Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 2420590, 11 pages
https://doi.org/10.1155/2022/2420590

Research Article

Formal Analysis of the Security Protocol with Timestamp Using SPIN

Meihua Xiao, Weiwei Song, Ke Yang, Ri OuYang, and Hanyu Zhao

School of Software, East China Jiaotong University, Nanchang 330013, China

Correspondence should be addressed to Weiwei Song; songww@ecjtu.edu.cn

Received 19 May 2022; Accepted 28 June 2022; Published 23 August 2022

Academic Editor: Dalin Zhang

The verification of security protocols is an important basis for network security. Now, some security protocols add timestamps to messages to defend against replay attacks by network intruders. Therefore, verifying the security properties of protocols with timestamps is of great significance to ensure network security. However, previous formal analysis method of such protocols often extracted timestamps into random numbers in order to simplify the model before modeling and verification, which probably cause time-dependent security properties that are ignored. To solve this problem, a method for verifying security protocols with timestamps using model checking technique is proposed in this paper. To preserve the time-dependent properties of the protocol, Promela (process meta language) is utilized to define global clock representing the protocol system time, timer representing message transmission time, and the clock function representing the passage of time; in addition, a mechanism for checking timestamps in messages is built using Promela. To mitigate state space explosion in model checking, we propose a vulnerable channel priority method of using Promela to build intruder model. We take the famous WMF protocol as an example by modeling it with Promela and verifying it with model checker SPIN (Simple Promela Interpreter), and we have successfully found two attacks in the protocol. The results of our work can make some security schemes based on WMF protocol used in the Internet of things or other fields get security alerts. The results also show that our method is effective, and it can provide a direction for the analysis of other security protocols with timestamp in many fields.

1. Introduction

With the development of the industrial Internet of things, 5G, blockchain, and cloud computing, many related security issues [1, 2] have arisen at the same time. As an important basis for security in these fields, security protocols are now attracting more and more researchers' interest. The main purpose of using a security protocol is to ensure the security of network communication, but the protocol itself is vulnerable to attacks by network intruders. A seemingly simple and correct security protocol may have vulnerabilities; a typical example is the Needham-Schroeder public key protocol, whose simplified version has only three exchanged messages, but it was discovered by Lowe [3] 17 years after its publication that there is a security vulnerability, which allows intruders to destroy the authentication of the protocol. Because security protocols are difficult to check intuitively and analyzing the security of protocols by relying on empirical principles is inefficient, currently the most effective way to analyze and verify security protocols is formal methods [4]. Model checking [5] is one of the formal verification methods, and SPIN [6], as a simple and efficient model checker, has been widely used in the field of protocol verification. Reference [7] proposed a method of statically analyzing the knowledge of the intruder and then using SPIN to formally verify the classical Needham-Schroeder public key authentication protocol and successfully find the vulnerability. After that, some scholars made further research on the basis of this method and used SPIN to analyze and verify the MANET protocol [8], the RFID three-party authentication protocol [9], and the OAuth 2.0 protocol [10], respectively. Reference [11] proposed a method for formal analysis and verification of the authentication and secrecy of a class of security protocols using SPIN/Promela. In that paper, the intruder

can dynamically intercept message in the network, which help the intruder perform replay attacks or forgery attacks on security protocols. The intruder model of this method is efficient but the number of state transitions is large; if this method is used to analyze security protocols with timestamps, the problem of state space explosion is prone to occur. The methods in [12, 13] alleviated the problem of state spaces explosion, but they aimed at security protocols without timestamps. The above methods can use SPIN to analyze and verify the authentication or secrecy of security protocols with random numbers, but the properties of security protocols with timestamps [14–18] are closely related to specific time factors, so the above methods are difficult to work for them. In fact, the formal analysis of security protocols with timestamps is difficult, and the difficulties are mainly reflected in two aspects: First, the security protocols with timestamps are applied in real-time systems, and not only the protocol agents but also a time model to describe various time factors in the protocol needs to be modeled during analysis, so the overall model is more complex and prone to state explosion problems; second, it is difficult to convert time-dependent properties into formulas or languages that can be recognized by the verification model. Therefore, timestamps were often extracted into random numbers which probably caused time-dependent security properties that are ignored in the previous work of formal analysis of security protocols with timestamps. Aiming at the above problems, this paper proposes a modeling method that preserves the time factors of the protocol by using the Promela assertion that SPIN can recognize to express the time-dependent property, so that the freshness of protocol message can be verified. We take the well-known WMF (wide-mouth frog) protocol [19] that plays an important role in some IoT security scenarios [20, 21], which contains timestamps as an example to illustrate our method. By applying our method, one attack path [22, 23] that violates key freshness and another that violates the authentication in the WMF protocol are successfully found.

The main contributions of this paper are as follows:

(1) A modeling method that preserves the time factors of the protocol with timestamp is proposed

(2) A way of expressing key freshness property that can be converted into Promela assertions is proposed

(3) We propose a method of using Promela to build intruder model based on vulnerable channel priority to mitigate state space explosion and successfully find the attacks of WMF protocol

The structure of this paper is arranged as follows. In the next section, we give the overall scheme for analyzing security protocol with timestamp using SPIN. In Section 3, preliminary knowledge about WMF protocol is given. We dedicate Section 4 to use Promela to model the WMF protocol. In Section 5, we use assertion and linear temporal logic (LTL) to indicate key freshness and authentication property of WMF protocol, respectively. In Section 6, we present the experiment result with SPIN. We conclude in Section 7 by a summary and outlook.

2. The Overall Scheme

In this section, we give the overall scheme which shows all the modeling and verification work of this paper and the overall analysis diagram is shown in Figure 1. Our scheme is divided into three steps as follows:

Step 1. Use Promela to build a complete model M, which includes four model parts: discrete-time model part, protocol agent model part, timestamp checking model part, and intruder model part.

Step 2. Use assertion to represent time-dependent properties, such as key freshness, and use LTL represent other time-independent properties, such as authentication.

Step 3. Input the above model and security properties into model checker SPIN to automatically verify whether the protocol model satisfies the properties.

Note that the verification process is automated because of the excellent algorithm design within SPIN and if the protocol does not satisfy the security property, SPIN will generate a counterexample and give the specific attack path of the counterexample.

In fact, each step above corresponds to one of the third to fifth sections where you can see the details of the analysis process of the WMF protocol. However, we must introduce some preliminary knowledge about the WMF protocol in next section before we analyze the protocol.

3. WMF Protocol

3.1. The Description of WMF Protocol. The WMF protocol is an authentication and key agreement protocol which was specially designed to provide secure data transmission and authentication services to insecure networks. The protocol can be described as follows:

$$\begin{aligned} A -> S:\ & A, \{T_A, B, K\_AB\}K_{AS}, \\ S -> B:\ & \{T_S, A, K\_AB\}K_{BS}. \end{aligned} \tag{1}$$

As you see above, the protocol is divided into two steps.

Step 1. Agent A as an initiator is responsible for generating a temporary session key K_AB shared with B, encrypting it together with B's agent identifier and A's current time T_A to form a ciphertext block $\{T_A, B, K\_AB\}\ K_{AS}$. Then A attaches its own agent identifier A to form Message1: A, $\{T_A, B, K\_AB\}\ KAS$ and send it to the trusted server S.

Step 2.S is responsible for encrypting the key K_AB and the agent identifier of A in the received message with the server's current time to form *Message2*: $\{Ts, A, K\_AB\}\ K_{BS}$, and send it to B.

It should be noted that when the server S receives the message sent by agent A, it will check the timestamp T_A in the message to determine whether the message is fresh. When agent B receives a message from server S, it will check whether the timestamp Ts in the message is newer than the timestamp sent by S before (to resist replay attacks). If Ts is

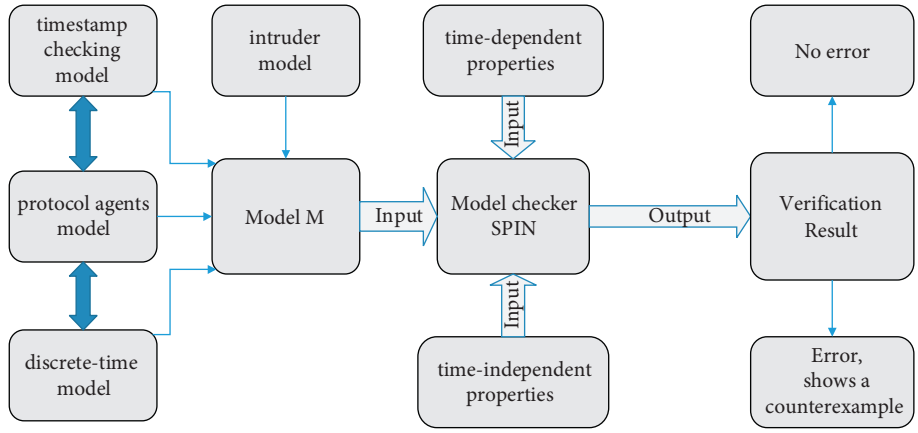

Figure 1: Security protocol with timestamp verification scheme.

the latest, the message will be determined by B that is fresh. And B determines the initiator of the protocol based on the agent identifier in the message to complete one-way authentication.

3.2. Security Properties of WMF Protocol. Two fundamental security properties of the WMF protocol are key freshness and authentication. The key freshness requires that the session key received by the receiver must be generated by the current round of protocol sessions. The authentication, which includes one-way authentication and two-way authentication, is used to determine whether the identity of the communicating party is consistent with the identity claimed in the message. What the WMF protocol needs to satisfy is the one-way authentication. In order to strictly describe the key freshness and one-way authentication, some basic symbols are first defined. The specific symbols and their meanings are shown in Table 1. Based on the basic notation, the definitions of key freshness and one-way authentication are given below.

Definition 1. (*key freshness*). Assuming that the agent's processing of messages is completed immediately, protocol P satisfies *key freshness* if and only if

$$N * D\max \geq Tn - Ts \&\& N * D\min \leq Tn - Ts. \quad (2)$$

Definition 2. (*one-way authentication*). The protocol P satisfies the *one-way authentication,* if and only if *Rec* confirms its receipt or indirect receipt via *Ser* message from *Init* after *Init* initiates or indirectly initiates via *Ser* a session with *Rec*.

3.3. Intruder Rule Description. For the intruder, this paper adopts the Dolev-Yao [24] intruder model specification. In this model specification, an intruder has complete control over the network, and the intruder has the following capabilities:

(1) Ability to eavesdrop, block, and intercept all messages on the network

Table 1: Basic symbols and meanings.

Symbol	Meaning
P	Authentication and key agreement protocol
Init	Protocol initiator
Rec	Protocol responder
Ser	Trusted server
sk	Session key generated by the protocol initiator
Ts	The generation time of *sk*
Tn	The reception time of *sk*
Dmax	Maximum time spent in message transmission
Dmin	Minimum time spent in message transmission
N	The number of messages in a protocol

(2) Ability to send and resend messages

(3) Decompose and combine messages

(4) Ability to impersonate any protocol participant

(5) After knowing the decryption key, the intruder can decrypt the encrypted message

(6) Familiarize all the agent identifiers participating in the protocol

Although the intruders are powerful, they are not omnipotent. For example, the intruder cannot decrypt the ciphertext without the corresponding key, and the intruder cannot infer the key according to the ciphertext. The model specification assumes that the cryptographic algorithm used by the protocol is perfect, which is beneficial for us to focus on the protocol design level without caring about the cryptographic system.

Although intruders can intercept messages from the network, they need to follow certain rules, which can be used to constrain the Promela model of the intruder in Section 4.4, to decompose and combine messages [11]. After an intruder gets a message and wants to learn knowledge from it, he needs to follow the rules in Figure 2, where $I \sim x$ indicates that the intruder can deduce and get x. Rule 1 means that if the intruder possesses some knowledge, he can deduce the same knowledge in the message; Rule 2 and Rule 3 state that if the intruder can get some knowledge, he can deduce the part of knowledge in the message; Rule 4 means that if the intruder can get both the encrypted message component

$\{x\}K$ and the key K, then he can decrypt the message component and gain knowledge.

Meanwhile, the intruder can create new messages based on their knowledge. The creation of new messages needs to follow the construction rules of Figure 3.

Rule 1 means that if the intruder has some knowledge, he can create an integral message based on this knowledge; Rule 2 indicates that if the intruder can get some integral knowledge, he can create message components which is part of integral knowledge; Rule 3 means that if the intruder can get the knowledge x and the corresponding key K, it can create encrypted message components $\{x\}K$.

(1) $\dfrac{x \in \mathrm{K}}{\mathrm{I} \sim x}$ possess (2) $\dfrac{x \sim (x_1, x_2)}{\mathrm{I} \sim x_1}$ proj 1

(3) $\dfrac{x \sim (x_1, x_2)}{\mathrm{I} \sim x_2}$ proj 2 (4) $\dfrac{\mathrm{I} \sim \mathrm{x}\{x\}\mathrm{K} \quad \mathrm{I} \sim \mathrm{K}}{\mathrm{I} \sim x}$ decrypt

Figure 2: Knowledge deconstruction rules.

(1) $\dfrac{x \in \mathrm{K}}{\mathrm{I} \sim x}$ possess (2) $\dfrac{\mathrm{I} \sim x_1 \quad \mathrm{I} \sim x_2}{\mathrm{I} \sim (x_1, x_2)}$ pair

(3) $\dfrac{\mathrm{I} \sim x \quad \mathrm{I} \sim \mathrm{K}}{\mathrm{I} \sim \{x\}\mathrm{K}}$ encrypt

Figure 3: Knowledge construction rules.

4. WMF Protocol Model

Promela (process meta language) [5] is a modeling language for describing concurrent systems, and it is also the input language for the model checker SPIN. Promela uses processes to represent system behaviors and achieves information exchange between different processes through message channels. Promela can simulate the behavior of protocol agents and the communication between protocol agents, so it is very suitable for modeling security protocols.

In this section we take the WMF protocol as an example to illustrate the entire process of building the security protocol model with timestamps. The WMF protocol is modeled using Promela language, including four parts: discrete-time model, timestamp checking model, protocol agent model, and intruder model. It should be noted that the model is based on the following assumptions, which prevent other factors from affecting the protocol model.

Assumption 1. The receiver trusts the initiator to be capable of generating a perfect key.

Assumption 2. The clocks between different agent and servers in the protocol runtime environment are synchronized.

Assumption 3. Intruders cannot tamper with the clock, and all timestamps in the protocol indicate the real current time.

```
#define timer int
timer T_global;
timer SerT;
timer IniT;
timer IntT;
#define tick (x) \
if
:: x>=1->atomic {x=x-1;T_global++}
:: else;
fi
#define set (x, y) x=y
#define expire (x) (x==0)
#define delay (x, y) set (x, y); expire (x)
proctype Timers ()
{
do
:: timeout ->atomic { tick (x);}
od;
}
```

Figure 4: Discrete-time model.

4.1. Discrete-Time Model. Security protocols with timestamps not only need to consider the time order of process communication, but also need to consider the specific time spent in communication, so we build a discrete-time model as shown in Figure 4. The first macro *timer* is used as an alias for *int*, which is used to represent discrete-time type. The second macro *tick* (*x*) represents the passage of time, and the parameter *x* is a timer variable. Each time the tick statement is executed, the value of *x* is reduced by one, which means that a unit of time has passed. At the same time, the value of the global clock variable *T_global* increases by one, which means one unit of time of the system elapses. *T_global* also plays the role of clock synchronization in the system because the current time of all agents in the model is represented by it. The value of the *T_global* will change with the elapse of time caused by any agent transmitting a message. The third macro *set (x, y)* is used to assign a value represented by *y* which indicates message transmission time to the timer variable *x*. The fourth macro *expire(x)* is a blocking statement stopping the program, and only if *x* is equal to 0, which indicates that the message has been transmitted in the channel, the program will continue to execute at this time; instead, the process in which the statement is located will be blocked if *x* is not equal to 0, which means there are still messages in transit. The *timeout* is a predefined Boolean variable in Promela. The value of *timeout* is *true* when all other processes in the system enter the blocking state, and in this situation the *tick* statement is executed to simulate the passage of time. The *Timer* process works with *set (x, y)* and *expire(x)* to model the protocol transmission time. For example, if *statement_sent* and *statement_rec* represent message sending and message

receiving in the protocol model, respectively, then *{statement_sent; set(x,5); expire(x); statement_rec}* statement indicates that the message receiving statement will be executed after 5 units of time after the message sending statement is executed. Therefore, *set (x, y)* and *expire(x)* are combined into the fifth macro *delay (x, y)* to directly represent the transmission time of the message.

4.2. Timestamp Checking Model. We define two functions *TsTimelyCheck(x)* and *TsNewstCheck(x)* to achieve the checking of timestamps by the server and the receiver, respectively. In the WMF protocol, to ensure that the message is fresh, the server not only determines whether the timestamp in the received message from an agent is newer than the timestamp sent by the agent before, but also determines whether the timestamp in the received message is within the allowed time window. The receiver will decide whether the timestamp in the received message is newer than the timestamp sent by the server before, and the receiver will accept the message if it receives the latest timestamp. Therefore, the timestamp checking model of the server and receiver is shown in Figure 5. Of course, the model will be embedded in the server process and the receiver process in the following section.

4.3. Protocol Model. The first step in modeling the agent of the protocol is to construct a set of variable names. We define a set of names representing all possible variables (except timestamps) in WMF protocol, including honest agent A, agent B, server S and intruder I, and all possible keys *Kab, Kas, Kbs, Kis, Kbi, Kai*. But the timestamp is not placed in the variable names set, because the timestamp variable is part of the time model. Set of variable names for the WMF protocol is as follows:

$$m\text{type} = \{A, B, I, S, Kab, Kas, Kbs, Kis, Kbi, Kai\}. \quad (3)$$

The second step is to model the message channel through which the communication takes place. There are two messages in the WMF protocol, and different messages correspond to different message structures, so we define two synchronization channels, each of which is responsible for transmitting messages of one structure. Promela model for two synchronization channels is as follows:

$$\begin{aligned} \text{chan}c1 &= [0]\text{of}\{m\text{type}, \text{timer}, m\text{type}, m\text{type}, m\text{type}\}; \\ \text{chan}c2 &= [0]\text{of}\{\text{timer}, m\text{type}, m\text{type}, m\text{type}\}. \end{aligned} \quad (4)$$

In there, *mtype* represents the enumeration type, and its value range is all the variables in the variable name set constructed in the first step, which are used to represent all other types of variables in the protocol message except the timestamp. In messages containing timestamps, the timer type variable is used to represent the timestamp. For example, timer $T_A = 2$, indicating that the value 2 of the timestamp T_A in the message is the time when the message was sent. The third step is the realization of the protocol role. An honest agent in the WMF protocol can play roles as initiator, server, and receiver, and each role corresponds to a process in our model.

```
#define delay_1   1
#define delay_2   2
timer SerTs_latest;
timer T_Rec_latest;
#define TsTimelyCheck (x)
((x>=T_global-delay_2)&& (x<=T_global) &&  ((x>=T_Rec_latest))
#define TsNewstCheck (x)  (x = = SerTs_latest)
```

Figure 5: Timestamp checking model.

```
proctype Initiator (mtype a ; mtype b)
{
  mtype s, kas, kab, t;
  s=S;
  atomic {
  SessionKey (a, b, kab);
  kas=ShareSKey (a);
  if
  ::delay (IniT, delay_2) ;
  ::delay (IniT, delay_1) ;
  fi;
  Init (a,b);
  c1 ! a, Tstart, b, kab, kas;
  }
}
```

Figure 6: Process of initiator.

(1) The realization of initiator process is as shown in Figure 6.

The initiator role process has two parameters representing the initiator itself and the agent for which it wants to establish a temporary session key. In the macro *Sessionkey (a, b, kab)*, the parameters a and b represent two agents that want to communicate with each other, and the parameter *kab* represents the session key for secret communication between agent a and agent b. According to the pairwise combination of agents A, B, and I, there are three possible session keys in total as follows:

$$\begin{aligned} &\#\text{defineSessionKey}(a,b,k)\text{if} \\ &:: ((a == A)\&\&(b == B))\|((a == B)\&\&(b == A)) - > k = Kab \\ &:: ((a == I)\&\&(b == B))\|((a == B)\&\&(b == I)) - > k = Kbi \\ &:: ((a == A)\&\&(b == I))\|((a == I)\&\&(b == A)) - > k = Kai \\ &\text{fi} \end{aligned} \tag{5}$$

The full definition of the macro *Init (a, b)* is

$$\text{define Init}(a,b)\ \text{if} :: (a == A\&\&b == B) - > InitAB = 1; :: \text{else; fi.} \tag{6}$$

The specific meaning of it is that the initiator A initiates a protocol session for key establishment with the receiver *B*. *ShareSKey(a)* represents the shared key between agent *A* and server *S*. The *delay* statement is used to indicate the transmission time from the initiator sending the message to the server receiving the message. In order to be able to validate models using both asynchronous and synchronous channels without changing the order of statements, we place the *delay* statement before the sending statement, but this does not cause an error in the time model in the protocol, because in a synchronous channel in Promela, sending a message and receiving a message happen at the same time. For example, if A is a sending statement in one process and B is a receiving statement in another process, then for the entire system, the state after executing *atomic {delay (x, y); A; B}* and the state after executing *atomic {A; delay (x, y); B}* are completely equivalent. In Section 6 we will present two experimental results, one for asynchronous channels and another for synchronous channels.

The body of the *if* statement in Figure 6 is a selection structure. The statement marked with "::" in this structure indicates that the statement will be executed randomly; that is, only one of the multiple *delay* statements in the *if* will be executed randomly. This can be done by doing the simulation of random transit time.

(2) The realization of the server role is shown in Figure 7.

In the server role process, *IsShareSKey(kas)* is statement whose function is to decide whether the parameter *kas* is a key shared with the server, so as to identify whether the message is sent to the server. *TsTimelyCheck(t)* is used to check whether the received message is fresh. *Atomic* is a reserved word in the Promela language, and the statements in the program block identified by *atomic* are not executed in cross-execution with the statements in other processes even in a concurrent environment but are executed in a step-by-step sequence. This mechanism can play a role in state compression, thereby alleviating the problem of state space explosion.

(3) Then comes the receiver role realization, as shown in Figure 8.

In Figure 8, *eval* is a matching function predefined by Promela, which is used to check whether the value sent from the channel is equal to the value in the parameter. If it is equal, the message is received and if it is not equal, it is not received. *eval(ShareSKey(b))* indicates that the key used to encrypt the message on the channel must be the shared key between the server and the receiver. This checking ensures that messages on the channel can only be received by the specified agent. *TsNewstCheck(t)* is used to check whether the timestamp in the message is the latest timestamp generated by the server. If it is the latest, it means that the message is within the valid time range. *Reci(a, b)* means that the receiver confirms that it has received the key sent by the initiator. The *assert* statement, which determines whether the value of its parameter is true, is used here to verify the freshness of the key in the received message.

(4) The fourth step is instantiation of roles, as shown in Figure 9.

In Figure 9, the *init* process is the main process of Promela. When the system starts, the *init* process will be executed first, which is equivalent to the *main* function in the C language. But before the system starts, the parameters of the process are assigned by real agent in the protocol and keyword *run* is used to activate each role process and the *Timers* process representing the time model in the system. So far, the entire protocol model has been built. The next step is to build the intruder model and put it into the protocol model to simulate the real insecure network environment.

```
proctype Service (mtype s)
{
  mtype a, b, kab, kbs, kas;
  timer t;
  timer real_delay;
  do
   :: atomic {c1?a, t, b, kab, kas;
        IsShareSKey (kas);
        TsTimelyCheck (t);
      }
      atomic{
        kbs = ShareSKey (b);
        if
         :: delay (SerT, delay_2);
          real_delay = delay_2;
         :: delay (SerT, delay_1) ;
            real_delay = delay_1;
        fi;
        SerTs_latest = T_global- real_delay;
        Ts = T_global- real_delay;
        c2!Ts, a, kab, kbs;
      }
  od
}
```

Figure 7: Process of server.

```
proctype Receiver (mtype b)
{
  mtype s, a, kab, kbs, t;
  s=S;
  atomic{
  c2?t, a, kab, eval (ShareSKey (b));
  TsNewstCheck (t);
  Reci (a, b);
  assert (Keyfresh);
 }
}
```

Figure 8: Process of receiver.

```
init {
  atomic
  {
    if
    :: run Initiator (A, B);
    :: skip;
    fi;
    run Intruders (I);
    run Service (S);
    run Receiver (B);
    run Timers ();
  }
}
```

Figure 9: Process of role instantiation.

4.4. Intruder Model. In model checking, the number of states increases exponentially with model complexity. If the traditional dynamic analysis method is used to analyze all possible knowledge of the intruder, the problem of state explosion is prone to occur because our model has more time factors than the general model. Therefore, the intruder model must be adjusted to avoid the problem.

Our solution is to refine the intruder's capabilities by using vulnerable channel priority, which is a simple but useful modeling method. The method includes two steps. The first step is to manually analyze and find out the vulnerable channel, that is, the channel where it is easier for the intruder to implement the intrusion. The second step is to model the intruder according to the identified vulnerable channels. It should be noted that we must gradually increase intruder capabilities in our model if no vulnerabilities are found until an attack is found or a state space explosion occurs. Compared with the method that intruders dynamically intercept messages on all channels, this modeling method can greatly reduce the number of state transitions in the process of model checking because the intruder capability in this method is not so strong at first. We take the WMF protocol as an example to describe the method. In WMF protocol, we discard the intruder's ability to intercept messages on the first channel, in which the initiator sends message to server, and retain the ability to intercept messages on the second channel. The reason is that, in practice, the server checks the timestamp much more strictly than the receiver and the intruder prefers to start to attack from places with lower security defenses. So, we chose the second channel as the vulnerable channel in the WMF protocol through simple analysis. Fortunately, we quickly found WMF protocol attacks using the vulnerable channel priority intruder model building method. The intruder's Promela model is shown in Figure 10.

We divide the intruder's capability model into two parts. The first part corresponds to the outermost *if* structure in Figure 10, and its function is to send any possible message into the channel using its own initial knowledge. The second part corresponds to the *do* loop structure in Figure 10. Its function is to make the intruder intercept and utilize messages in the communication channel. The intruder's ability to intercept and process messages is a loop operation that will not actively stop, so as to maintain constantly intercepting messages on the network so that the intruder can learn more knowledge. In addition, after intercepting the message, if the intruder has the key of the message, he can learn and store the knowledge in the message, so as to use these messages to replay or forge new messages at an appropriate time. If the intruder cannot decrypt the encrypted component in the message, the entire encrypted component can be forwarded to the channel or nothing can be done.

After the intruder model is established, the next section needs to put the protocol's security specification into the model to verify whether the security properties of the protocol are still satisfied in the network environment where the intruder exists.

```
proctype Intruder (mtype i)
{
   mtype agent, agent1, agent2;
   mtype x1, x2, x3, px1, px2, px3;
   timer t, pt, real_delay;
   mtype s=S, kis, k;
   if
     :: skip;
     :: atomic {
     if
          :: agent1=A; :: agent1=I; :: agent1=B;
      fi;
      if
          :: agent2=B; :: agent2=A
      fi;
          SessionKey (i, agent2, k);
          kis=ShareSKey (i);
      if
         :: delay (IniT, delay_2);
         :: delay (IniT, delay_1);
      fi;
       c1 ! agent1, Ts, agent2, k, kis;
     }
    fi;
  do
     :: atomic    {
        c2 ? t, x1, x2, x3 ;
        if
          :: skip;
          :: x3=ShareSKey (i) -> pt=t; px1=x1; px2=x2; skip;
        fi;
        if
          :: agent=B; :: agent=A; :: agent=I;
        fi;
        if
           :: delay (IntT, delay_2);  c1 ! agent, t, x1, x2, x3;
           :: delay (IntT, delay_1);  c1 ! agent, t, x1, x2, x3;
           :: delay (IntT, delay_2);  c1 ! agent, pt, px1, px2, x3;
           :: delay (IntT, delay_1);  c1 ! agent, pt, px1, px2, x3;
           :: skip;
        fi;
     }
  od;
}
```

Figure 10: Process of intruder.

5. Security Properties

The two security properties that the WMF protocol needs to satisfy are freshness and authentication. Freshness requires that the message received by the recipient must be fresh, which contains two meanings: the first meaning is that the message must be generated recently, which can be guaranteed by the timestamp checking mechanism in the model, so no verification is required. The second meaning is that the session key received by the receiver must be the fresh key generated by the current round of the protocol session, and we use an assertion to verify this property. Promela assertion is a function in the format *assert(expression)*, whose function is to determine whether the value of the *expression* is true. If false, SPIN will report an error and give a counterexample.

According to the definition of security property in Section 3.2, *assertion* can be used to verify the key freshness in WMF protocol as *assert (Keyfresh)* where the logical expression *Keyfresh* is completely defined as follows:

$$\begin{aligned} &\#\text{defineKeyfresh}\,((\text{T\_global} - \text{Tstart}) \\ &<= 2 * \text{delay\_2})\&\&((\text{T\_global} - \text{Tstart}) \\ &>= 2 * \text{delay\_1}). \end{aligned} \tag{7}$$

In this expression, *delay_1* and *delay_2* represent the minimum and maximum time consumed by the communication channel to transmit a message, respectively. The number *2*, which is equivalent to the value of N in Section 3.2, indicates that one round of WMF protocol has two messages containing the key transmitted through the channel. *Tstart* indicates the key generation time, which is also the time when the initiator initiates the session. *T_global* is the global clock variable, which is used here to indicate the moment when the message was received.

This assertion indicates that the time elapsed from the time the key is generated to the time it is received by the receiver must be within a reasonable transmission time range; otherwise the key is not a valid fresh key.

The WMF protocol also needs to complete the one-way authentication of the key receiving agent to the key generating agent. In order to use LTL to represent the authentication of the WMF protocol, we must define the following variables.

bool $InitAB = 0$, set to 1 when the agent A sends a message to the message channel;

bool $ReceAB = 0$, set to 1 when the agent B confirms receipt of the message from A. The LTL formula which indicates the authentication based on the above variables is as follows:

$$([]([]!\text{Reci}AB \| (!\text{ReciAB U InitAB}))). \tag{8}$$

This formula means that the receiver B can never acknowledge receipt of a message from initiator A until A initiates a session with B.

The assertion representing the key freshness and the LTL formula representing the authentication with the Promela model of the WMF protocol in Section 4 of this paper are imported into the model checking tool SPIN, and SPIN can automatically verify whether the protocol satisfies the security properties.

6. Experimental Results

In the Promela model of WMF protocol (called Model 1) established in Section 4, different processes communicate through synchronization channels. However, in practical applications, asynchronous channels are often used for communication between different processes. Therefore, we changed the message channel in Model 1 to an asynchronous channel with a channel capacity of 1, and the other parts remained unchanged to obtain a new Promela model (called Model 2). Using SPIN to verify these two models, the experimental results show that the models using these two message channels can successfully find the attack paths, which are shown in Figures 11 and Figures 12, of the

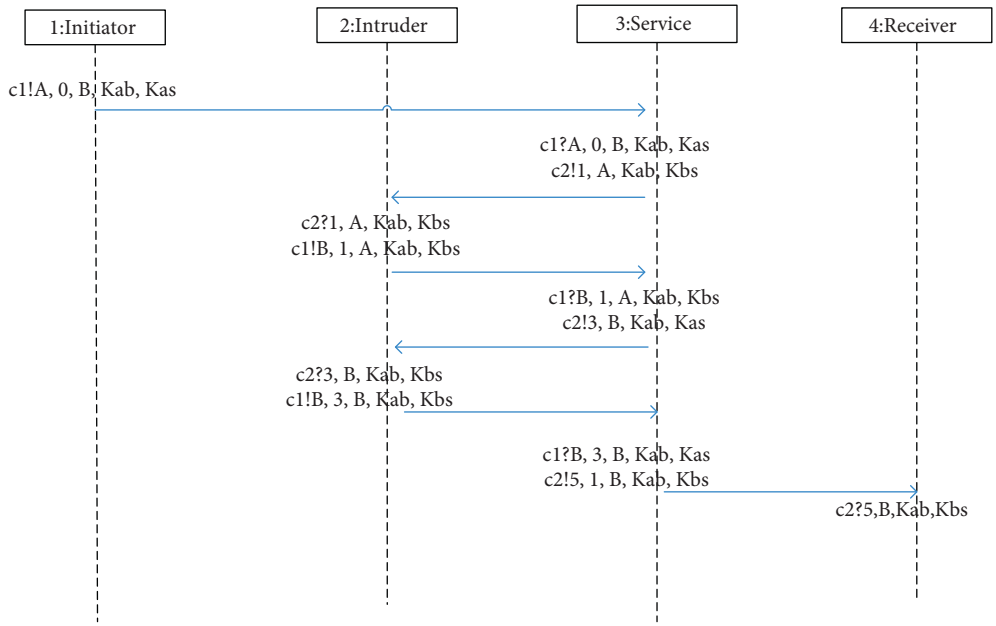

Figure 11: Attack path of WMF protocol violating key freshness.

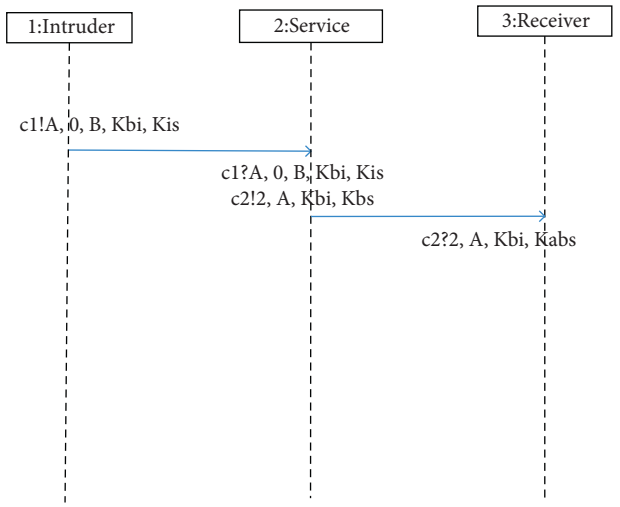

Figure 12: Attack path of WMF protocol violating authentication.

Table 2: Experimental data of WMF protocol.

WMF protocol model	Model 1 (using synchronization channel)		Model 2 (using asynchronous channels)	
Security properties	States	Transitions	States	Transitions
Authentication	323	398	361	448
Freshness	47	50	51	54

Note. The experimental results are obtained by using SPIN6.5.1, CPU Intel(R) Core (TM)i5-6300HQ (2.3 GHz), RAM 2 GB, and the operating system platform Ubuntu 18.04.6 virtual machine experimental environment. The parameter of the state search method is depth-first search, and the parameter value of the search depth is 10000 (default value).

Table 3: Experimental results of WMF protocol.

Verification results	Reference [22]	Reference [23]	This paper
Attack path 1	√	√	√
Attack path 2	—	—	√

counterexamples that are the same. The experimental data are shown in Table 2. The number of stored states can reflect the complexity of the model; the number of state transitions can reflect the efficiency of state search in model checking.

As can be seen from Table 2, our models have no state space explosion, and the number of stored states and state transitions is small, which is mainly due to the experimental strategy of vulnerable channel priority.

Table 3 shows the results of using different methods to find the attack path of the WMF protocol. It can be seen from the table that the methods of [22] and [23] can only find the attack path 1 of the WMF protocol, but the method in this paper can find both the attack path 1 and the attack path 2. We must state that an important premise of the attack is that the server in actual implementation of the protocol can infer the key, which is owned by the server, to the ciphertext from the message format.

Figure 11 shows the attack path of WMF protocol violating freshness (attack path 1). We denote by $I(X)$ that the intruder I impersonates the agent X; then, the attack path of violating key freshness is summarized as follows:

$$A->S:\ A, \{T_A, B, K\_AB\}K_{AS}. \tag{9}$$

Agent A sends a message with the key K_AB to server S;

$$S->I(B):\ \{T_S, A, K\_AB\}K_{BS}. \tag{10}$$

After the server S receives the message, it updates the timestamp and sends the message containing K_AB to the agent B with the identifier of A, but the message is intercepted by the intruder I;

$$I(B) - > S: B, \{T_S, A, K\_AB\}K_{BS}. \quad (11)$$

The intruder I pretends to be B, and sends the encrypted message that was intercepted last time with the identifier of B to the server S;

$$S - > I(A): \{T'_S, B, K\_AB\}K_{AS}. \quad (12)$$

S believes that the received message is a new protocol session message sent by agent B, so it adds the new timestamp into the message and sends the message to agent A, but the intruder I intercepts the message again;

$$I(A) - > S: A, \{T'_S, B, K\_AB\}K_{AS}. \quad (13)$$

The intruder I pretends to be A and sends the intercepted encrypted message to the server S with the identifier of A;

$$S - > B: \{T''_S, A, K\_AB\}K_{BS}. \quad (14)$$

S thinks that a new protocol session message is sent by A, so it updates the value of the timestamp and sends the message to B. After B receives the message, it determines that the timestamp is the latest, so B thinks that the key K_AB in the message is the latest.

But in fact, the key K_AB in the message is not the fresh key generated by the current round of the protocol, but the old key generated by the previous session. If B uses this key to encrypt important information and send it to A, agent A may think that the key has expired and refuse to receive it, which may cause adverse consequences.

Figure 12 shows the attack path of WMF protocol violating authentication (attack path 2); the attack path can be described as follows:

$$I(A) - > S: A, \{T_I, B, K\_IB\}K_{IS}. \quad (15)$$

The intruder I generates the key K_IB and encrypts it with the timestamp T_I and the key receiving agent B with K_{IS} to form a ciphertext block. Immediately I attaches the agent identification of A to form a message and then sends it to S;

$$S - > B\{T_S, A, K\_IB\}K_{BS}. \quad (16)$$

S determines that the message comes from A according to the identifier in the received message, so it updates the timestamp and encrypts it together with the agent identifier A and the key K_IB *to* form a message and then S sends the message to agent B. When receiving the message, B detects that the timestamp is the latest and then considers the message to be fresh. Finally, B determines that the key generator is A according to the identifier in the encrypted message.

But in fact, the agent A is faked by the intruder I; even the honest agent A does not participate in the operation of the protocol. If B encrypts important information with the received key K_IB and sends it to A, it will be intercepted and decrypted by an intruder I who pretends to be A. And if the intruder obtains the content of the message, there will be serious security problems.

7. Conclusion

The model checking technique and model checker SPIN are used to verify security protocols with timestamps and the model building method for security protocols with timestamps is introduced in this paper. We take the WMF protocol as an example to describe the modeling method in detail and the experimental results show that our method can successfully find the vulnerabilities of WMF protocol and there is no state space explosion problem. The results also show that our method is effective and it can provide a direction for the formal analysis of other security protocols with timestamp.

Since the time model represented by Promela in this paper is discrete and the time precision that can be represented is limited, the next work can be to try to use a more fine-grained time representation method to make our protocol model as identical as possible to the real-world protocol environment. In addition, another future work is to determine a unified principle for automatically finding out the vulnerable channels in protocols to help optimize the intruder's model.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61962020 and 61562026) and the Major Academic Discipline and Technical Leader of Jiangxi Province (Grant No. 20172BCB22015).

References

[1] S. Lee, H. Jeon, and G. Park, "Design of automation environment for analyzing various IoT malware[J]," *Tehnički Vjesnik*, vol. 28, no. 3, pp. 827–835, 2021.

[2] W. Nie and L. Liu, "A ring signature trust model for project review based on blockchain smart contract[J]," *Tehnički Vjesnik*, vol. 28, no. 2, pp. 347–356, 2021.

[3] G. Lowe, "An attack on the needham-schroeder public-key authentication protocol," *Information Processing Letters*, vol. 56, no. 3, pp. 131–133, 1995.

[4] R. Xue and D. Feng, "Formal analysis techniques and methods of security protocols [J]," *Chinese Journal of Computers*, vol. 9, no. 01, pp. 1–20, 2006.

[5] E. M. Clarke, O. Grumberg, D. Kroening, and S. l. Bernd-Holger, *Model checking[M]*, MIT press, Cambridge, MA, USA, 2018.

[6] M. Ben-Ari, *Principles of the Spin Model checker[M]*, Springer Science & Business Media, Berlin, Germany, 2008.

[7] P. Maggi and R. Sisto, "Using SPIN to Verify Security Properties of Cryptographic protocols," in *Proceedings of the International SPIN Workshop on Model Checking of Software*, pp. 187–204, Springer, Grenoble, France, April 2002.

[8] T. R. Andel and A. Yasinsac, "Automated Evaluation of Secure Route Discovery in MANET Protocols," in *Proceedings of the International SPIN Workshop on Model Checking of Software*, pp. 26–41, Springer, Los Angeles, CA, USA, August 2008.

[9] J. Chen, M. Xiao, K. Yang, W. Li, and X. Zhong, "Formal Analysis and Verification for Three-Party Authentication Protocol of RFID," in *Proceedings of the National Conference of Theoretical Computer Science*, pp. 46–60, Springer, Shanghai, China, October 2018.

[10] M. Xiao, D. Cheng, W. Li, Y. N. Li, X. Liu, and Y. Mei, "Formal analysis and verification of OAuth 2.0 protocol improved by key cryptosystems," *Chinese Journal of Electronics*, vol. 26, no. 3, pp. 477–484, 2017.

[11] N. Ben Henda, "Generic and efficient attacker models in SPIN," in *Proceedings of the 2014 International SPIN Symposium on Model Checking of Software*, pp. 77–86, San Jose, CA, USA, July 2014.

[12] S. Chen, H. Fu, and H. Miao, "Formal verification of security protocols using SPIN," in *Proceedings of the 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pp. 1–6, IEEE, Okayama, Japan, June 2016.

[13] T. Ninet, A. Legay, R. Maillard, T. Louis-Marie, and Z. Olivier, "Model checking the IKEv2 protocol using SPIN," in *Proceedings of the 2019 17th International Conference on Privacy, Security and Trust (PST)*, pp. 1–7, IEEE, Fredericton, NB, Canada, August 2019.

[14] P. Bedi and A. Dua, "Network steganography using the overflow field of timestamp option in an IPv4 packet," *Procedia Computer Science*, vol. 171, pp. 1810–1818, 2020.

[15] A. Tewari and B. B. Gupta, "Secure timestamp-based mutual authentication protocol for IoT devices using RFID tags," *International Journal on Semantic Web and Information Systems*, vol. 16, no. 3, pp. 20–34, 2020.

[16] S. Szymoniak, O. Siedlecka-Lamch, and M. Kurkowski, "Timed analysis of security protocols," in *Proceedings of the 37th International Conference on Information Systems Architecture and Technology–ISAT 2016–Part II*, pp. 53–63, Karpacz, Poland, September 2017.

[17] L. J. Xu, Y. Liu, and Y. Lu, X. Zeng, Y. Zhang, X. Li, A novel efficient MAKA protocol with desynchronization for anonymous roaming service in Global Mobility Networks," *Journal of Network and Computer Applications*, vol. 107, pp. 83–92, 2018.

[18] X. Li, T. Liu, and M. S. Obaidat, F. Wu, P. Vijayakumar, N. Kumar, A lightweight privacy-preserving authentication protocol for VANETs," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3547–3557, 2020.

[19] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18–36, 1990.

[20] R. Selvaraj, V. M. Kuthadi, S. Baskar, and R. Abhishek, "Creating security modelling framework analysing in internet of things using EC-GSM-IoT[J]," *Arabian Journal for Science and Engineering*, pp. 1–13, 2021.

[21] L. Chen and Z. Ye, "A security, privacy and trust methodology for IIoT[J]," *Tehnički Vjesnik*, vol. 28, no. 3, pp. 898–906, 2021.

[22] G. Delzanno and P. Ganty, "Automatic verification of time sensitive cryptographic protocols," *Tools and Algorithms for the Construction and Analysis of Systems*, in *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 342–356, Amsterdam, The Netherlands, April 2004.

[23] G. Jakubowska and W. Penczek, "Modelling and checking timed authentication of security protocols[J]," *Fundamenta Informaticae*, vol. 79, no. 3-4, pp. 363–378, 2007.

[24] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.