : \forall A, B : Id.\forall k, k : \mathrm{Key}.\forall a : Atom; MatchingKeys(k, k') \Leftrightarrow \\ MatchingKeys(k', k) \wedge MatchingKeys(Symm(a); k) \Leftrightarrow \\ k = Symm(a) \wedge MatchingKeys(PrivKey(A); k) \Leftrightarrow \\ k = A \wedge MatchingKeys(A; k) \Leftrightarrow \\ k = PrivKey(A) \wedge PrivKey(A) = PrivKey(B) \Leftrightarrow \\ A = B \end{array}\right) \tag{5}$$

(2) Causal axioms

The causal axiom includes three axioms: AxiomR, AxiomV and AxiomD, which correspond to event classes receive, verify and decrypt. These three axioms associate the event classes receive, verify and decrypt with three other event classes send, signature and encrypt, respectively (Bickford and Constable 2010), as shown in formula (6).

$$\left(\begin{array}{c} AxiomR : \forall e : E(Rcv).\exists e' : E(Send).(e' < e) \wedge Rcv(e) = Send(e') \\ AxiomV : \forall e : E(Verify).\exists e' : E(Sign).(e' < e) \wedge Verify(e) = Sign(e') \\ AxiomD : \forall e : E(Decrypt).\exists e' : E(Encrypt).e' < e \wedge DEMatch(e, e') \\ DEMatch(e, e') \equiv_{def} plaintext(e) = plaintext(e') \wedge \\ ciphertext(e) = ciphertext(e') \wedge \\ \mathrm{MatchingKeys(key(e); key(e'))} \end{array}\right) \tag{6}$$

AxiomR and AxiomV are almost the same, both of which illustrate the behavior matching in the process of protocol interaction. AxiomD is similar to the previous two axioms, except that key is introduced. Therefore, AxiomD states that the corresponding event encrypt will hold the same information before the event decrypt, and the key can be matched.

(3) Disjointness axioms

The disjointness axioms contain two disjoint relations: one is about the disjoint of seven event classes, and the other is about the disjoint of nonces, private keys, signatures and ciphertext. As shown in formula (7), it is specified that any event in seven special classes is not in other event classes (Bickford and Constable 2010).

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

The second disjoint axiom is shown in formula (8), states that the nonces generated by a principal do not intersect with the private keys, signatures or ciphertext held by the same principal.

$$\left(\begin{array}{c} NonceCiphersAndDisjoint : \forall \mathrm{n} : E(New). \\ \forall s : E(Sign).\forall e : E(Encrypt).\forall A : Id. \\ New(n) \neq signature(e) \wedge New(n) \neq ciphertext(e) \wedge \\ New(n) \neq Private(A) \wedge ciphertext(e) \neq Private(A) \wedge \\ signature(s) \neq Private(A) \wedge signature(s) \neq ciphertext(e) \end{array}\right) \tag{8}$$

(4) Honesty axiom

The private key of the honest principal will not be released, so the signature event, encrypt event and decrypt event all occur on the same honest principal. AxiomS carves the properties of the honest principal as follows (9).

$$\begin{array}{c} AxiomS : \forall A : Id.\forall s : E(Sign).\forall e : E(Encrypt). \\ \forall d : E(Decrypt).Honest(A) \Rightarrow \\ \left\{\begin{array}{c} signer(s) = A \Rightarrow (loc(s) = A) \wedge \\ key(e) = PrivateKey(A) \Rightarrow (loc(e) = A) \wedge \\ key(d) = PrivateKey(A) \Rightarrow (loc(d) = A) \end{array}\right\} \end{array} \tag{9}$$

(5) Flow relation

Flow relation is a complex axiom, which is the relation between causal events of nonces. The *Act* type contains seven event classes, called *actions* (Yanan 2018). (e *has a*) is true if and only if *Atom a* is included in *Action e*, as defined below (10).

$$\left(\begin{array}{c} e has\ a \equiv_{def} \\ (e \in E(New) \wedge New(e)\ has\ a) \vee \\ (e \in E(Send) \wedge Send(e)\ has\ a) \vee \\ (e \in E(\mathrm{Receive}) \wedge \mathrm{Receive}(e)\ has\ a) \vee \\ (e \in E(Encrypt) \wedge Encrypt(e)\ has\ a) \vee \\ (e \in E(Decrypt) \wedge Decrypt(e)\ has\ a) \vee \\ (e \in E(Sign) \wedge Sign(e)\ has\ a) \vee \\ (e \in E(Verify) \wedge Verify(e)\ has\ a) \end{array}\right) \tag{10}$$

The flow direction of *Atom a* from e_1 to e_2 is recorded as $e_1 \xrightarrow{a} e_2$, including the following situations: e_1 and e_2 occur in the same principal; *Atom a* is sent through the plaintext between the event send and receive; *Atom a* in

the plaintext of encrypt event, and the ciphertext flows to a matching decrypt event. The specific flow relation recursion is as shown in (11).

$$\begin{gathered} e_1 \overset{a}{\rightarrow} e_2 =_{rec} (e_1 \, has \, a \wedge e_2 \, has \, a \wedge e_1 \leq_{loc} e_2) \\ \vee \\ \begin{pmatrix} \exists s : E(Send). \exists r : E(Rcv). e_1 \leq s < r \leq e_2 \\ \wedge Send(s) = Rcv(r) \wedge e_1 \overset{a}{\rightarrow} s \wedge r \overset{a}{\rightarrow} e_2 \end{pmatrix} \\ \vee \\ \begin{pmatrix} \exists e : E(Encrypt). \exists d : E(Decrypt). e_1 \leq e < d \leq e_2 \\ \wedge DEMatch(d, e) \wedge key(d) \neq Symma(a) \\ \wedge e_1 \overset{a}{\rightarrow} e \wedge e \overset{ciphertext}{\rightarrow} d \wedge d \overset{a}{\rightarrow} e_2 \end{pmatrix} \end{gathered} \tag{11}$$

(6) Nonce axiom

AxiomF is the nonce axiom, which consists of three parts: AxiomF$_1$, AxiomF$_2$ and AxiomF$_3$. The first part of AxiomF is about the flow properties, as shown in (12).

$$\begin{aligned} &Axiom F_1 : \forall e_1 : E(New). \forall e_2 : E. e_2 has \, New(e_1) \\ &\Rightarrow e_1 \overset{New(e_1)}{\rightarrow} e_2 \end{aligned} \tag{12}$$

AxiomF$_2$, AxiomF$_3$ introduces signature, ciphertext and the relationship between two events. It does not stipulate that signature or ciphertext is related to special events. If an event contains signature or ciphertext, signatures or encrypt events with the same information can be inferred (Bickford and Constable 2010), as shown in (13).

$$\begin{pmatrix} Axiom F_2 : \forall e_1 : E(Sign). \forall e_2 : E. e_2 \, has signature(e_1) \Rightarrow \\ \exists e' : E(Sign). Sign(e') = Sign(e_1) \wedge e' \overset{signature(e_1)}{\rightarrow} e_2 \\ Axiom F_3 : \forall e_1 : E(Encrypt). \forall e_2 : E. e_2 \, has \\ ciphertext(e_1) \Rightarrow \exists e' : E(Encrypt). \\ Encrypt(e') = Encrypt(e_1) \wedge e' \overset{ciphertext(e_1)}{\rightarrow} e_2 \end{pmatrix} \tag{13}$$

(7) Persist Rule

In logic of events, the freshness of nonces has not been defined, so the concept of $Fresh$ needs to be introduced first. Assuming that there is an $Atom\ a$, $Fresh$ means that no principal other than the principal itself can have any message containing $Atom\ a$, and in logic of events, it will not be introduced in the form of logic of events, but the result type is defined as Boolean value. $Fresh$ can be used to refer to any information, but here it is only used to describe the nonces, so as to determine the freshness of the nonces. In addition, when an event does not have $Atom\ a$, it does not have the freshness of $Atom\ a$.

$Fresh$ is introduced to judge whether the nonces have been sent to other principals. In order to ensure the consistency of protocol state and behavior predicate, reduce the complexity and redundancy of protocol analysis process, persist rule is proposed. As shown in formula (14), P is used to refer to persist, which stipulates that the freshness of messages or the Boolean value of events will not change when certain behaviors are executed.

$$\begin{pmatrix} \exists e_1, e_3 : E. \forall e_2 : Send. e_1 < e_2 < e_3 \wedge \\ P(e_1, e_3) \Rightarrow \\ e_1 \notin E(New) \vee Send(e_2) \neq New(e_1) \\ \exists e_1 : E(New). \exists e_2, e_3 : E. e_1 < e_2 < e_3 \wedge \\ P\{Fresh(e_1, New(e_1)), Fresh(e_3, New(e_1))\} \Rightarrow \\ plaintext(e_2) || New(e_1) \wedge \\ ciphertext(e_2) || New(e_1) \end{pmatrix} \tag{14}$$

4.3 Formal description system

The basic sequence is the parameter list of the protocols, and the parameter is the principals' identifiers, which is composed of two or more events. The principal who abides by the protocol participates in multiple threads, and the thread is the basic sequence instance of the protocol and complies with the protocol.

(1) Threads

Thread is an ordered list of events at single location, satisfying formula (15).

$$Thread \equiv_{def} \{thr : Act \, List | \forall i : thr[i] <_{loc} thr[i+1]\} \tag{15}$$

The message in a thread is a collection of all send and receive events in this thread, as shown in (16).

$$\begin{aligned} isMsg(e) &\equiv_{def} e \in E(Send) \vee e \in E(Rcv) \\ messages(thr) &\equiv_{def} filter(isMsg, thr) \end{aligned} \tag{16}$$

For the messages s and r, s is the sent message, r is the received message, if s and r deliver the same message, there is a weak matching relationship between the two messages, represented by $s \sim r$; if there is a direct causal relationship between s and r, and s occurs before r, then there is a strong matching relationship between s and r, represented by $s \mapsto r$, as follows formula (17).

$$\begin{pmatrix} s \sim r \equiv_{def} s \in E(Send) \wedge r \in E(Rcv) \wedge \\ Send(s) = Rcv(r) \\ s \mapsto r \equiv_{def} s \sim r \wedge s < r \end{pmatrix} \tag{17}$$

(2) Matching conversations

Thread thr_1 and thread thr_2 form a matching session with a length of n, containing at least n messages. When the first n messages in a thread are paired, each pair of

$< m_1, m_2 >$ satisfies $m_1 \mapsto m_2 \vee m_2 \mapsto m_1$, and the strong matching session is defined as $thr_1 \overset{n}{\approx} thr_2$. If each pair of $< m_1, m_2 >$ satisfies only $m_1 \sim m_2 \vee m_2 \sim m_1$, a weak matching session will be obtained, which is recorded as $thr_1 \overset{n}{\sim} thr_2$.

The protocol guarantees that the thread matching session satisfies the strong matching property in different locations, and the strong matching property avoids replay attacks, which has more causality proof than the weak matching property.

In the protocol, protocol actions (pas) correspond to threads, which are recorded as $pas(thr)$. If they have the same length, $||pas|| = ||thr||$, then the event matching meets $\forall i < ||\text{thr}||.\text{pas}[i](thr[i])$.

(3) Basic sequences

Basic sequence is a parameter list of basic protocol events, and the parameters of principals are identifiers; principals abide by the protocol and participate in multiple threads of the protocol. Threads are instances of the protocol and interact with another principal in different event positions. The protocol studied by c allows multiple principals to participate (Bickford and Constable 2010). A principal is a member instance of the basic sequence, if an error occurs in the signature or encrypt event, the corresponding verify or decrypt event will not occur (Bickford and Constable 2010). If a principal fails to connect or is not trusted in the protocol, the corresponding receive event of the interaction sequence will not appear. If a principal abides by the protocol, the session will end with receive, verify and decrypt events as a complete basic sequence.

The basic sequence is the relationship between two events' positions and a thread. When the thread is the position parameter given by the basic sequence, the relationship is true. The basic sequence members are as follows (18).

$$Basic \equiv_{def} Id \rightarrow Id \rightarrow Thread \rightarrow P \tag{18}$$

(4) Protocol Definition

Logic of events uses the basic sequence relation table bss to define the protocol, and the protocol is the assertion of the storage location, and the formula is expressed as $Id \rightarrow P$. And the $Protocol\ (bss)$ is defined as follows.

$$\left(\begin{array}{c} \lambda A.\forall e : Act.loc(e) = A \Rightarrow \\ (\exists thr.inOneof(e, thr, bss, A)) \wedge \\ \forall thr_1, thr_2.(inOneof(e, thr_1, bss, A) \wedge \\ inOneof(e, thr_1, bss, A)) \Rightarrow thr_1 \simeq thr_2 \end{array}\right) \tag{19}$$

The event of principals is an instance member of the basic sequence. If the event is one or more instance members, the instance is compatible. The compatibility needs to meet the consistency of parameter selection in the two instances (Bickford and Constable 2010).

5 Proving mutual authentication property of PUF-based protocol

5.1 Proof procedures

In the process of using theorem proving method to prove security properties in cryptographic protocols, the general proof step is to define two situations based on the initiator or responder of the protocol, and then prove the matching session according to the precondition and postcondition of the protocol state.

Finally, two aspects are considered in detail, the proof of weak authentication property and strong authentication property.

However, when using logic of event to prove the security properties of the protocol, besides the proof of mutual weak authentication property and strong authentication property, the definition of the basic sequence of protocol and the analysis of the unilateral sequence of initiator and responder should be taken into account. Only when the bilateral sequences are proved to have strong authentication property can the recognition of protocol be proved. The specific proof process of protocol security based on logic of event is shown in Fig. 2, and detailed information is as follows.

(1) First, it is necessary to formally describe the protocol, standardize the basic sequence of the initiator and responder, and define the strong authentication attributes that the protocol needs to verify;
(2) On the premise of an honest subject, analyze the thread message and assume that the selected thread is a basic sequence instance, define the action on the thread, confirm the matching session that needs to be proved, and then prove the authentication of the initiator or receiver unilaterally;
(3) Confirm whether the matching event matches the current matching session. If it matches, enter the follow-up proof. If not, continue to select the matching session until it is confirmed that the current matching event meets the weak match;
(4) When it has been proved that the weak match is satisfied, analyze the match length in the protocol interaction process, and prove the strong match session according to the relevant axioms and rules in LoET;
(5) If the strong authentication of one party has been successfully proved, it is necessary to prove the strong authentication of the other party. Only one party meets the strong

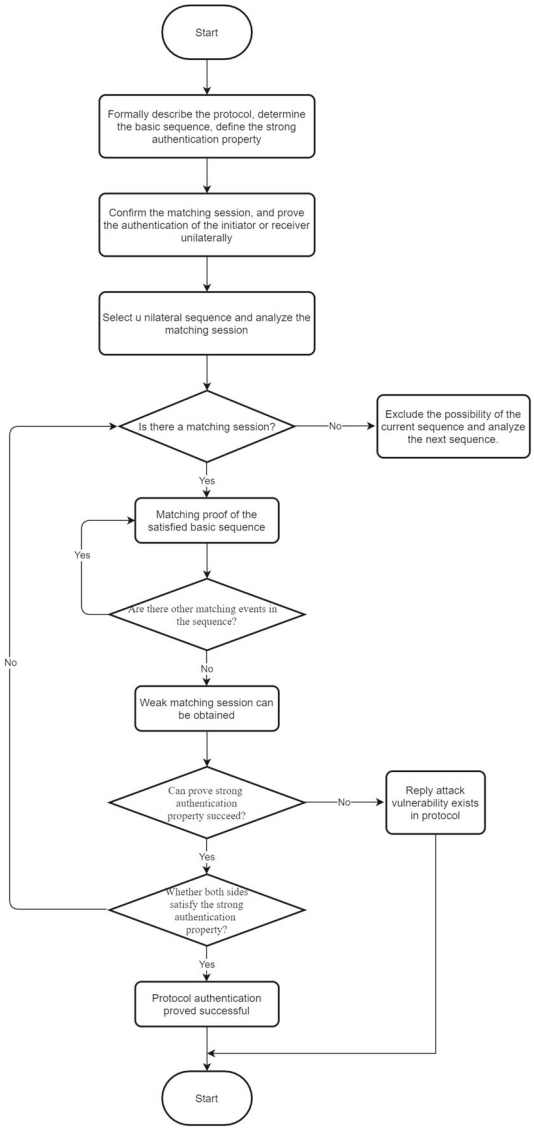

Fig. 2 The process of proving protocol security property based on logic of event

authentication, indicating that the entire agreement still does not meet the strong authentication property.

5.2 Detailed proof

In the research of PUF-based mutual authentication protocol, this paper makes the following assumptions: the response of PUF held by the initiator is credible; in the process of protocol interaction, the nonce is not considered to be leaked due to internal reasons before send event; the interference attack of PUF caused by noise and environmental disturbance is not considered; all principals except honest one can steal the transmission information.

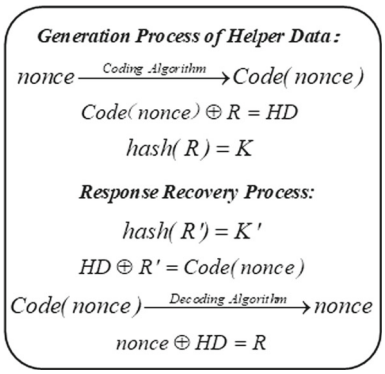

Fig. 3 The principle of PUF implementation

First, we need to abstract the functions implemented by PUF. The main principle of PUF to realize its own security is shown in Fig. 3.

To be specific, the security of PUF is mainly achieved by obtaining HD through XOR of encoded response and nonce of the responder, and then obtaining the encoded nonce through XOR of response R and HD of the initiator, decoding the encoded nonce with XOR and restoring the response R' of the responder, and obtaining the second secret key K through R.

The above functions are abstracted by using logic of events, as shown in the following formula.

$$\begin{pmatrix} New(nonce_1), \\ Encrypt(<< K' >, K >), \\ Send(< B, nonce_1, << K' >, K >>), \\ Rcv(< B, nonce_1, << K' >, K >>), \\ Decrypt(<< K' >, K) \\ New(nonce_2) \\ \text{Encrypt}(<< \text{nonce}_2 || \text{nonce}_1 >, \text{K}' >) \\ \text{Send}(<< \text{nonce}_2 || \text{nonce}_1 >, \text{K}' >) \\ \text{Rcv}(<< \text{nonce}_2 || \text{nonce}_1 >, \text{K}' >) \\ \text{Decrypt}(<< \text{nonce}_2 || \text{nonce}_1 >, \text{K}' >) \end{pmatrix}$$

By abstracting the functions of PUF, we can use logic of events to describe the PUF-based mutual authentication protocol according to the abstracted functions. Firstly, I_1, I_2, I_3, I_4, I_5 are defined as the basic sequences generated by the initiator, and R_1, R_2, R_3, R_4 are the basic sequences generated by the responder. The protocol description based on logic of events is shown in Fig. 4:

To define the identity authentication properties through LoET and verify the authentication of the PUF-based mutual authentication protocol, the PUF-based mutual authentication protocol is sorted by the concept of basic sequence in logic of events, as shown in Fig. 5.

If a protocol can guarantee a strong matching session between two threads in different event locations, the protocol satisfies the strong authentication properties. Therefore,

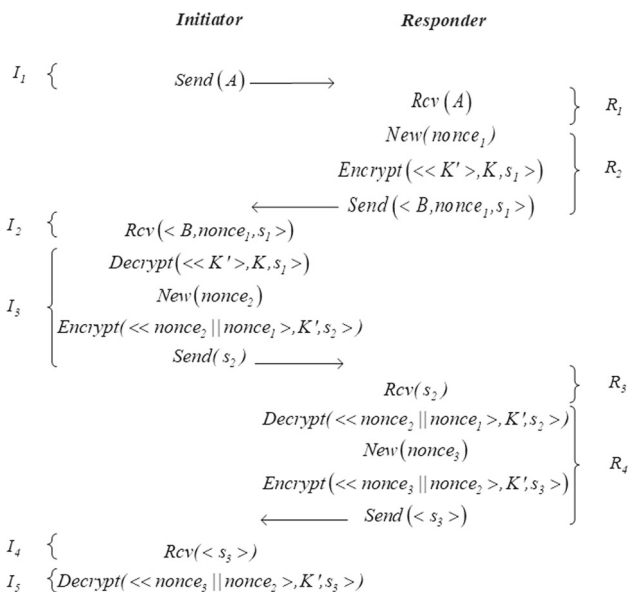

Fig. 4 Description of interactive information in mutual authentication

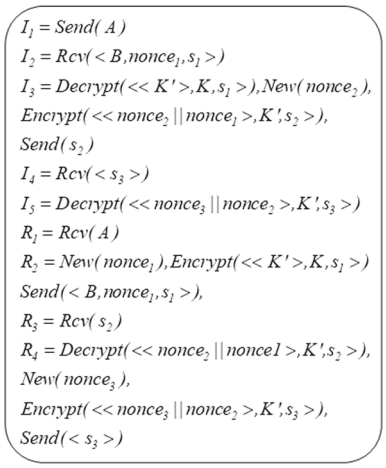

Fig. 5 The basic sequence of PUF-based mutual authentication protocol

by analyzing the basic sequence and defining the PUF-based mutual authentication protocol as $Protocol([I_1, I_2, I_3, I_4, I_5, R_1, R_2, R_3, R_4])$, we can see that the strong authentication properties to be verified in the security certification of the protocol are $Nse| = auth(I_5, 4) \wedge Nse| = auth(R_4, 3)$.

First, we should prove the formula $Nse| = auth(I_5 4)$. Suppose $Initiator \neq Responder$ (A and B, respectively, refer to later), and both principals follow the PUF-based mutual authentication protocol. 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:

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

According to AxiomD and AxiomS, there is an event and this event can make the formula $e' < e_6 \wedge DEMatch(e_1, e') \wedge loc(e') = B \vee loc(e') = A$ hold. Since B follows the PUF-based mutual authentication protocol, event e' must be an instance member in the basic sequence of the PUF-based mutual authentication protocol. In the basic sequence, there are $I_2, I_3, I_4, I_5, R_2, R_3, R_4$ that contain the event $Encrypt()$. In order to ensure the validity of the event, the event classes that the initiator A and the responder B participate in must be two or more sides, as shown in the following formula.

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

Therefore, I_2, I_3, I_4, I_5 can be excluded, which means that there is a basic sequence in R_2, R_3, R_4 that may form a matching session with I_5.

Suppose e' is an example of R_2, then for atoms $nonce_1'$, K_1', K_2, s_1', C (K_1' is the evolution of K, K_2 is the evolution of K_1'), e_0', e_1', e_2', e_3' exist on principal B, which are:

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

From the above formula, it can be found 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 in R_2 can be excluded. Similarly, R_3 can also 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 principal D, there

are events e_0', e_1', e_2', e_3', e_4', e_5', e_6', e_7', e_8' at principal B such that:

$$\begin{pmatrix} \mathrm{e}_0' <_{loc} e_1' <_{loc} e_2' <_{loc} e_3' <_{loc} e_4' <_{loc} e_5' <_{loc} e_6' <_{loc} e_7' <_{loc} e_8' \wedge \\ Rcv(\mathrm{e}_0') =< D > \wedge New(e_1') =< nonce_1' > \wedge \\ Encrypt(e_2') =<< K_2' >, K_1', s_1' > \wedge \\ Send(e_3') =< B, nonce_1', s_1' > \wedge \\ Rcv(e_4') =< s_2' > \wedge \\ Decrypt(e_5') =<< nonce_2' || nonce_1' >, K_2', s_2' > \wedge \\ New(e_6') =< nonce_3' > \wedge \\ Encrypt(e_7') =<< nonce_3' || nonce_2' >, K_2', s_3' > \wedge \\ Send(e_8') =< s_3' > \end{pmatrix}$$

Through the above formula, the following can be found:

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

It can be seen that $e_7' = e', D = A, nonce_3' = nonce_3, nonce_2' = nonce_2, K_2' = K', s_3' = s_3$, and it can be concluded that:

$$\begin{pmatrix} \mathrm{e}_0' <_{loc} e_1' <_{loc} e_2' <_{loc} e_3' <_{loc} e_4' <_{loc} e_5' <_{loc} e_6' <_{loc} e_7' <_{loc} e_8' \wedge \\ Rcv(\mathrm{e}_0') =< A > \wedge New(e_1') =< nonce_1 > \wedge \\ Encrypt(e_2') =<< K' >, K, s_1 > \wedge \\ Send(e_3') =< B, nonce_1, s_1 > \wedge \\ Rcv(e_4') =< s_2 > \wedge \\ Decrypt(e_5') =<< nonce_2 || nonce_1 >, K', s_2 > \wedge \\ New(e_6') =< nonce_3 > \wedge \\ Encrypt(e_7') =<< nonce_3 || nonce_2 >, K', s_3 > \wedge \\ Send(e_8') =< s_3 > \end{pmatrix}$$

After it is known that, e' must be an instance in R_4, there must be an event e'', and this event can make $\mathrm{e}'' < \mathrm{e}_5' \wedge \mathrm{DEMatch}(\mathrm{e}_5', \mathrm{e}'') \wedge \mathrm{loc}\left(\mathrm{e}''\right) = \mathrm{A} \vee \mathrm{loc}\left(\mathrm{e}''\right) = \mathrm{B}$ hold, which is equivalent to $Encrypt\left(e''\right) =<< nonce_2 || nonce_1 >, K', s_2 >$.

Therefore, e'' must be an instance member in the basic sequence of the PUF-based mutual authentication protocol. In the basic sequence, I_2, I_3, I_4, I_5, R_2, R_3, R_4 contain the event $Encrypt()$. Since it is based on the current events that have occurred, if all events that have not occurred are not taken into account, I_4, I_5 can be excluded. Then, according to the above proof formula, R_2, R_3, R_4 can be excluded. It can be concluded that there is a basic sequence in I_2, I_3 that contains sequence e''.

Suppose e'' is an instance on I_2, for atoms K_1', K_2', s_1', $nonce_1'$, E, event e_0'', e_1'', e_2'' exist in principal A, then there are:

$$\begin{pmatrix} \mathrm{e}_0'' <_{loc} e_1'' <_{loc} e_2'' \wedge \\ Encrypt(\mathrm{e}_0'') =<< K_2' >, K_1', s_1' > \wedge \\ Send(e_1'') =< E, nonce_1', s_1' > \wedge \\ Rcv(e_2'') =< E, nonce_1', s_1' > \end{pmatrix}$$

From the above formula, it can be concluded that the encrypted event e_0' in I_2 does not match the decrypt event e_5', so the possibility that e'' is an instance of I_2 is excluded.

Suppose e'' is an instance of I_3, for atoms K_1', K_2', s_1', s_2', $nonce_1'$, $nonce_2'$, F, e_3'', e_4'', e_5'', e_6'' exist in principal A, which are:

$$\begin{pmatrix} \mathrm{e}_0'' <_{loc} e_1'' <_{loc} e_2'' <_{loc} e_3'' <_{loc} e_4'' <_{loc} e_5'' <_{loc} e_6'' \wedge \\ Encrypt(\mathrm{e}_0'') =<< K_2' >, K_1', s_1' > \wedge \\ Send(e_1'') =< E, nonce_1', s_1' > \wedge \\ Rcv(e_2'') =< E, nonce_1', s_1' > \wedge \\ Decrypt(\mathrm{e}_3'') =<< K_2' >, K_1', s_1' > \wedge \\ New(e_4'') =< nonce_2' > \wedge \\ Encrypt(e_5'') =<< nonce_2' || nonce_1' >, K_2', s_2' > \wedge \\ Send(e_6'') =< s_2' > \end{pmatrix}$$

According to the above results, we can get:

$$\begin{pmatrix} Encrypt(e'') =<< nonce_2 || nonce_1 >, K', s_2 > \\ =<< nonce_2' || nonce_1' >, K_2', s_2' >= Encrypt(e_5'') \end{pmatrix}$$

Then, we can see that there are $e_5'' = e'', F = B, nonce_2' = nonce_2, nonce_1' = nonce_1, K_2' = K', s_2' = s_2$, and we can get:

$$\begin{pmatrix} \mathrm{e}_0'' <_{loc} e_1'' <_{loc} e_2'' <_{loc} e_3'' <_{loc} e_4'' <_{loc} e_5'' <_{loc} e_6'' \wedge \\ Encrypt(\mathrm{e}_0'') =<< K' >, K, s_1 > \wedge \\ Send(e_1'') =< B, nonce_1, s_1 > \wedge \\ Rcv(e_2'') =< B, nonce_1, s_1 > \wedge \\ Decrypt(\mathrm{e}_3'') =<< K' >, K, s_1 > \wedge \\ New(e_4'') =< nonce_2 > \wedge \\ Encrypt(e_5'') =<< nonce_2 || nonce_1 >, K', s_2 > \wedge \\ Send(e_6'') =< s_2 > \end{pmatrix}$$

When e'' is an instance in I_3, there must be another event e''' and this event can make the following formula $e''' < e_3'' \wedge DEMatch(e_3'', e''') \wedge loc\left(e'''\right) = A \vee loc\left(e'''\right) = B$ true, that is $Encrypt\left(e'''\right) =<< K' >, K, s_1 >$.

If e''' exists, then event e''' must be an instance member in the basic sequence of the PUF-based mutual authentication protocol, among which I_2, I_3, I_4, I_5, R_2, R_3, R_4 contain the event $Encrypt()$. Since the event status point is based on the event that has occurred, the event that has not occurred will not be taken into account, R_3, R_4, I_4, I_5 can be excluded. Then, according to the above proof formula, I_1, I_2 can be excluded. It can be concluded that there is a basic sequence in R_1, R_2 that contains sequence e'''.

Suppose e''' is an instance on R_1, there is no atom on R_1, and e''' needs to match e_5'', so the possibility that e''' is an instance on R_1 is excluded. Suppose e''' is an instance on R_2, for atoms $nonce_1'$, K_1', K_2', s_1', G, event e_0''', e_1''', e_2'''

exist in principal B, then there are:

$$\begin{pmatrix} e_0''' <_{loc} e_1''' <_{loc} e_2''' \wedge \\ New(e_0''') =< nonce_1' > \wedge \\ Encrypt(e_1''') =<< K_2' >, K_1', s_1' > \vee \\ Send(e_2''') =< G, nonce_1', s_1' > \end{pmatrix}$$

From the above description of events, it can be found that:

$$\begin{pmatrix} Encrypt(e''') =<< K' >, K, s_1 >= \\ << K_2' >, K_1', s_1' >= Encrypt(e_1''') \end{pmatrix}$$

Then, we can see that there are $e_1''' = e'''$, $F = B$, $K = K_1'$, $K_2' = K'$, $s_1' = s_1$, and we can get:

$$\begin{pmatrix} e_0''' <_{loc} e_1''' <_{loc} e_2''' \wedge \\ New(e_0''') =< nonce_1 > \wedge \\ Encrypt(e_1''') =<< K' >, K, s_1 > \wedge \\ Send(e_2''') =< B, nonce_1, s_1 > \end{pmatrix}$$

According to the above results, we can get:

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

There is already a weak match session of length 3.

The next, to prove the strong match session, we must first prove that $e_3' < e_0, e_4 < e_4', e_8' < e_5$. In the above-mentioned proof, according to AxiomF and the Flow relation, we can see that there is a sending event between event e_3', e_0, combining the persist rule, and we can know that the freshness of $nonce_1$ changes after the send event, and in e_0, the fresh state of nonce no longer exists.

Suppose $n = thr[j], n \in E(New), e = thr[i], j < i$, then there is no k between j and i, so that $thr[k] \in E(Send)$, and $New(n)$ is not released before e.

For events e_3', e_0, if $e_3' \leq j$, the resulting sequence is $e_3' < e_0$. Conversely, if $e_0 <_{loc} j <_{loc} e_3'$, then j must be a member of some other threads of B, but according to the above inference, there is no sending event between e_3', e_0, that is, the freshness of the nonce has been kept before e_0, and $e_3' < e_0$ can be proved by AxiomF. In addition, we can use the same logic to combine persist rule, AxiomF and flow relation to prove $e_4 < e_4'$ and $e_8' < e_5$, then $Nse| = auth(I_5, 4)$ is proved.

The same reason can prove that $Nse| = auth(R_4, 3)$, and we can get the final formula that needs to be proved, that is, $Nse| = auth(I_5, 4) \wedge Nse| = auth(R_4, 3)$, the PUF-based mutual authentication protocol meets two strong authentication properties at the same time. We can judge that in the authentication process of the protocol, there is no possibility that the attacker disguises the initiator and responder and attacks by replay attacks. The security of the PUF-based mutual authentication protocol is proved.

6 Conclusions and future work

In this paper, we use the method of theorem proving which based on logic of event to analyze the security property of PUF-based mutual authentication protocol. The main work is as follows.

(1) On the basis of logic of event, the freshness of nonce is defined for the first time, and persist rule is proposed according to the freshness concept to ensure the consistency of protocol states and behavior predicates in the proof process, and reduce the complexity and redundancy in protocol analysis process.
(2) The interaction of PUF-based mutual authentication protocol is formally described by using logic of event, and the security property of the protocol is formally analyzed to verify its strong authentication property.
(3) Logic of event is used to abstract hardware security functions for the first time, verify protocols based on hardware security and expand the application scope of logic of event. In addition, although this paper has successfully verified the security of PUF-based mutual authentication protocol, there are still some problems.
(4) This paper does not consider the PUF interference attack caused by noise and environmental disturbance in the real environment. This is also a defect that theorem proving cannot consider in some aspects. Next, we can consider using other formal methods to analyze the security of the protocol.
(5) With the rapid development of various types of cryptographic protocols, logic of event lacks corresponding axioms and rules in the process of dealing with some protocols. Therefore, in order to verify the new protocols and the new security properties other than authentication, logic of event needs to be further expanded.

References

Barbareschi M, De Benedictis A, Mazzocca Nicola (2018) A PUF-based hardware mutual authentication protocol. J Parall Distrib Comput 119:107–120

Bickford M (2008) Unguessable atoms: a logical foundation for security. In: Working conference on verified software: theories, tools, and experiments. Springer, Berlin, Heidelberg, pp 30–53

Bickford M, Constable RL (2003) A logic of events. Cornell University

Bickford M, Constable RL (2010) Automated proof of authentication protocols in a logic of events. In: VERIFY@ IJCAR, pp. 13-30

Brzuska C et al. (2011) Physically uncloneable functions in the universal composition framework. Annual cryptology conference. Springer, Berlin, Heidelberg

Chatterjee D, Hazra A, Mukhopadhyay D (2019) Formal analysis of PUF instances leveraging correlation-spectra in Boolean functions. In: Security, privacy, and applied cryptography engineering. SPACE 2019. Lecture Notes in Computer Science, vol 11947

Delvaux J et al (2015) A survey on lightweight entity authentication with strong PUFs. ACM Comput Surv (CSUR) 48(2):26

Focardi R, Luccio FL (2020) Automated analysis of PUF-based protocols. In: 2020 IEEE 33rd Computer security foundations symposium (CSF), pp. 304-317

Meguerdichian S, Potkonjak M (2011) Security primitives and protocols for ultra low power sensor systems. SENSORS, 2011 IEEE. IEEE

Lee YS, Lee HJ, Alasaarela E (2013) Mutual authentication in wireless body sensor networks (WBSN) based on physical unclonable function (PUF). In: 2013 9th International wireless communications and mobile computing conference (IWCMC). IEEE

Li Y (2018) Formal analysis of wireless mesh network client authencation protocols based on event logic. East China Jiaotong University

Meadows C (2003) Formal methods for cryptographic protocol analysis: emerging issues and trends. IEEE J Select Areas Commun 21(1):44–54

Meguerdichian S, Potkonjak M (2011) Security primitives and protocols for ultra low power sensor systems. SENSORS, 2011 IEEE. IEEE

Meihua X, Ma C, Deng C, Zhu K (2015) A novel approach to automatic security protocol analysis based on authentication event logic. Chinese J Electron 24(1):187–192

O'Neill M (2016) Insecurity by design: todays IoT device security problem. Engineering 2(1):48–49

Pappu R et al (2002) Physical one-way functions. Science 297(5589):2026–2030

Ruhrmair Ulrich, Jan S, Frank S (2009) On the foundations of physical unclonable functions. IACR Cryptology ePrint Archive 2009:277

Suh GE, Devadas S (2007) Physical unclonable functions for device authentication and secret key generation. In: 2007 44th ACM/IEEE Design Automation Conference. IEEE

van Dijk Marten, Ruhrmair U (2012) Physical unclonable functions in cryptographic protocols: security proofs and impossibility results. IACR Cryptology ePrint Archive 2012:228

Wachsmann Christian, Sadeghi A-R (2014) Physically unclonable functions (PUFs): Applications, models, and future directions. Synth Lect Inf Secur, Privacy, Trust 5(3):1–91

Xiao M, Bickford M (2009) Logic of events for proving security properties of protocols. In: 2009 International conference on web information systems and mining (WISM 2009). IEEE

Yang K et al (2018) Proving mutual authentication property of KerNeeS protocol based on logic of events. IEEE Access 6:51853–51863