Automatic Verification of Security of Identity Federation Security Protocol Based on SAML2.0 with ProVerif in the Symbolic Model

Jintian Lu[1,2], Xudong He[1], Yitong Yang[1], Dejun Wang[1], and Bo Meng[1]
(Corresponding author: Bo Meng)
School of Computer Science, South-Central University for Nationalities[1]
Wuhan 430074, China
School of Data and Computer Science, Sun Yat-Sen University[2]
Guangzhou 510006, China
(Email: mengscuec@gmail.com)
(Received June 20, 2018; Revised and Accepted Nov. 22, 2018; First Online June 17, 2019)

Abstract

In recent years, several Identity Federation security protocols have been introduced to enhance the security of Identity authentication. Owning to the complexity, assessing security of Identity Federation security protocols has becoming a hot issue. Hence, in this study, we firstly review the development of formal methods on Identity Federation Security Protocol Based on SAML. And then, an Identity Federation Security Protocol Based on SAML is formalized with Applied PI calculus. After that, the formal model is translated into the inputs of ProVerif. Finally, we run ProVerif to analyze the security properties of Identity Federation Security Protocol Based on SAML. The result shows it has not secrecy, but it has some authentications. At the same time, we present a solution to address the security problems.

Keywords: Applied PI Calculus; Authentication; Formal Method; Security Protocol

1 Introduction

Identity Federation has been playing an increasingly important role in information security [2, 9, 26] and can allow the end users to use the same set of credentials to obtain access to multiple resources in different organization. Identity Federation security protocols typically include Microsoft U-Prove, OASIS SAML, and Liberty. But the OASIS Security Assertion Markup Language (SAML) is the emerging standard in this context and it is the most important technology to establish and manage Identify Federation. According to the related researches, the security of Identity Federation based on SAML2.0 has not been analyzed based on rigorous proofs and has been challenged by several analysis.

In order to obtain the strong confidence on security properties of security protocols [2, 8, 10, 15, 18, 25], the symbolic model and the computational model are introduced. Firstly, each model formally defines security properties of security protocol, and then propose methods for strictly proving and analyzing that whether given security protocols meet these requirements in adversarial environments or not. The computational model is too extreme complicated and difficult to get the support of automatic tools. In contrast, the symbolic model is considerably simpler than the computational model, hence proofs are also simpler, and can sometimes benefit from automatic tools support. For example: SMV, NRL, Casper, Isabelle, Athena, Revere, SPIN, Brutus, Coq [4, 7], ProVerif [6], Scyther [22, 24]. ProVerif is an automatic security protocol verifier and accepts the Applied PI calculus [27] as its input. It can process a lot of the different cryptographic primitives and an unbounded number of sessions of the security protocol in an unbounded message space. ProVerif has been tested on security protocols of the literature with very great results.

Therefore, in this paper, we use ProVerif to formally verify security properties of Identity Federation Security Protocol Based on SAML2.0 in the symbolic model.

2 Contribution

Several Identity Federation security protocols have been introduced in the recent years. Owning to the complexity, how to assess its security has become a challenging issue. Formal method is crucial to assess its security. So in this paper, we firstly review the development of the formal methods on Identity Federation Security Protocol Based on SAML 2.0 and apply the automatic tool developed by Blanchet to analyze its security properties. Hence,

firstly, Identity Federation Security Protocol Based on SAML is modeled with the Applied PI calculus. And then the model is translated into the inputs of ProVerif. Finally the translated model is performed by automatic tool ProVerif. The result shows that it has not secrecy of some keys, but it has some authentications based on the model implemented by us. At the same time, we present a solution to address the security problems.

We use the Applied PI calculus to model Identity Federation Security Protocol Based on SAML according to the fact that the Applied PI calculus allows the modeling of relations between data in a simple and precise manner using equational theories over term algebra. The general analysis model is presented in Figure 1.

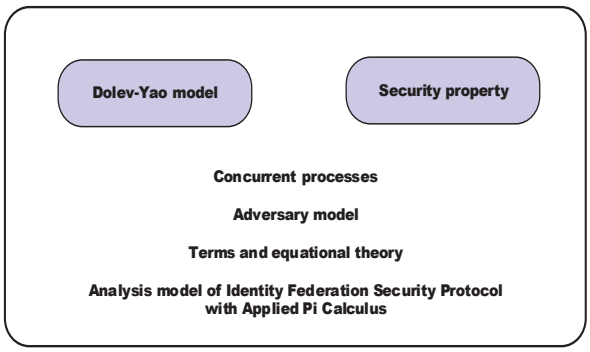

Figure 1: Analysis model of identity federation security protocol based on SAML with the applied PI calculus

There, the security properties model is equivalence between processes, while the attacker is modeled as an arbitrary process running in parallel with the protocol process representing the adversary model, which is the parallel composition of the protocol participants processes. The considered attacker is stronger than the basic Dolev-Yao attacker since it can exploit particular relations between the messages by using particular equational theories stating the message relations. Figure 2 presents the automatic verification of Identity Federation Security Protocol Based on SAML 2.0.

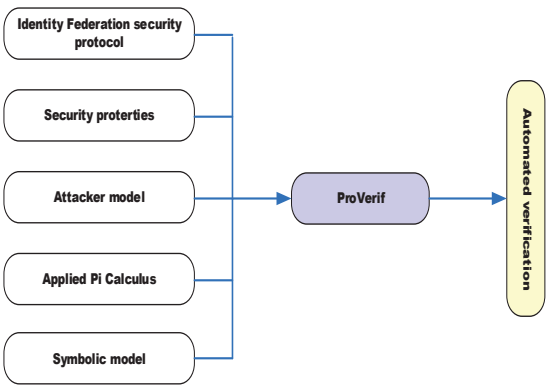

Figure 2: Automatic verification of identity federation security protocol based on SAML

3 Related Work

Here, we present the state-of-art of the analysis of the Identity Federation Security Protocol Based on SAML. Armando *et al.* [1] mechanically analyzed SAML SP-Initiated SSO profile with the model checker SATMC based on HLPSL++ as the specification language and found a severe security vulnerability that allows a dishonest service provider to impersonate a user at another service provider. Cabarcos *et al.* [20] proposed a generic extension for the SAML standard for facilitating the creation of federation relationships in a dynamic way between prior unknown parties. But its security is not proved by formal methods. ter Beek *et al.* [5] used model checker PaMoChSA to analyze the security aspects of the Identity Federation protocol proposed by Telecom Italia and found a man-in-the-middle attack [3, 12, 14, 16, 19].

Ferdous and Poet [17]presented a simple approach based on SAML Profile and allow users to create federations using SAML between two prior unknown organizations in a dynamic fashion. Ghazizadeh *et al.* [11]presented an overview on Identity Federation in the cloud computation environment and pay a special attention on the identity theft issue. Cabarcos *et al.* [21]introduced the IdMRep which is a decentralized reputation-based mechanism which allows trust relationships to be established on-demand driven by user's needs. While they do not analyze its security.

Wang *et al.* [13] presented a browser-based mutual authentication for Federated Identity management to protect the token mutually by binding the client certificate and using TLS protocol. Apart from that they also analyze the security in the Random model and prove that it supports authentication. Saklikar and Saha [23] proposed the VoIP Identity Federation Framework which can make a user to establish Identity Federation and the assertion of any relevant Identity information from one VoIP context to another based on the federate-out and federate-in primitives. While they do not prove its security with formal methods.

4 Applied PI Calculus and ProVerif

The Applied PI calculus is a formal language for describing concurrent processes and their interactions based on Dolev-Yao model. Applied PI calculus is an extension of the PI calculus that inherits the constructs for communication and concurrency from the pure PI calculus. It preserves the constructs for generating statically scoped new names and permits a general systematic development of syntax, operational semantics equivalence and proof techniques. At the same time, there are several powerful automatic tool supported the Applied PI calculus, for example, ProVerif. The Applied PI calculus with ProVerif has been used to study a variety of complicated security protocols.

In the Applied PI calculus, terms consists of names variables and signature $\sum$. $\sum$ is the set of function symbols, each with an arity. Terms and function symbols are sorted, and of course function symbol application must respect sorts and arties. Typically, we let a,b and c range over channel names. Let x,y and zrange over variables, and u over variables and names.over variables and names. We abbreviate an arbitrary sequence of terms $M_1, \cdots, M_i$ to $\widetilde{M}$. In applied PI calculus, it has plain processes and extended processes. Plain processes are built up in a similar way to processes in the PI calculus, except that messages can contain terms and that names need not be just channel names. The process 0 is an empty process. The process Q|p is the parallel composition of P and Q. The replication !p produces an infinite number of copies of P which run in parallel. The process vn.p firstly creates a new, private name then executes as P.The abbreviation $v\widetilde{n}$ is a sequence of name restrictions $vn_1, \cdots, vn_i$. The process in (u, x). P receives a message from channel u,and runs the process P by replacing formal parameter x by the actual message. We use in(u, $\widetilde{M}$). P is the abbreviation for the output of terms $N_1, \cdots, N_i$. The conditional construct if M=N then P else Q runs that if M and N are equal, execute P, otherwise execute Q.

Extended processes add active substitutions and restriction on variables. We write { M / x } for active substitution which replaces the variable x with the term M. The substitution typically appears when the term M has been sent to the environment, but the environment may not have the atomic names that appear in M; The variable x is just a way to refer to M in this situation.

In general an event is used to mark important steps of the security protocol under study but do not otherwise affect its behavior. It can be used to record the context of the sending or receiving message in security protocol. In the applied PI calculus, event event(M) just outputs message M through a special channel. So event event(M) does not reveal M to the adversary. Hence, the execution of the process P after inserting events is the execution of P without events, plus the recording of event(M). The process the event(M). P executes the event(M), then executes P.

ProVerif is an automatic cryptographic protocol verifier based on a representation of the protocol by Horn clauses and the Applied PI calculus. It can handle many different cryptographic primitives, including shared- and public-key cryptography, hash function, and Deffie-Hellman key agreements, specified both as rewrite rules and as equations. It can also deal with an unbounded number of sessions of the protocol and an unbounded message space. When ProVerif cannot prove a property, it can reconstruct an attack, that is, an execution trace of the protocol that falsifies the desired property. ProVerif can prove the following properties: secrecy, authentication and more generally correspondence properties, strong secrecy, equivalences between processes that differ only by terms. ProVerif has been tested on protocols of the literature with very encouraging results. When ProVerif cannot prove a security property, it can reconstruct an attack, ProVerif can prove secrecy, authentication and more generally correspondence properties, strong secrecy, equivalences between processes that differ only by terms.

5 Identity Federation Security Protocol Based on SAML2.0

Identity federation security protocol is mainly made up of three principles: User Agent (UA), Service Provider (SP) and Identity Provider (IdP). Generally there is a Single Sign-On (SSO) service component in the identity provider. SSO allows the end users to provide their credentials once and obtain access to multiple resources. In other words, the identity provider can provide the SSO service. Service provider has the components of access check and assertion consumer service. Hence it has the ability to check and verify the identity of user and assertion consumer. User agent can be browser which is the agent of the users. There are two models in identity federation security protocol based on SAML. One is IdP -initiated model. The other is SP-initiated model. Figure 3 describes identity federation security protocol based on SP-initiated SAML 2.0 using HTTP.

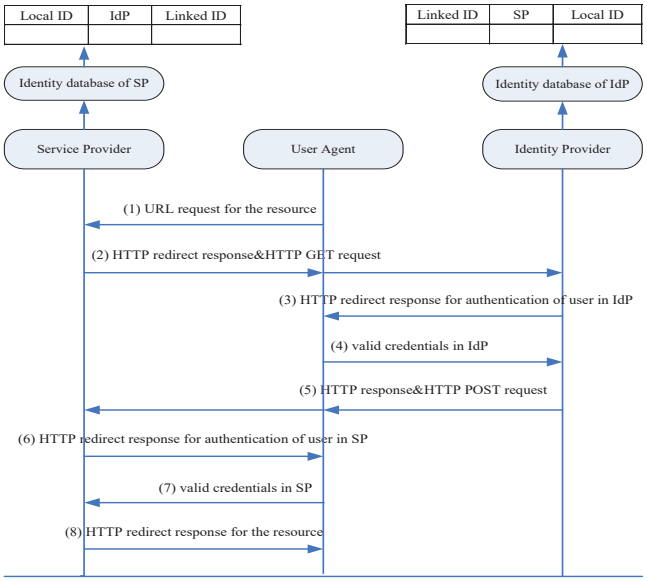

Figure 3: Identity federal security protocol based on SP-initiated SAML 2.0 using HTTP

Apart from that we assume that the service provider has the digital signature public key PU_{sp} and and private key PR_{sp} and the identity provider has two pairs of digital signature public key and private key (PU^1_{idp}, PR^1_{idp}), (PU^2_{idp}, PR^2_{idp}). Service provider and identity provider has an identity database in which it has the information of local ID, SP and linked ID, respectively. The linked ID is the identifier that is used to establish the federation between the local ID in SP identity database and

local ID in IdP database. Apart from that, the identity federation security protocol based on SP-initiated SAML provides the authentication from service provider to user agent and from identity provider to user agent.

The identity federation security protocol based on SP-initiated SAML 2.0 includes eight messages exchanged among service provider, user agent and identity provider.

$$\left\{\begin{array}{l}\text{URLrequest}\\ \text{fortheresource}\end{array}\right\}:=\left\{\begin{array}{l}\text{URLrequest}\\ \text{foraresource}\end{array}\right\} \quad (1)$$

The user agent generates Message (1) which is used to request a target resource that is a secured resource at the service provider and sends it to the service provider.

$$\left\{\begin{array}{l}\text{HTTPredirectresponse}\\ \text{\&HTTPGETrequest}\end{array}\right\}:=\left\{\begin{array}{l}\text{URI}||\text{SAML}-\\ \text{Request}\\ ||\text{RelayState}\end{array}\right\}$$

$$\text{SAMLRequest}:=\left\{\begin{array}{l}\text{ID[Required]}||\\ \text{Version[Required]}\\ ||\text{IssueInstant}\\ \text{[Required]}\\ ||<\text{saml}:\text{Issuer}>\\ \text{[Optional]}\\ ||<\text{cds}:\text{Signature}>\\ \text{[Optional]}\\ ||<\text{NameIDPolicy}>\\ \text{[Optional]}\end{array}\right\} \quad (2)$$

If the parameters of SAMLRequest and RelayState are present in Message (1), the user agent has already been verified by the identity provider and can access the resource in service provider. Here we assume that the parameters of SAMLRequest and RelayState are not included in Message (1). Hence service provider constructs Message (2) and sends it to identity provider by means of service provider. Message (2) mainly consists of URI, SAMLRequest and RelayState parameters. The parameter URI is the address of SSO service component and is generated by service provider. The parameter SAMLRequest is URL-encoded <AuthnRequest> element in SAML and is also generated by service provider. <AuthnRequest> element is used to authenticate the user agent and is mainly composed of ID, Version, IssueInstant, <saml:Issuer> and <cds: Signature> elements. <cds: Signature> element is used to store the digital signature of the <AuthnRequest> element. The digital signature of the <AuthnRequest> element is generated with the private key PR_{sp} of service provider. The parameter RelayState is used to describe the state information maintained at the service provider, for example, URL in Message (1). Apart from that, SP sets the AllowCreate attribute on the NameIDPolicy element to 'true' value to allow the IdP to generate a new identifier for the user that is not already exist.

$$\left\{\begin{array}{l}\text{HTTPredirect}\\ \text{responsefor}\\ \text{authenticationinIdP}\end{array}\right\}:=\left\{\begin{array}{l}\text{HTTPredirect}\\ \text{responsefor}\\ \text{useragent}\end{array}\right\} \quad (3)$$

The SSO component in the identity provider uses the public key PU_{sp} of service provider to verify the digital signature stored in the <cds: Signature> element which is included in the <AuthnRequest> element. If the verification is successful, the identity provider executes a security check. If the user agent does not have a valid logon security context, the identity provider requires the user to provide the valid logon credentials made up of usernameIdP and passwordIdP to be verified by the the identity provider. Thus the identity provider generates Message (3) and sends it to user agent.

$$\left\{\begin{array}{l}\text{Validcredentials}\\ \text{inIdP}\end{array}\right\}:=\left\{\begin{array}{l}\text{usernameIdP}\\ ||\text{passwordIdP}\end{array}\right\} \quad (4)$$

The user agent receives Message (3) and generates Message (4) which is made up of usernameIdP and passwordIdP and sends it to the identity provider through HTTP protocol.

$$\left\{\begin{array}{l}\text{HTTPresponse\&}\\ \text{HTTPPOSTrequest}\end{array}\right\}:=\left\{\begin{array}{l}\text{SAMLResponse}\\ ||\text{RelayState}\end{array}\right\}$$

$$\text{SAMLResponse}:=\left\{\begin{array}{l}\text{ID[Required]}||\\ \text{InResponseTo}\\ \text{[Required]}||\text{Version}\\ \text{[Required]}\\ ||\text{IssueInstant}\\ \text{[Required]}||\\ \text{Destination}\\ \text{[Optional]}\\ ||<\text{saml}:\text{Issuer}>\\ \text{[Optional]}||\\ <\text{cds}:\text{Signature}>\\ \text{[Optional]}||\\ <\text{Status}>\text{[Required]}\\ ||<\text{saml}:\text{Assertion}>||\\ <\text{Extensions}>\\ \text{[Optional]}\end{array}\right\} \quad (5)$$

$$<\text{saml}:\text{Assertion}>:=\left\{\begin{array}{l}\text{Version[Required]}||\\ \text{ID[Required]}\\ ||\text{IssueInstant}\\ \text{[Required]}\\ ||<\text{Issuer}>\\ \text{[Required]}\\ ||<\text{ds}:\text{Signature}>\\ \text{[Optional]}\\ ||<\text{AuthnStatement}>\\ ||<\text{Subject}>\\ \text{[Optional]}\end{array}\right\}$$

$$<\text{AuthnStatement}>:=\left\{\begin{array}{l}\text{AuthnInstant}\\ \text{[Required]}||\\ <\text{AuthnContext}>\\ \text{[Required]}\end{array}\right\}$$

When the identity provider receives Message (4), it firstly checks the validity of the credential of the user, which are usernameIdP and passwordIdP. If the verification is successful, then the SSO Service in the identity provider checks whether usernameIdP in its identity

database is or not and whether the AllowCreate attribute is true or not. If all is true, it creates a persistent name identifier SPandIdP, which is stored in the element persistentid in element <Extensions>, to be used for the session at the service provider. The persistent name identifier SP and IdP is used to link the account username IdP in the identity provider and username SP in the service provider. Apart from that, the identity provider produces Message (5) which is made up of SAMLResponse and RelayState parameters. The parameter RelayState is gotten through the service provider. The parameter SAMLResponse is mainly composed of ID, InResponseTo,Version, IssueInstant, Destination, <saml: Issuer>, <cds: Signature>, <Status> and <saml: Assertion>. <saml: Assertion> is the most important element in SAML Response. The local logon security context generated by the identity provider is stored in the SAML Assertion <saml: Assertion> element. The content in InResponseTo element is identical to the content in the ID element in <AuthnRequest> element. The digital signature of SAMLResponse generated with the private key PR^1_{Idp} of IdP is stored in <cds:Signature> element. <saml: Assertion> element is mainly composed of Version, ID, IssueInstant, <Issuer>, <ds: Signature>, <subject> and <AuthnStatement>. Among these elements the <ds: Signature> element is important because the digital signature of <saml:Assertion> generated with the private key PR^2_{Idp} is stored in <ds: Signature> elements. The authentication context information is stored in <AuthnStatement> element which is composed of AuthnInstant and <AuthnContext> elements. The usernameIdP is stored in the element <subject>. After Message (5) is produced, then it is sent to the service provider. Assertion Consumer Service component in the service provider will process Message (5).

$$\left\{\begin{array}{l}\text{HTTPredirect}\\ \text{responsefor}\\ \text{authentication}\\ \text{inSP}\end{array}\right\} := \left\{\begin{array}{l}\text{HTTPredirect}\\ \text{responseforuser}\\ \text{agent}\end{array}\right\} \quad (6)$$

When Message (5) arrives at the service provider, Assertion Consumer Service will processes it. Firstly, it uses the public key PU^1_{Idp} of the identity provider to verify the digital signature of <Response> stored in the element <cds: Signature>, and then uses the public key PU^2_{Idp} of the identity provider to verify the digital signature of <saml: Assertion> in <ds: Signature> element. Secondly, the service provider generates the local logon security context using the information stored in <saml: Assertion> element. Thirdly, the supplied name identifier SPandIdP is then used to check whether a previous federation has been established in the service provider identity database. If no federation exists for the persistent identifier in the assertion, then the service provider needs to determine the local identity to which it should be assigned. Finally, service provider sends Message (6) HTTP redirect response to user agent to challenge the usernameSP at the service provider.

$$\left\{\begin{array}{l}\text{validcredentials}\\ \text{inSP}\end{array}\right\} := \left\{\begin{array}{l}\text{usernameSP}\\ ||\text{passwordSP}\end{array}\right\} \quad (7)$$

When Message (6) arrives at user agent, the user provides valid credentials and identifies his account at the service provider as usernameSP. The persistent name identifier SPandIdP is then stored and registered with the usernameSP account along with the name of the identity provider that created the name identifier.

$$\left\{\begin{array}{l}\text{HTTPredirect}\\ \text{responsefor}\\ \text{theresource}\end{array}\right\} := \left\{\begin{array}{l}\text{HTTPredirect}\\ \text{response}\end{array}\right\} \quad (8)$$

After the service provider receives Message (7) which is made of usernameSP and passwordSP and makes a verification of the identity of user agent, If the verification is successful, a local logon security context is generated for user usernameSP. Apart from that, the federation is established between the usernameSP and usernameIdP through the persistent identifier SPandIdp in the service provider identity database. Finally the service provider generates Message (8) for the user agent for the desired resource. If the access check passes, the desired resource is returned to the browser.

6 Formalize Identity Federation Security Protocol Based on SAML 2.0 Using the Applied PI Calculus

6.1 Function and Equational Theory

The functions and equational theory are introduced in this section. We use the Applied PI calculus to formalize Identity Federation security protocol based on SAML 2.0. We model cryptography in a Dolev-Yao model as being perfect. Figure 4 describes the functions and the equational theory in the Identity Federation security protocol based on SAML.

$$\left\|\begin{array}{l}\text{fun sign(x,PR).}\\ \text{fun PU(c).}\\ \text{fun PR(c).}\\ \text{fun decsign(x,PU)}\\ \text{fun versign(y,PU)}\\ \text{equation versign(sign(x,PR),PU)=true.}\\ \text{equation decsign(sign(x,PR),PU)=x}\end{array}\right\|$$

Figure 4: The functions and the equational theory

.

Digital signature is modeled as being signature with message recovery, i.e. the signature itself contains the

signed message which can be extracted using the function. Digital signature algorithm includes the generation signature algorithm sign(x,PR) sign the message x with private key PR and the verification algorithm versign(y,PU) verify the digital signature y with public key PU. And the decsign(x,PU) recover the message from the digital signature x with the public key PU. The function PU(c) accepts private value c as input and produces public key as output. The function PR(c) accepts private value c as input and produces private key as output.

6.2 Process

The complete formal model of Identity Federation security protocol based on SAML 2.0 in the Applied PI calculus is given in Figures 5, 6, 7 and 8, which report the basic process include main process, user agent process, service provider process and identity provider process forming the model of Federation security protocol based on SAML. The main process IFSAML in Figure 5 sets up the process User Agent, Service Provider and Identity Provider.

```
IFSAML ≜
(!User Agent | !Service Provider | !Identity Provider)
```

Figure 5: Main process

The process User Agent is modeled using the Applied PI calculus in Figure 6.

```
User Agent ≜                                        (*User Agent(UA) process*)
 new url; new finish;
 out(pub,url);                                      (*UA sends the message1 to SP*)
 in(pub,httpgetrequest);                            (*UA rceives the message2 form SP*)
 let (uria,samlrequesta,relaystatea)=httpgetrequest in
 if relaystatea=url then out(pub,httpgetrequest);   (*UA sends the message2 to IdP*)

 in(pub,m3);                                        (*UA receives the message3 from IdP *)
 let reauthuseridp=m3 in
 new authuseridp;
 if authuseridp=reauthuseridp then
 let secretX= passwordidp in
 let valididp=(usernameidp,passwordidp) in
 out(pub,valididp);                                 (*UA sends the message4 to IdP*)

 in(pub,m5);                                        (*UA receives the message5 from IdP*)
 let httppostrequest=m5 in
 let (samlresponsea,responserelaystatea)=httppostrequest in
 if responserelaystatea=url then out(pub,httppostrequest); (*UA sends the message5 to SP*)

 in(pub,m7);                                        (*UA receives the message6 from SP*)
 let reauthusersp=m7 in
 if authusersp=reauthusersp then
 let secretY= passwordsp in
 let validsp=(usernamesp,passwordsp) in
 out(pub,validsp);                                  (*UA sends the message7 to SP*)

 in(pub,m8);if m8=resourse then out(pub,finish).(*UA receives the message8 form SP*)
```

Figure 6: Server agent process

Firstly, the User Agent produces the target resource address url by the statement new url and sands sends it to the service provider through the public channel pub. At the same time it also generates the information finish by y the statement new finish which shows that the protocol ends. After that, the User Agent receives the message httpgetrequest using the public channel pub by the statement in (pub, httpgetrequest).And then it extract the elements uria,samlrequest, relaystate from the message httpgetrequest the item uria is the address of SSO service component and is generated by service provider. The item samlrequest is URL-encoded <AuthnRequest> element in SAML and is also generated by service provider. The item relaystate the state information maintained at the service provider. User Agent compare the value relaystate with url. If they are equal then User Agent forwards the message httpgetrequest to the process Identity Provider through the public channel pub.

The User Agent receives Message m3 from the process Identity Provider through the public channel pub.Then it extracts the message reauthuseridp which shows that the user should provide the valid logon credentials. After that, the User Agent provides the username usernameidp and password passwordidp through valididp=(usernameidp,passwordidp).And also it sends valididp to Identity provider process by the public channel pub.

And then it receives Message m5 from the public channel c which is sent from the Identity provider process. The User Agent gets the message samlresponsea and responseerelaystatea from httpposttrequest. samlresponsea is mainly composed of ID, InResponseTo,Version, IssueInstant, Destination, <saml: Issuer>, <cds: Signature>, responserelaystatea is the target resource address. If the responserelaystatea is equal to url, and then the message httpposttrequest is sent to the Service Provider through the public channel pub.

After that, the User Agent receives message m7 from the Service Provider from the public channel c. Then it gets the message reathuserp which shows the user should provide the username and password. And then it generates his username usernamesp and password passwordsp and construct the message secrectY. The user Agent sends the message validsp through the public channel pub to the Servicer Provider.

Finally, it receives Message m9 through the public channel pub. If Message m9 is equal to resource, and then it sends the message finish from the public channel pub. The protocol ends.

The Service Provider process in Figure 7 receives Message (1) urlx from the public channel pub. In order to construct Message (2), firstly, it generates ID id, Version version, IssueInstant issueinstant, <saml: Iusser> iussuer and nameidppolicy nameidpolicy using the statements: new id; New version; New issuestant; New issuer; New nameidpolicy. And then it uses the digital signature function sign() to generate the digital signature signature of id, version, issuestant, iussuer,

nameippolicy with the Service Provider's private key PR(keysp). Then the SAMLRequest samlrequest is produced though let samlrequest=(id, version, issuestant, issuer, nameipdolicy) in . The SAMLRequest samlrequest mainly consists of id, version, issuestant, issuer, nameipdolicy. Finally uri,samlrequest,relaystate are used to construct Message (2) httpredirectresponse which is sent to the User Agent through the public channel pub.

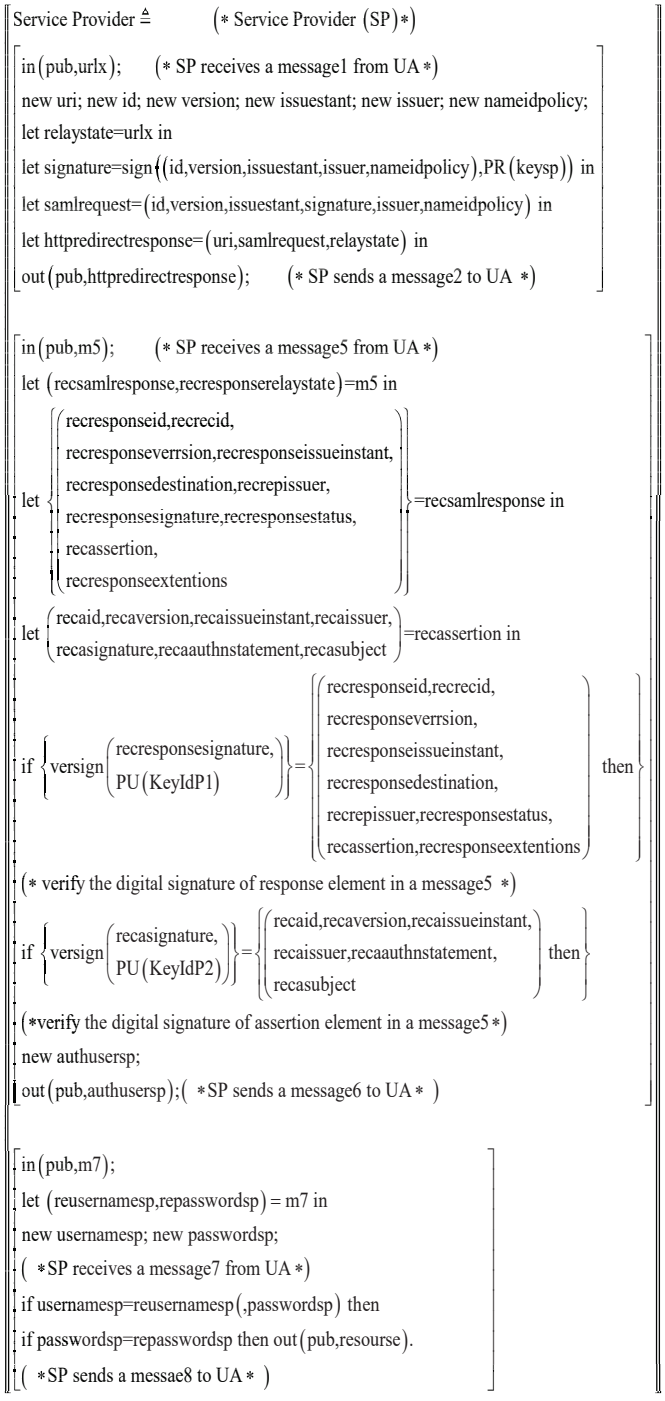

Figure 7: Server provider process

After that, it receives Message (5) from the User Agent process and gets the SAMLResponse recsamlresponse and RelayState recresponserelaystate form Message (5). Based on the SAMLResponse recsamlresponse, it generates ID recresponseid, InResponseTo recrecid, Version recresponseverrion, IssueInstantre recresponseissueinstant, Destination recresponsedestination, <saml: Issuer> recrepissuer, <cds: Signature> recresponsesignature, <Status> recresponsestatus and <saml: Assertion> recassertion. From the <saml: Assertion> element recassertion, Version recaversion, ID recaid, IssueInstant recaissueinstant, <Issuer> recaissuer, <ds: Signature> recasignature, <subject> recasubject and <AuthnStatement> recaauthnstatementare gotten. After that, the digital signature of <cds: Signature> recresponsesignature is verified by the function verign (recresponsesiganture, PU(KeyIdp1)) with the public key PUIdP1 of the Identity Provider. At the same time the digital signature of <ds: Signature> recsiganture is verified by the function versign (recsignature, PU(KeyIdp2)) with the public key PUIdP2 of the Identity Provider. If the two digital signature are all successful, the HTTP redirect response authusersp is generated and is sent to the User Agent through the public channel pub.

When Service Provider process receives Message m7 from the public channel pub, the usernameSP usernamesp and passwordSP passwordsp and makes a verification of the identity of user agent. If the verification is successful, then Service Provider generates Message (8) resource for the User Agent for the desired resource through the public channel pub.

The Identity Provider process in Figure 8 generates the elements responseid, responseverrion, responseissueinstant, responsedetination, aid, aversion, aissuestant, aissuer,aauthnstatement,asubject. And then it receives message m2 through the public channel pub. The Identity Provider process gets the elements URI recuri, SAMLRequest recsamlrequest and RelayState recrelaystate from Message m2 through the public channel pub. After that it extracts the elements ID recid, Version recversion, IssueInstant recissuestant, <saml: issue> recissuer and <cds: Signature> recsignature and NameID policy recnameidpolicy from the element SAMLRequest recsamlrewuest. Then, the Identity Provider process verifies the digital signaturerecsignature using the function versign (recsignature, PU(Keysp)) with the public key PU(Keysp) of Service Provider. If If the verification is successful, it generates message3 authuseridp which shows that the user should provide the valid logon credentials made up of usernameIdP and passwordIdP to be verified by the IdP. Thus the Identity Provider process sends Message (3) authuseridp to user agent process through the public channel pub.

After that, the Identity Provider process receives Message (4) m4 from the public channel pub. And then it extracts the usernameIdP usernameidp and passwordIdP passwordidp of the User Agent. It checks the validity of the credential of the user, which are usernameIdP and passwordIdP. If the verification is ok, it creates a per-

Table 1: The authentications

Non-Injective agreement	Authentications
ev:endauthUSERIDP(x)− >ev:geginauthUSERIDP(x)	Identity Provider authenticates User Agent
ev:endauthUSERSP(x)− > ev:geginauthUSERSP(x)	Server Provider authenticates User Agent
ev:endauthSAMLREQ(x) − > ev:beginauthSAMLREQ(x)	Identity Provider authenticates Server
ev:endauthSAMLRSP(x) − > ev:beginauthSAMLRSP(x)	Service Provider authenticate Identity Provider

sistent name identifier SPandIdP, which is stored in the element persistentid in element <Extensions>, to be used for the session at the service provider.

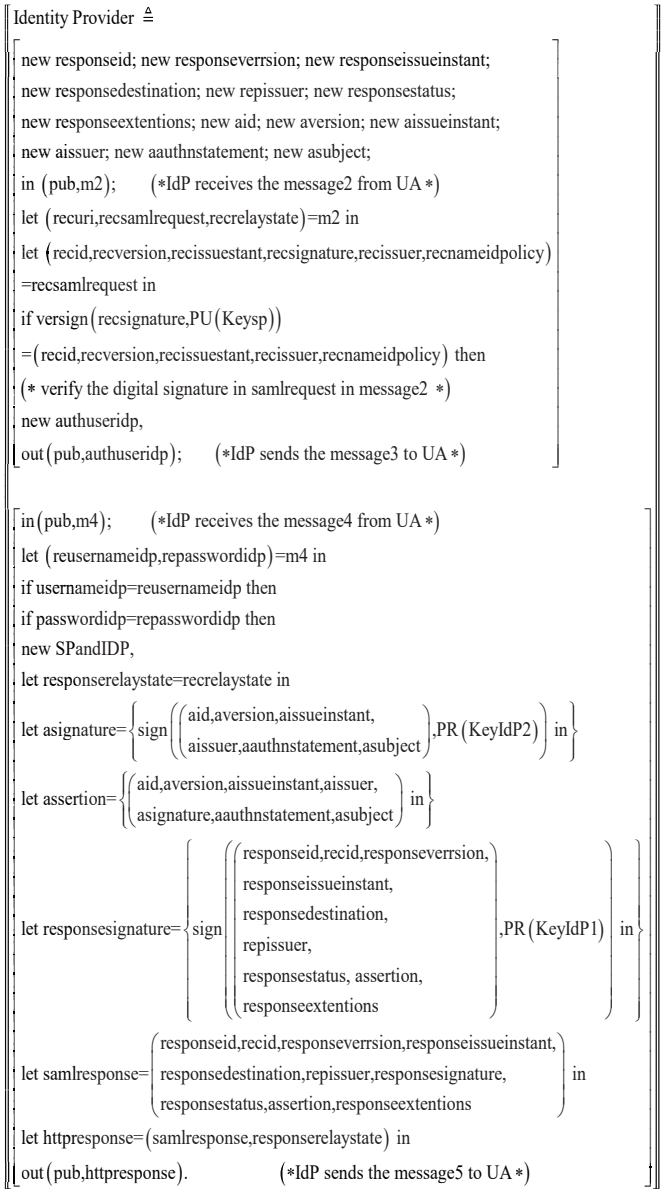

Figure 8: Identity provider process

Apart from that, the <ds: Signature> element asignature is produced by the digital signature function sign() with the inputs of <saml: Assertion> (aid, aversion, aissueinstant, aissuer, aathnstatement, asubject) and the private key PR(KeyIdP2) of Identity Provider. The <saml: Assertion> element assertion is mainly composed of Version aservsion, ID aid,IssueiInstant aissueinstant, <Issuer> aissuer, <dc: Signature> asignature, <subject> asubject and <AuthnStatement> aauthnstatement. At the same time the element <cds: Signature> responsesignature is generated by the digital signature function sign() with the inputs of (responseid,recid, responseserverrion, responseissueinstant, responsedestination,repissuer,reponsestatus, assertion,responseextentions) and the private key PR(KeyIdP1) of Identity Provider. Finally Message (5) httpresponse is generated which is made up of SAMLResponse samlresponse and RelayState responserelaystate parameters. The parameter SAMLResponse samlresponse is mainly composed of ID responseid, InResponseTo,Version responseverrion, IssueInstant responsedestination, <saml: Issuer> repissuer, <cds: Signature> responsesignture, <Status> responsestatus and <saml: Assertion> assertion and Message (5) httpresponse is sent to the user agent process through the public channel pub.

7 Automatic Verification of Secrecy and Authentications with ProVerif

Here we use the statements query attacker:secretX in ProVerif to verify the secrecy of which is the password of passwordidp the User Agent to assess the Identity Provider and query attacker:secretX is used to verify the secrecy of passwordidp to assess the Service Provider.

ProVerif uses the non-injective agreement to model the authentication. So we use query ev: event one–¿ ev:event two to model the authentication. It is true when if the event one has been executed, then the event event two must have been executed (before the event one). Here we use the non-injective agreement to model the authentications showed in Table 1 .

ProVerif can take two formats as input. The first one is in the form of Horn. The second one is in the form of a process in an extension of the Applied PI calculus. In both cases, the output of the system is essentially the same. In this study we use the Applied PI calculus as the input of ProVerif. In order to prove the authentication in Identity Federation security protocol based on SAML. The model using the Applied PI calculus is needed to be translated into the syntax of ProVerif and generated the

ProVerif inputs in extension of the PI calculus. Figures 9, 10, 11, 12, 13 and 14 are the inputs for Identity Federation security protocol based on SAML 2.0. We use the ProVerif to run the input for Identity Federation security protocol based on SAML 2.0 showed in Figure 9, 10, 11, 12, 13 and 14.

```
free pub.
free authuseridp,authusersp.
free usernamesp,passwordsp,usernameidp,passwordidp,resourse.
fun sign/2.
fun PU/1.
fun PR/1.
fun versign/2.
fun decsign/2.

equation versign(sign(x1,PR(y1)),PU(y1))=true.
```

Figure 9: The functions and equation in ProVerif

```
query attacker:secretX; (* the secrecy of passwordidp *)
query attacker:secretY. (* the secrecy of passwordsp *)

query ev:endauthUSERIDP(x) → ev: beginauthUSERIDP(x).
 (* Identity Provider authecticates User Agent *)
query ev:endauthUSERSP(x) → ev: beginauthUSERSP(x).
(* Service Provider authecticates User Agent *)
query ev:endauthSAMLREQ(x) → ev: beginauthSAMLREQ(x).
(* Identity Provider authecticates Service Provider *)
query ev:endauthSAMLRESP(x) → ev: beginauthSAMLRESP(x).
(* Service Provider authecticates Identity Provider *)
```

Figure 10: Query secrecy and authentications in ProVerif

```
let processuseragent =                    (*User Agent(UA) process *)
 new url; new finish;
 out(pub,url);  (*UA sends the message1 to SP *)
 in(pub,httpgetrequest);    (*UA rceives the message2 form SP *)
 let (uria,samlrequesta,relaystatea)=httpgetrequest in
 if relaystatea=url then out(pub,httpgetrequest);
 (*UA sends the message2 to IdP *)

 in(pub,m3);       (*UA receives the message3 from IdP *)
 let reauthuseridp=m3 in
 new authuseridp;
 if authuseridp=reauthuseridp then
 let secretX= passwordidp in
 let valididp=(usernameidp,passwordidp) in
 event beginauthUSERIDP(valididp);
 out(pub,valididp);
 (*UA sends the message4 to IdP *)

 in(pub,m5);       (*UA receives the message5 from IdP *)
 let httppostrequest=m5 in
 let (samlresponsea,responserelaystatea)=httppostrequest in
 if responserelaystatea=url then out(pub,httppostrequest);
  (*UA sends the message5 to SP *)

 in(pub,m7);       (*UA receives the message6 from SP *)
 let reauthusersp=m7 in
 if authusersp=reauthusersp then
 let secretY= passwordsp in
 let validsp=(usernamesp,passwordsp) in
 event beginauthUSERSP(validsp);
 out(pub,validsp);
  (*UA sends the message7 to SP*)

 in(pub,m8);if m8=resourse then out(pub,finish).
 (*UA receives the message8 form SP*)
```

Figure 11: The user agent process in ProVerif in ProVerif

```
let processserviceprovider =          (* Service Provider (SP)*)
 in(pub,urlx);      (* SP receives a message1 from UA *)
 new uri; new id; new version; new issuestant; new issuer;
 new nameidpolicy;
 let relaystate=urlx in
 let signature={sign((id,version,issuestant,issuer,nameidpolicy),PR(keysp))} in
 let samlrequest=(id,version,issuestant,signature,issuer,nameidpolicy) in
 let httpredirectresponse=(uri,samlrequest,relaystate) in
 event beginauthSAMLREQ(signature);
 out(pub,httpredirectresponse);
 (* SP sends a message2 to UA *)

 in(pub,m5);       (* SP receives a message5 from UA *)
 let (recsamlresponse,recresponserelaystate)=m5 in
 let {(recresponseid,recrecid,recresponseverrsion,
       recresponseissueinstant,recresponsedestination,
       recrepissuer,recresponsesignature,recresponsestatus,
       recassertion,recresponseextentions)}=recsamlresponse in
 let (recaid,recaversion,recaissueinstant,recaissuer,
      recasignature,recaauthnstatement,recasubject)=recassertion in
 if {versign(recresponsesignature,PU(KeyIdP1))}={(recresponseid,recrecid,
       recresponseverrsion,
       recresponseissueinstant,
       recresponsedestination,
       recrepissuer,recresponsestatus,
       recassertion,
       recresponseextentions)} then
 (* verify the digital signature of response element in a message5 *)
 event endauthSAMLRESP(recresponsesignature);
 if versign(recasignature,PU(KeyIdP2))={(recaid,recaversion,
       recaissueinstant,recaissuer,
       recaauthnstatement,
       recasubject)} then
 (*verify the digital signature of assertion element in a message5 *)
 new authusersp; out(pub,authusersp);( *SP sends a message6 to UA * )

 in(pub,m7);
 let (reusernamesp,repasswordsp) = m7 in
 new usernamesp; new passwordsp; ( *SP receives a message7 from UA *)
 if usernamesp=reusernamesp then
 if passwordsp=repasswordsp then
 event endauthUSERSP(m7);
 out(pub,resourse).
 ( *SP sends a messae8 to UA * )
```

Figure 12: The service provider process in ProVerif

```
process
new Keysp;
new KeyIdP1;
new KeyIdP2;
out(pub, PU(Keysp));
out(pub, PU(KeyIdP1));
out(pub, PU(KeyIdP2));
!processuseragent | !processserviceprovider | !processidentityprovider
```

Figure 13: The identity provider process in ProVerif

```
let processidentityprovider =
new responseid; new responseverrsion; new responseissueinstant;
new responsedestination; new repissuer; new responsestatus;
new responseextentions; new aid; new aversion; new aissueinstant;
new aissuer; new aauthnstatement; new asubject;
in (pub,m2);      (*IdP receives the message2 from UA*)
let (recuri,recsamlrequest,recrelaystate)=m2 in
let {(recid,recversion,recissuestant,
      recsignature,recissuer,recnameidpolicy)}=recsamlrequest in
if versign(recsignature,PU(Keysp))={(recid,recversion,
                                     recissuestant,recissuer,
                                     recnameidpolicy)} then
(* verify the digital signature in samlrequest in message2 *)
event endauthSAMLREQ(recsignature);
new authuseridp,
out(pub,authuseridp);      (*IdP sends the message3 to UA*)

in(pub,m4);      (*IdP receives the message4 from UA*)
let (reusernameidp,repasswordidp)=m4 in
if usernameidp=reusernameidp then
if passwordidp=repasswordidp then
new SPandIDP,
let responserelaystate=recrelaystate in
let asignature={sign((aid,aversion,aissueinstant,
                      aissuer,aauthnstatement,
                      asubject),PR(KeyIdP2))} in
let assertion={(aid,aversion,aissueinstant,
                aissuer,asignature,aauthnstatement,
                asubject)} in
let responsesignature={sign((responseid,recid,responseverrsion,
                             responseissueinstant,
                             responsedestination,repissuer,
                             responsestatus, assertion,
                             responseextentions),PR(KeyIdP1))} in
let samlresponse={(responseid,recid,responseverrsion,
                   responseissueinstant,responsedestination,
                   repissuer,responsesignature,
                   responsestatus,assertion,responseextentions)} in
let httpresponse=(samlresponse,responserelaystate) in
event endauthUSERIDP(m4);
event beginauthSAMLRESP(responsesignature);
out(pub,httpresponse).      (*IdP sends the message5 to UA*)
```

Figure 14: The main process in ProVerif

Figure 15 shows the result of the secrecy of query attacker:secrectX and query attacker:secrectY. From the result we find that the secrectX and secrectY have not secrecy. The result is consistent with the fact. That is because the secrectX and secrectY are sent in the way of plaintext.

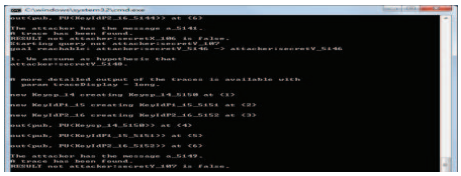

Figure 15: The results of secrecy

Hence the attacker can monitor the public channel to get the secrectX and secrectY. Hence the secrectX and secrectY have not secrecy. In order to implement the secrecy of the secrectX and secrectY some security mechanism must be used, for example, encryption.

Figure 16 shows the result that Identity Provider does not authenticate User Agent because the User Agent sends the password passwordidp in the way of plaintext to the Identity Provider. Hence the attacker can get the password passwordidp and launch an impersonation attack. We can use the encryption cipher or digital signature to address the problem.

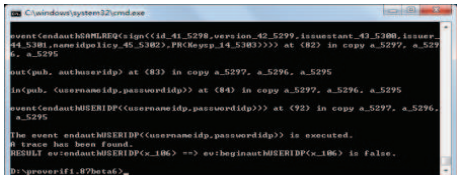

Figure 16: The result that identity provider does not authenticate user agent

Figure 17 shows the result that Service Provider does not authenticate User Agent because the User Agent sends the password passwordsp in the way of plaintext to the Service Provider. Hence the attacker can get the password passwordsp and launch an impersonation attack. We can use the encryption cipher or digital signature to address the problem.

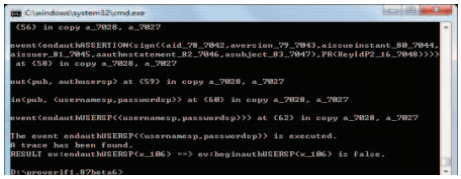

Figure 17: The result that service provider does not authenticate user agent

Figure 18 shows the result that Identity Provider can authenticate Service Provider because the Service Provider sends the its digital signature sign((id,version,issuestant,issuer,nameidpolicy),PR(keysp)) to the Identity Provider. Hence the Identity Provider can authenticate Service Provider.

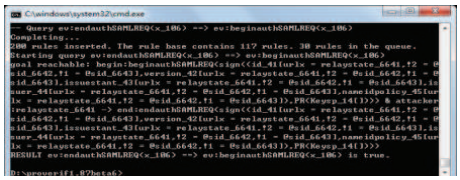

Figure 18: The result that identity provider authenticates service provider

Figure 19 shows the result that Service Provider can authenticate Identity Provider because the Identity Provider sends the digital signature sign((aid,aversion,aissueinstant,aissuer,aauthnstatement, asubject),PR(KeyIdP2)) to the Service Provider. Hence Service Provider can authenticate Identity Provider

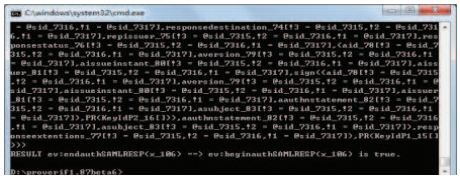

Figure 19: The result that service provider authenticates identity provider

8 Conclusion and Discussion

Owning to the complexity of Identity Federation Security Protocol, security analysis is important and has become a challenging issue. Therefore, in this paper, we firstly review the development of the formal methods on Identity Federation Security Protocol Based on SAML2.0, and then apply the ProVerif tool to analyze its security properties. The result shows that it has not secrecy for some keys and but it has some authentications. At the same time we present solutions to address the vulnerabilities.

Our method is basically similar to the method in the reference [1]. But there are two differences between the reference [1] and our work. The first difference is that in the reference [1], SAML SP-Initiated SSO profile is analyzed, but in our work, the identity federation security protocol based on SP-initiated SAML 2.0 includes eight messages exchanged among service provider, user agent and identity provider is analyzed. The second difference is that in reference [1] the model checker SATMC based on HLPSL++ as the specification language are used, but in our work, the an automatic cryptographic protocol verifier ProVerif and the Applied PI calculus are used.

In the near future, we will automatically analyze it in the computational model. At the same time, we will use the automatic method to generate the secure implementation in programming languages, for examples, JAVA language, C++ language.

Acknowledgments

This work was supported in part by the Fundamental Research Funds for the Central Universities, South Central University for Nationalities No. CZZ19003 and QSZ17007, and in part by the natural science foundation of Hubei Province under the grants No.2018ADC150."

References

[1] A. Alessandro, R. Carbone, L. Compagna, J. Cuéllar, G. Pellegrin-o, and A. Sorniotti, "An authentication flaw in browser-based single sign-on protocols: Impact and remediations," *Computers & Security*, vol. 33, pp. 41–58, 2013.

[2] D. S. AbdElminaam, "Improving the security of cloud computing by building new hybrid cryptography algorithms," *International Journal of Electronics and Information Engineering*, vol. 8, no. 1, pp. 40–48, 2018.

[3] R. Amin, N. Kumar, G.P. Biswas, R. Iqbal, , and V. Chang, "A light weight authentication protocol for iot-enabled devices in distributed cloud computing environment," *Future Generation Computer System*, vol. 78, no. 3, pp. 1005–1019, 2018.

[4] A. Bauer, J. Gross, P.L. Lumsdaine, M.Shulman, M. Sozeau, and B.Spitters, "The hott library: a formalization of homotopy type theory in coq," in *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs (CPP2017)*, pp. 164–172, Paris, France, Jan. 2017.

[5] M. Beek and P.Moiso C.Petrocchi, "Towards security analyses of an identity federation protocol for web services in convergent networks," in *Proceedings of the 3rd Advanced International Conference on Telecommunications*, pp. 1–8, Morne, Mauritius, May. 2007.

[6] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre. "Proverif 2.00: Automatic cryptographic protocol verifier, user manual and tutoria,". Tech. Rep. http://prosecco.gforge.inria.fr/personal/bblanche/proverif/manual.pdf, Apr. 2018.

[7] Q. Carbonneaux, J. Hoffmann, T. Reps, and Z. Shao, "Automated recourse analysis with coq proof objects," in *Proceedings of 29th International Conference on Computer-Aided Verification (CAV2017)*, pp. 64–85, Heidelberg, Germany, July. 2017.

[8] C. Guo, C. Chang, and S. Chang, "A secure and efficient mutual authentication and key agreement protocol with smart cards for wireless communications," *International Journal of Network Security*, vol. 20, no. 2, pp. 323–331, 2018.

[9] M. Y. Chen, C. W. Liu, and M. S. Hwang, "Securedropbox: A file encryption system suitable for cloud storage services," in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, pp. 1–334, Miami,FL,USA, Aug. 2013.

[10] K. Chetioui, G. Orhanou, and S. Hajji, "New protocol e-dnssec to enhance dnssec security," *International Journal of Network Security*, vol. 20, no. 1, pp. 19–24, 2018.

[11] M. Zamani E.Ghazizadeh, J. A. Manan and A, "Pashang. a survey on security issues of federated identity in the cloud computing," in *Proceedings of IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 1–4, Taipei, Taiwan, Dec. 2012.

[12] M. A. Elakrat and J. C. Jung, "Development of field programmable gate array–based encryption module to mitigate man-in-the-middle attack for nuclear power plant data communication network," *Nuclear Engineering and Technology*, vol. 50, no. 5, pp. 780–787, 2018.

[13] Y. F. Zhu K. Wang and M. Lin, "Provably secure browser-based mutual authentication protocol for federated identity management," *Application Research of Computers*, vol. 30, no. 6, pp. 1843–1846, 2013.

[14] X. H. Li, S. X. Li, J. Hao, Z. Y. Feng, and B. An, "Optimal personalized defense strategy against man-in-the-middle attack," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pp. 593–599, San Francisco,USA, Feb. 2007.

[15] J. Ling, Y. Wang, and W. Chen, "An improved privacy protection security protocol based on nfc," *International Journal of Network Security*, vol. 19, no. 1, pp. 39–46, 2017.

[16] V. Mittal, S. Gupta, and T. Choudhury, "Comparative analysis of authentication and access control protocols against malicious attacks in wireless sensor networks," in *Proceedings of the First International Conference on SCI*, pp. 555–262, San Francisco,USA, Jan. 2018.

[17] M. S. Ferdous. and R. Poet, "Dynamic identity federation using security assertion markup language (saml)," in *Proceedings of the 3rd IFIP WG 11.6 Working Conference*, pp. 131–146, London, UK, Apr. 2013.

[18] F. Nabi Muhammad and Mustafa Nabi, "A process of security assurance properties unification for application logic," *Seventh International Conference on Complex, Intelligent, and Software Intensive Systems*, vol. 48, no. 6, pp. 40–48, 2017.

[19] G. Oliva, S. Cioabă, and C. N. Hadjicostis, "Distributed calculation of edge-disjoint spanning trees for robustifying distributed algorithms against man-in-the-middle attacks," *IEEE Transaction on Control of Network System,*, no. DOI: 10.1109/TCNS.2017.2746344, pp. 1–1, 2017.

[20] A. Marín-López P. A .Cabarcos, F. A. Mendoza and D. Díaz-Sánchez, "Enabling saml for dynamic identity federation management," in *Proceedings of the Second IFIP WG 6.8 Joint Conference on Wireless and Mobile Networking(WMNC'09)*, pp. 173–184, Gdańsk, Poland, Sep. 2009.

[21] F. G. Mármol P. A. Cabarcos, F. Almenárez and A. Marín, "To federate or not to federate: A reputation-based mechanism to dynamize cooperation in identity management," *Wireless Personal Communications*, vol. 75, no. 3, pp. 1769–1786, 2014.

[22] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, "Secure and substainable load balancing of edge data centers in fog computing," *IEEE Communication Magazine*, vol. 56, no. 5, pp. 60–65, 2018.

[23] S. Saklikar and S. Saha, "Identity federation for voip systems," *Journal of Computer Security*, vol. 18, no. 4, pp. 499–540, 2010.

[24] K. Suthar and J. Patel, "Encryscation: An secure approach for data security using encryption and obfuscation techniques for iaas and daas services in cloud environment," in *Proceedings of International Conference on Communication and Networks, Advances in Intelligent System and Computing 508*, pp. 323–331, India, July. 2017.

[25] O. Wahballa1, Ab. Wahaballa, F. Li, I. Idris, and C. Xu, "Medical image encryption scheme based on arnold transformation and id-ak protocol," *International Journal of Network Security*, vol. 19, no. 5, pp. 776–784, 2017.

[26] C. Y. Yang, Y. Lin, and M. S. Hwang, "Downlink relay selection algorithm for amplifyand-forward cooperative communication systems," in *Seventh International Conference on Complex, Intelligent, and Software Intensive Systems*, pp. 331–334, Dalian, China, July 2013.

[27] L. Yao, J. Liu, D. Wang, J. Li, and B. Meng, "Formal analysis of sdn authentication protocol with mechanized protocol verifier in the symbolic model," *International Journal of Network Security*, vol. 20, no. 6, pp. 1125–1136, 2018.

Biography

Jintian Lu received his M.S degree at school of computer, South-Center University for Nationalities, China. Now he is pursuing the Ph.D. degree with School of Data and Computer Science, Sun Yat-sen University, Guangzhou, Guangdong, China. His current research interests include the security of security protocol and its implementations and cloud security.

Xudong He was born in 1991 and is now a postgraduate at school of Computer Science, South-Central University for Nationalities. His research interests include: security protocol implementations and reverse engineering.

Yitong Yang was born in 1991 and is now a postgraduate at the school of computer, South-Center University for Nationalities, China. Her current research interests include security protocols and formal methods.

Dejun Wang was born in 1974 and received his Ph.D. in information security at Wuhan University in China. Currently, he is an associate professor in the school of computer, South-Center University for Nationalities, China.

He has authored/coauthored over 20 papers in international/national journals and conferences. His current research interests include security protocols and formal methods.

Bo Meng was born in 1974 in China. He received his M.S. degree in computer science and technology in 2000 and his Ph.D. degree in traffic information engineering and control from Wuhan University of Technology at Wuhan, China in 2003. From 2004 to 2006, he worked at Wuhan University as a postdoctoral researcher in information security. Currently, he is a full Professor at the school of computer, South-Center University for Nationalities, China. He has authored/coauthored over 50 papers in International/National journals and conferences. In addition, he has also published a book "secure remote voting protocol" in the science press in China. His current research interests include Cyberspace security.