Multimed Tools Appl (2020) 79:15793–15811
https://doi.org/10.1007/s11042-017-5548-2

Smart card-based secure authentication protocol in multi-server IoT environment

Won-il Bae [1,2] · Jin Kwak [2,3]

Received: 12 May 2017 / Revised: 8 November 2017 / Accepted: 14 December 2017 /
Published online: 26 December 2017

Abstract In recent years, the internet of things has been widely utilized in various fields, such as in smart factories or connected cars. As its domain of application has expanded, it has begun to be employed using multi-server architectures for a more efficient use of resources. However, because users wishing to receive IoT(Internet of Things) services connect to multi-servers over wireless networks, this can expose systems to various attacks and result in serious security risks. To protect systems (and users) from potential security vulnerabilities, a secure authentication technology is necessary. In this paper, we propose a smart card-based authentication protocol, which performs the authentication for each entity by allowing users to go through the authentication process using a smart card transmitted from an authentication server, and to login to a server connected to the IoT. Furthermore, the security of our proposed authentication protocol is verified by simulating a formal verification scenario using AVISPA(Automated Validation of Internet Security Protocols and Applications), a security protocol-verification tool.

Keywords User authentication · Multi server · Internet of things · Formal verification · Security

1 Introduction

By enabling devices such as machines to exchange information with embedded software, sensors, and so on via internet networks and consequently enhancing the functionality and

✉ Jin Kwak
security@ajou.ac.kr

Won-il Bae
wibae.isaa@gmail.com

[1] Department of Computer Engineering, Ajou University, Suwon, South Korea

[2] Industry-University Cooperation, Ajou University, 260 Worldcup-ro, Yeongtong-gu, Suwon-si, Gyunggi-do 16499, South Korea

[3] Department of Cyber Security, Ajou University, Suwon, South Korea

performance of individual devices, the IoT enables the provision of new intelligent services. In recent years, as the applications of the IoT have expanded, it has undergone considerable progress through interactions with a diverse range of other industries, and has been employed in various applications such as smart factories, intelligent buildings, and connected cars [11].

Although the IoT is utilized in various fields, the performance of its sensors is not particularly high. Therefore, it is employed with multi-servers, in order to increase the resource efficiency. In particular, in terms of reducing the number of operations, the authentication protocol of a smart card-based multi-server environment involves the operations of one-way functions and the exclusive or (XOR) function. Thus, this authentication protocol has been the subject of ongoing research [1, 3–6].

However, if the multi-server authentication system is vulnerable, then attacks such as user impersonation, session key leakages, and replay attacks may occur in the process of connecting to the multi-server where the user stores IoT information over wireless networks [8]. This may result in leakages of confidential information by an unauthorized attacker, data sniffing and forgery/tampering, and other attacks during the communication process, and may lead to issues with the service availability of the multi-server [9, 13, 14]. Therefore, to mitigate the potential security vulnerabilities in a multi-server IoT environment, a secure authentication protocol is required.

In this study, we analyze the threats that may occur in multi-server IoT environment networks during the communication process, and propose a secure authentication protocol that can respond to such security threats. In addition, we verify the security of the proposed protocol by performing a formal verification of the proposed authentication protocol using the AVISPA.[1]

In the authentication protocol proposed in this paper, a user logs in to the IoT server using a smart card, and the authentication server can verify the user and the IoT server. By generating the same session key, the authentication process between the user, the IoT server, and the authentication server is performed. The composition of this paper is as follows.

In Section 2, we describe the security threats that may occur in a multi-server IoT environment and explain AVISPA, a formal verification tool for security protocols. In Section 3, we propose a secure authentication protocol for multi-server IoT environments, and describe the authentication protocol specified by the HLPSL(high-level protocol specification language). In Section 4, we describe the security analysis of the proposed protocol, and in Section 5, we simulate the formal verification of the authentication protocol via AVISPA. Finally, our conclusions are listed in Section 6.

2 Related work

2.1 Multi-server IoT security threats

The vast amount of data collected by sensors connected to a multi-server can be considered an attractive target for attackers. In a multi-server IoT environment, a user may access the multi-server over a wireless network, which can be a serious security threat. The security threats that

[1] AVISPA, DIST, Eidgenoessische Technische Hochschule Zuerich (ETHZ), CASSIS, Siemens Aktiengesellschaft, http://www.avispa-project.org/

can occur in the network communication process of a multi-server IoT environment are as follows [7, 12].

- User impersonation attack

  In this kind of attack, assuming that the malicious attacker has knowledge of the user's login request message from the previous session with the IoT server, the attacker masquerades as the authorized user by deriving the login request message of the current session.

- Session key disclosure attack

  A public channel can occur when a user connects to the server over a wireless network. In this public channel, a malicious attacker can leak the session key by extracting the secret values. Through this attack, leakage and forgery/tampering of data stored on the server can occur.

- Denial-of-service attack

  If one or more attackers generate a large number of identical login request messages using their smart cards and send them to the server connected to the sensor, then there may be a problem in service availability in the server.

- Replay attack

  In this type of attack, an attacker can authenticate the user from the server connected to the sensor by storing the message that was communicated to the authentication server in the previous session for the authenticated user, and retransmitting the message to the current session or a subsequent session.

- Server spoofing attack

  A malicious attacker can impersonate the IoT server when the user logs in. Therefore, the attacker can masquerade as the server to obtain the user login information.

- Invasion of privacy

  Invasion of privacy is a violation of privacy through the revelation of private material, exposing various information and communication subjects of the user on the communication networks between the user and the server.

2.2 AVISPA

AVISPA is a tool that formally verifies the security of the internet protocol, and notifies the user with messages when it discovers attacks against the protocol [10]. AVISPA is composed of independently developed modules and the HLPSL, which is used as the input for the protocol specification. HLPSL is a module-type role-based language that can express various operators, as well as data flow and structure, intruder models, and complex security properties [15].

The roles can be divided into two types according to their purposes. One type consists of descriptions of the values required to describe each entity, and performs message transmission/reception using SND and RCV commands. The other role is to contain the overall scenario, and include the contents of the declared constants, the information known to an attacker, and the verification properties for the authentication protocol.

Furthermore, AVISPA is automatically generated in intermediate format (IF) via the HLPSL2IF translator, and used as input to the OFMC(On-the-fly Model-Checker), CL-AtSe(CL-based Attack Searcher), SATMC(SAT-based Model-Checker), and TA4SP(Tree Automata-based Protocol Analyser) models. The schematic of its architecture is shown in Fig. 1 [2].

3 Proposed authentication protocol

The authentication protocol proposed in this paper consists of the user $\boldsymbol{U_i}$, IoT server $\boldsymbol{S_j}$, and authentication server ***CS***. When the user logs into the IoT server, the protocol performs authentication for each entity. Table 1 lists the parameter values used in the proposed authentication protocol.

There are three phases in the proposed authentication protocol, and these are carried out in the following order: registration, login and authentication, and password change. *CS* is an authentication server that can be trusted, and it is responsible for the registration and authentication of the user and the IoT server. In addition, because the proposed authentication protocol employs timestamps, the authentication server *CS*, user U_i and IoT server S_j performs time synchronization.

3.1 Registration phase

During the registration phase, the user U_i and the IoT server S_j request registration to the authentication server CS. In return, the authentication server issues a smart card to the user U_i,

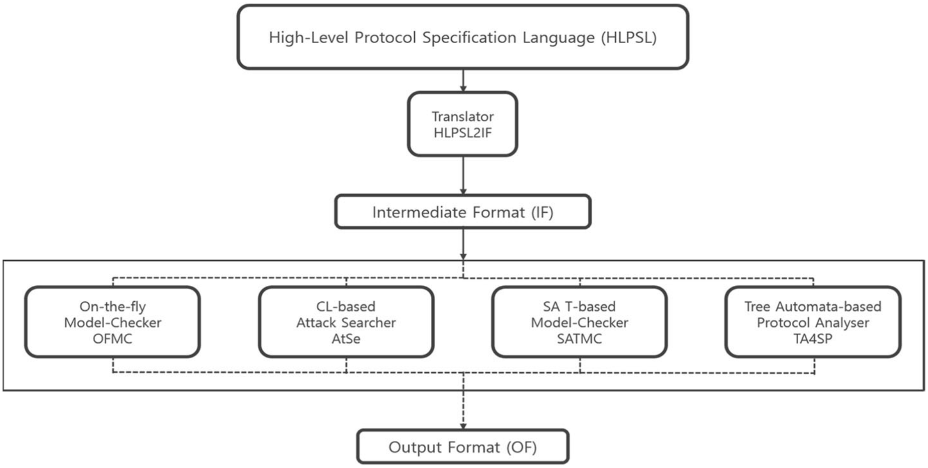

High-Level Protocol Specification Language (HLPSL)
Translator HLPSL2IF
Intermediate Format (IF)
On-the-fly Model-Checker OFMC
CL-based Attack Searcher AtSe
SA T-based Model-Checker SATMC
Tree Automata-based Protocol Analyser TA4SP
Output Format (OF)

Fig. 1 Architecture of the AVISPA tool

Table 1 Notation used in the proposed protocol

Notation	Description
U_i	i th user
S_j	j th user
CS	Control server
$ID_{\boldsymbol{i}}$	Identity of U_i
$P_{\boldsymbol{i}}$	Password of U_i
$UID_{\boldsymbol{i}}$	Anonymity value of U_i
SID_j	Identity of SID_j
x	Master secret key chosen by CS
Ts	Timestamp
N_{i1}	Random number generated by U_i's smart card for session key agreement
N_{i2}	Random number generated by $\mathrm{S_j}$ for session key agreement
N_{i3}	Random number generated by CS for session key agreement
SK	Common session key shared among U_i, $\mathrm{S_j}$, and CS
$h(*)$	Collision-free one-way hash function
$\oplus$	Exclusive OR operation
$\|$	Message concatenation operation

and sends the necessary values for the login and authentication phases to the IoT server S_j . This process is illustrated in Fig. 2.

STEP 1. S_j sends its identification value SID_j to CS via a secured channel, and CS computes the $Serinfor_j$ value that contains the information on the IoT server send to S_j via secure channel.

$$Serinfor_j = h\left(SID_j \| x\right) \tag{1}$$

STEP 2. U_i chooses the user ID_i and password P_i, computes $EncPass_i$, and sends the registration request message (ID_I, $EncPass_i$,UID_i) with the user's anonymity values to CS.

$$EncPass_i = h(ID_i \| h(P_i)) \tag{2}$$

STEP 3. CSgenerates the user's secret information value, $Userinfor_i$, and stores UID_i, $Userinfor_i$, $EncPass_i$, $h(*)$, $\;$ and $h(x)$ in the smart card.

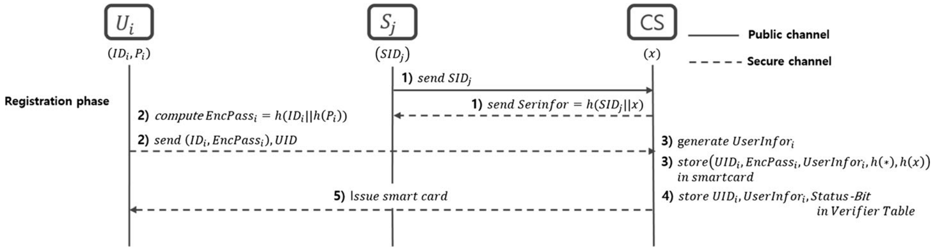

U_i
(ID_i, P_i)
S_j
(SID_j)
CS
(x)
Public channel
Secure channel
Registration phase
1) send SID_j
1) send $Serinfor = h(SID_j||x)$
2) compute $EncPass_i = h(ID_i||h(P_i))$
2) send $(ID_i, EncPass_i), UID$
3) generate $UserInfor_i$
3) store$(UID_i, EncPass_i, UserInfor_i, h(*), h(x))$ in smartcard
4) store $UID_i, UserInfor_i, Status\text{-}Bit$ in Verifier Table
5) Issue smart card

Fig. 2 Registration phase

$$Userinfor_i = h(EncPass_i \| x) \tag{3}$$

STEP 4. *CS* stores the user secret information value *Userinfor*$_i$, the user's anonymity value *UID*$_i$, and the status-bit value in the verifier table. If the user performs the registration process, then the status-bit value is stored as 1, and if there is no registration then the value is stored as 0. The verifier table is presented in Table 2.

STEP 5. *CS* issues the smart card to the user U_i.

3.2 Login and authentication phase

During the login and authentication phase, the verification of the legitimate smart card holder is performed. In order to login, the user U_i sends a login request message to the IoT server S$_j$, and *CS* performs the verification of each entity. Then, U_i, S_j, and *CS* all generate the same session key (Fig. 3).

STEP 1. U_i inserts the smart card into the card reader, and enters their ID, ID_i, and password, P_i. The smart card computes $EncPass_i^{'}$, and compares the information with $EncPass_i$ contained in the smart card. If the information matches, then the user is verified as the legitimate owner of the smart card. If the information does not match, then the session is terminated.

$$EncPass_i^{'} = h(ID_i \| h(P_i)) \tag{4}$$

$$EncPass_i = ?EncPass_i^{'} \tag{5}$$

STEP 2. The user U_i who is verified as the legitimate owner of the smart card, selects a random value N_{j1} to be generated for each session, and computes A_i and $Veru_i$ with $h(x)$ and $UserInfor_i$ contained in the smart card and the chosen random value N_{j1}. Then, the timestamp *Ts* is generated.

$$A_i = Userinfor_i \oplus h(x) \oplus N_{i1} \tag{6}$$

$$Veru_i = h\left(h(x) \| N_{i1}\right) \tag{7}$$

Table 2 The verifier table

Anonymity value	User-verifier	Status-bit
$\vdots$	$\vdots$	$\vdots$
UID_i	$Userinfor_i$	0/1
UID_i	$Userinfor_j$	0/1
$\vdots$	$\vdots$	$\vdots$

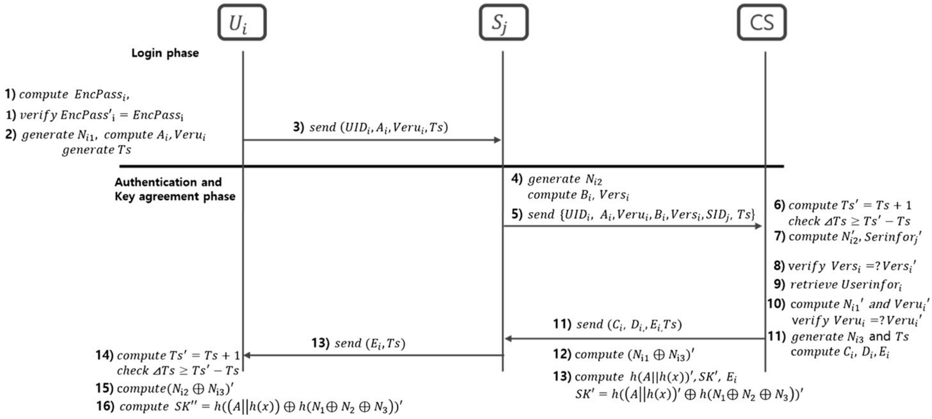

U_i
S_j
CS
Login phase
1) compute $EncPass_i$,
1) verify $EncPass'_i = EncPass_i$
2) generate N_{i1}, compute $A_i, Veru_i$ generate Ts
3) send $(UID_i, A_i, Veru_i, Ts)$
Authentication and Key agreement phase
4) generate N_{i2} compute B_i, $Vers_i$
5) send $\{UID_i, A_i, Veru_i, B_i, Vers_i, SID_j, Ts\}$
6) compute $Ts' = Ts + 1$ check $\Delta Ts \geq Ts' - Ts$
7) compute N'_{i2}, $Serinfor_j'$
8) verify $Vers_i =? Vers_i'$
9) retrieve $Userinfor_i$
10) compute N_{i1}' and $Veru_i'$ verify $Veru_i =? Veru_i'$
11) generate N_{i3} and Ts compute C_i, D_i, E_i
11) send $(C_i, D_i, E_i Ts)$
12) compute $(N_{i1} \oplus N_{i3})'$
13) compute $h(A\|h(x))'$, SK', E_i $SK' = h((A\|h(x))' \oplus h(N_1 \oplus N_2 \oplus N_3))'$
13) send (E_i, Ts)
14) compute $Ts' = Ts + 1$ check $\Delta Ts \geq Ts' - Ts$
15) compute$(N_{i2} \oplus N_{i3})'$
16) compute $SK'' = h((A\|h(x)) \oplus h(N_1 \oplus N_2 \oplus N_3))'$

Fig. 3 Login and authentication phase

STEP 3. The user U_i configures the login request message (A_i, $Veru_i$, UID_i,Ts) with his/her anonymity value UID_i, computes A_i and Ts, and sends the message to the IoT server S_j.

STEP 4. The IoT server S_j that received the login request message from U_i selects a random value N_{i2} to be generated for each session, and computes B_i and $Vers_i$ using the $Serinfor_j$ value received in the registration phase.

$$B_i = Serinfor_j \oplus N_{i2} \tag{8}$$

$$Vers_i = h\left(h\left(SID_j \| x\right) \| N_{i2}\right) \tag{9}$$

STEP 5. S_i sends the login request message (UID_i, A_i, $Veru_i$, B_i, $Vers_i$, SID_j, Ts) to CS. The message is configured for the A_i, UID_i (received from the user U_i), S_i's own identification value SID_j, B_i (which was generated earlier), and the timestamp Ts.

STEP 6. The CS that received the login request message from S_i computes $Ts' = Ts + 1$, and then confirms whether $\Delta Ts \geq Ts' - Ts$. Here, Ts' is the stamp for the time the server received the login message, and ΔTs is the minimum authentication time considering the time for the login message transmission.

STEP 7. CS generates $Serinfor_j'$ using the received SID_j value and its own master key, and extracts the N_{i2} value via the B_i value received from the login request message.

$$Serinfor_j' = h\left(SID_j \| x\right) \tag{10}$$

$$N_{i2}' = Serinfor_j' \oplus B_i \tag{11}$$

STEP 8. By using the computed N_{i2}' value, CS generates the $Vers_i'$ value, and if this matches the B_i value received by the login request message, it is authenticated as the legitimate IoT server S_j; if not, the session is terminated.

$$Vers_i{}' = h\left(h\left(SID_j \| x\right) \| N_{i2}{}'\right) \tag{12}$$

$$Vers_i = ?Vers_i{}' \tag{13}$$

STEP 9. By using the *UID$_i$* of the login request message, *CS* can search for *Userinfor$_i$* from the verifier table generated in the registration phase.

STEP 10. *CS* selects the random value N_{i3} and computes the $N_{i1}^{'}$value using the received A_i value, the generated $h(x)$ and the previously retrieved *Userinfor$_i$*. Using this computed $N_{i1}^{'}$ value and $h(x)$, it generates the $Veru_i^{'}$ value. Next, if this matches the *Veru$_i$* value received with the login request message, then it is authenticated as a legitimate user, and generates the following session key *SK*. If it does not match, then the session is terminated.

$$N_{i1}^{'} = Userinfor_i \oplus h(x) \oplus A_i \tag{14}$$

$$Veru_i{}' = h\left(h(x) \| N_{i1}{}'\right) \tag{15}$$

$$Veru_i = ?Veru_i{}' \tag{16}$$

$$SK_i = h\left(h\left(A \| h(x)\right) \oplus h(N_{i1} \oplus N_{i2} \oplus N_{i3})\right) \tag{17}$$

STEP 11. The *CS* generates the timestamp *Ts*. Next, it computes C_i, D_i, *and* E_i, and sends the mutual authentication message (C_i, D_i,E_i, *Ts*) to S_i.

$$C_i = N_{i1} \oplus N_{i3} \oplus h\left(SID_j \oplus N_{i2}\right) \tag{18}$$

$$D_i = h\left(A \| h(x)\right) \oplus h\left(SID_j \oplus N_{i2}\right) \tag{19}$$

$$E_i = N_{i2} \oplus N_{i3} \oplus h\left(A \| h(x)\right) \tag{20}$$

STEP 12. S_i receives the mutual authentication message, and computes $(N_{i1} \oplus N_{i3})^{'}$ via its own SID_j value and the random value N_{i2}.

$$(N_{i1} \oplus N_{i3})^{'} = C_i \oplus h\left(SID_j \oplus N_{i2}\right) \tag{21}$$

STEP 13. S_i computes $h(A\|h(x))^{'}$ via its own SID_j value and the random value N_{i2}, using the D_i value received in the mutual authentication message. It generates the session key *SK* by on operating its own random value N_{i2} with the previously computed $(N_{i1} \oplus N_{i3})^{'}$. Next, S_i computes E_i and sends a login response message (E_i, *Ts*) to the user U_i.

$$h\left(A\|h(x)\right)' = D_i \oplus h\left(SID_j \oplus N_{i2}\right) \tag{22}$$

$$SK' = h\left(h\left(A\|h(x)\right)' \oplus h(N_{i1} \oplus N_{i2} \oplus N_{i3})\right)' \tag{23}$$

$$E_i = (N_{i2} \oplus N_{i3}) \oplus h\left(A\|h(x)\right) \tag{24}$$

STEP 14. After receiving the login request message from the S_i, the user U_i computes $Ts' = Ts + 1$ to confirm that $\varDelta Ts \geq Ts' - Ts$. Ts' is the timestamp for when the login message is received by the server, and $\varDelta Ts$ is the minimum authentication time considering the transmission time for the login message.

STEP 15. By using the E_i value received in the mutual authentication message, U_i can compute the $(N_{i2} \oplus N_{i3})'$ value via the A_i value generated by the user and the $h(x)$ value contained in the smart card.

$$(N_{i2} \oplus N_{i3})' = E_i \oplus h\left(A\|h(x)\right) \tag{25}$$

STEP 16. U_i can operate on its own random value N_{i1} with $(N_2 \oplus N_3)'$ that was computed earlier, and can generate the session key SK via its own A_i value and the $h(x)$ value contained in the smart card. Therefore, the user U_i, IoT server S_j, and authentication server CS can perform the authentication by generating the same session key.

$$SK'' = h\left(A\|h(x)\right) \oplus (N_{i1} \oplus N_{i2} \oplus N_{i3})' \tag{26}$$

3.3 Password change phase

The password change phase is the process performed when the user U_i wants to change their password P_i to a new one P_i^{NEW}. The process is illustrated in Fig. 4.

STEP 1. The user U_i inserts a smart card into the card reader and inputs their ID, ID_i, and password, P_i. The smart card computes $EncPass_i$ and generates $Userinfor_i'$ using the computed $EncPass_i$.

$$EncPass_i = h(ID_i \| h(P_i)) \tag{27}$$

$$Userinfor_i' = h(EncPass_i \| x) \tag{28}$$

STEP 2. The smart card compares the generated $Userinfor_i'$ with $Userinfor_i$ contained in the smart card. If these match, then the user is verified as the legitimate owner of the smart card, and the user can change the password. The smart card requests a new password.

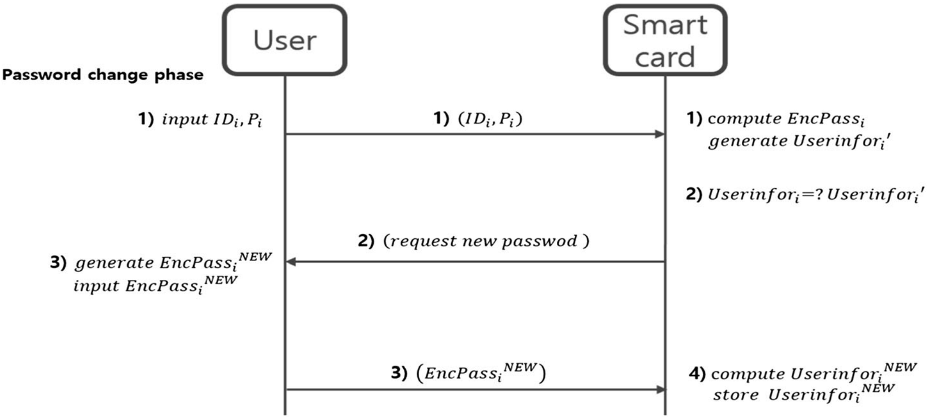


Fig. 4 Password change phase

$$Userinfor_i = ?Userinfor_i^{'} \tag{29}$$

STEP 3. Once verified as the legitimate smart card owner, the user enters the new password P_i^{NEW}, generates $EncPass_i^{NEW}$, and enters this into the smart card.

$$EncPass_i^{NEW} = h\left(ID_i \| h\left(P_i^{NEW}\right)\right) \tag{30}$$

STEP 4. By using the $EncPass_i^{NEW}$ generated in the above process, the new $Userinfor_i^{NEW}$can be computed, and the existing $Userinfor_i$ is replaced with $Userinfor_i^{NEW}$, which is stored in the smart card. In this manner, the password change process is completed.

$$Userinfor_i^{NEW} = h\left(EncPass_i^{NEW} \| x\right) \tag{31}$$

3.4 Authentication protocol specification utilizing HLPSL

In this study, we use the AVISPA web tool for the formal verification of the authentication protocol. The AVISPA web tool can specify the authentication protocol with the CAS and HLPSL languages. This section explains the registration, and login and authentication phases of the proposed authentication protocol written in the HLPSL language.

Because AVISPA is a role-based language, it assigns a role to each participant. It is composed of role_U, which represents the user, role_S, which is assigned to the IoT server, and role_CS, representing the authentication server. In the role environment (), the constants used by the specified protocol are defined, and can specify the property values known to the attacker. In addition, secrecy_of and weak_authentication_on are specified depending on the goal, in order to add the secrecy () and witness () functions. These functions are used to verify the security and authentication. The security and authentication are the properties of the protocol's verification.

In the part where the formula is specified, concatenation can be expressed as “.”, xor operates as “xor(A, B)” and the exponent E^N operates as “exp(E, N).” In addition, in the part where each role is specified, the message specified as RCV can be transmitted, and the received contents can be expressed as SND.

This section deals with the specifications for the sent and received messages, and for the security properties. Table 3 details the specification of the user, and Table 4 concerns the specification of the IoT server. Table 5 presents the specification of the authentication server, and Table 6 illustrates the role environment, which contains the specified constants. Finally, Table 7 shows the specification of the goal, by specifying a function to verify the authentication protocol.

4 Security analysis

4.1 User impersonation (i.e., masquerading) attack

An attacker impersonates the authorized user by extracting the login request message of the current session, assuming that he/she knows the login request message from the previous session of the IoT server.

Assume that the attacker has learned the login message (UID_i, A_i, Ts) of the previous session via the public channel. Then, because the malicious attacker cannot compute $EncPass_i$, the random value N_{i1}, and $h(x)$, they cannot compute A_i, which is composed of the $Userinfor_i \oplus h(x) \oplus N_{i1}$ operation. Therefore, they cannot extract the login request message for the current session by impersonating the authorized user, and the proposed authentication protocol is secure against user impersonation attacks.

4.2 Session key disclosure attack

Assuming that an attacker can intercept and steal A_i, B_i, C_i, D_i, and E_i through the previous session via public channels, they still cannot extract the session key. The attacker cannot

Table 3 Specification for the user

```
    transition
        3. State=0 /\ RCV(start) =|> State':=1 /\ Key_1':=new() /\
Key_set_U_CS':=cons(Key_1',Key_set_U_CS)
                /\ SND({ID,H(ID.H(P)),UId}_Key_1')
        4. State=1 /\ in(Key_2',Key_set_CS_U)
                /\ RCV({UId,H(H(X).NU),H(ID.H(P)),H(H(ID.H(P)).X')}_Key_2') =|>
State':=2 /\ Key_set_CS_U':=delete(Key_2',Key_set_CS_U) /\ NU':=new()
                /\ SND(UId,xor(xor(H(H(ID.H(P)).X'),H(X')),NU'))
        10. State=2 /\
    RCV(xor(xor(NS',NCS'),H(xor(xor(H(H(ID.H(P)).X),H(X)),NU).H(X)))) =|> State':=3
                    /\
    witness(CS,S,auth_14,xor(xor(NS',NCS'),H(xor(xor(H(H(ID.H(P)).X),H(X)),NU).H(
    X))))
    /\ secret(NU,auth_10,{S,U})
end role
```

Table 4 Specification for the IoT server

```
    transition
        1. State=0 /\ RCV(start) =|> State':=1
                /\ SND(SId)
2. State=1 /\ in(Key_1',Key_set_CS_S) /\ RCV({H(SId.X')}_Key_1') =|> State':=2 /\
Key_set_CS_S':=delete(Key_1',Key_set_CS_S)
    5. State=2 /\ RCV(UId', H(H(X).NU’),xor(xor(H(H(ID'.H(P')).X),H(X)),NU')) =|>
 State':=3
                /\ secret(P',auth_1,{U,S})
                /\ secret(H(X),auth_2,{U,S})
                /\ secret(NU',auth_3,{U,S}) /\ NS':=new()
    /\
SND(UId',H(H(X).NU’),H(H(SID.X).NS),xor(xor(H(H(ID'.H(P')).X),H(X)),NU'),xor(H(SId.
X),NS'),SId)
        7. State=3 /\
    RCV(xor(xor(NS,NCS'),H(xor(xor(H(H(ID.H(P)).X),H(X)),NU).H(X)))) =|> State':=4
                /\ secret(NCS',auth_6,{CS,S})
                /\ secret(NS,auth_7,{CS,S})
                /\ secret(NU,auth_8,{CS,S})
                /\ secret(H(X),auth_9,{CS,S})
                  /\ witness(S,CS,auth_13, xor(xor(NU',NCS'),H(xor(SId,NS'))))
        8. State=4 /\ RCV(xor(xor(xor(H(H(ID.H(P)).X),H(X)),NU).H(X),H(xor(SId,NS))))
    =|> State':=5
        9. State=5 /\ RCV(xor(xor(NU',NCS'),H(xor(SId,NS')))) =|> State':=6
                /\ SND(xor(xor(NS,NCS),H(xor(xor(H(H(ID.H(P)).X),H(X)),NU).H(X))))
 end role
```

compute $Userinfor_i$ and N_{i1} through the known value of A_i because they cannot determine $h(X)$. Furthermore, the attacker cannot compute the value N_{i2} via the publicly accessible B_i value, because they cannot determine $Serinfor_j$. For the same reason, the attacker cannot extract the random values N_{i1}, N_{i2}, and N_{i3} selected by the user U_i, IoT server S_j, and the authentication server CS, respectively, from the published C_i, D_i, and E_i values and $h(x)$ generated by the authentication server CS. Therefore, the proposed authentication protocol is secure against such attacks involving session key leakages.

4.3 Denial-of-service (Dos) attack

When an attacker uses their own smart card to send a large number of identical login request messages (A_{K1}, UID_{K1},Ts), (A_{K2}, UID_{K2},Ts),…, (A_{Kn}, UID_{Kn},Ts), the IoT server S_j may experience problems with its availability. However, in the registration phase the value of the status bit is stored as 1, using the verifier table given in Table 2. In this manner, the proposed protocol is designed not to receive such login request messages. Thus, the proposed authentication protocol is secure against the denial-of-service attacks.

4.4 Replay attack

The proposed authentication protocol uses the timestamp for the authentication of messages in communications between the user U_i, the IoT server S_j, and the authentication server CS.

Table 5 Specification for the authentication server

```
    transition
        1. State=0 /\ RCV(SId) =|> State':=1 /\ Key_1':=new() /\
Key_set_CS_S':=cons(Key_1',Key_set_CS_S)
                /\ SND({H(SId.X)}_Key_1')
        3. State=1 /\ in(Key_2',Key_set_U_CS)
                /\ RCV({ID,H(ID.H(P')), UId'}_Key_2') =|> State':=2 /\
Key_set_U_CS':=delete(Key_2',Key_set_U_CS) /\ Key_3':=new() /\
Key_set_CS_U':=cons(Key_3',Key_set_CS_U)
                /\ SND({UId′, H(ID.H(P)),H(H(ID.H(P′)).X)}_Key_3')
        6. State=2 /\
    RCV(UId,H(H(X).NU′),H(H(SID.X).NS),xor(xor(H(H(ID.H(P)).X),H(X)),NU'),xor(H(SI
    d.X),NS'),SId) =|> State':=3
                /\ secret(NS',auth_4,{S,CS})
                /\ secret(H(SId.H(X)),auth_5,{S,CS})
                /\ witness(CS,S,auth_11,xor(H(SId.X),NS'))
                /\ witness(CS,U,auth_12,xor(xor(H(H(ID.H(P)).X),H(X)),NU')) /\
NCS':=new()
                /\ SND(xor(xor(NS',NCS'),H(xor(xor(H(H(ID.H(P)).X),H(X)),NU').H(X))))
                /\ SND(xor(xor(xor(H(H(ID.H(P)).X),H(X)),NU′). H(X),H(xor(SId,NS′))))
                /\ SND(xor(xor(NU',NCS'),H(xor(SId,NS'))))
end role
```

Therefore, if an attacker participates in a session to perform eavesdropping (i.e., sniffing) or forgery/tampering attacks on transmitted or received messages, the value of the timestamp *Ts* changes, and the protocol terminates the session.

Table 6 Specification for the role environment

```
    role environment()
def=
    const
        hash_0:function,
        sid:text,
        contr:agent,
        man:agent,
        password:text,
        uid:text,
        serv:agent,
        secretvalue:text,
        id:text,
        auth_1, auth_2, auth_3, auth_4, auth_5, auth_6, auth_7, auth_8, auth_9, auth_10,
    auth_11, auth_12, auth_13, auth_14:protocol_id
    intruder_knowledge = {man,serv,contr,sid,uid,id}
    composition
        session1(password,uid,man,serv,contr,secretvalue,sid,id,{},{},{})
end role
```

Table 7 Specification for the goal

```
goal
    secrecy_of auth_1
    secrecy_of auth_2
    secrecy_of auth_3
    secrecy_of auth_4
    secrecy_of auth_5
    secrecy_of auth_6
    secrecy_of auth_7
    secrecy_of auth_8
    secrecy_of auth_9
    secrecy_of auth_10
    weak_authentication_on auth_11
    weak_authentication_on auth_12
    weak_authentication_on auth_13
    weak_authentication_on auth_14

end goal
```

4.5 Server spoofing attack

In a server spoofing attack, an attacker impersonates an IoT server to obtain the desired information. However, in the login phase of the proposed authentication protocol, the authenticated user is identified via the verification of the smart card as $EncPass_i = ?EncPass_i^{'}$. Because the session is terminated if this does not match, the attacker cannot masquerade as the IoT server. Thus, the proposed authentication protocol is secure against server spoofing attacks, because it verifies the IoT server S_i by checking $Vers_i = ?Vers_i^{'}$ with the CS.

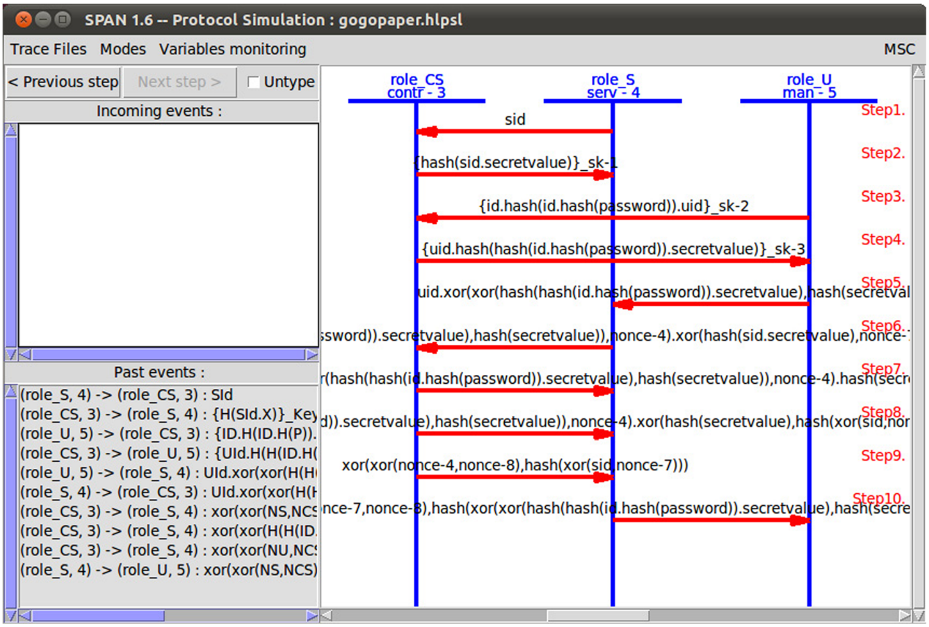
4.6 Invasion of privacy

Invasion of privacy is the infringement of privacy by exposing information regarding the subject during a communication procedure on a network. However, the proposed authentication protocol uses the identifiers of the user and the IoT server, UID_i and SID_j, and the anonymity value, to perform the communication procedure. Therefore, when the transmitted/received messages on a network are thought to have been eavesdropped, the communication subject cannot be identified, thus ensuring the privacy of the user and the IoT server.

5 Experimental result through formal verification

In this section, we analyze the experimental results by employing AVISPA, a formal verification tool for the authentication protocol, where the registration phase and login and authentication phase are specified using the HLPSL language, as described in Section 3.4.

The result of executing the specified HLPSL code file using the AVISPA web tool is shown in Fig. 5. From left to right, the figure shows CS, S_j, and U_i. The authentication messages are



```
(role_S, 4) -> (role_CS, 3) : SId
(role_CS, 3) -> (role_S, 4) : {H(SId.X)}_Key
(role_U, 5) -> (role_CS, 3) : {ID.H(ID.H(P)).
(role_CS, 3) -> (role_U, 5) : {UId.H(H(ID.H(
(role_U, 5) -> (role_S, 4) : UId.xor(xor(H(H
(role_S, 4) -> (role_CS, 3) : UId.xor(xor(H(H
(role_CS, 3) -> (role_S, 4) : xor(xor(NS,NCS
(role_CS, 3) -> (role_S, 4) : xor(xor(H(H(ID.
(role_CS, 3) -> (role_S, 4) : xor(xor(NU,NCS
(role_S, 4) -> (role_U, 5) : xor(xor(NS,NCS)
```

```
Step1.  sid
Step2.  {hash(sid.secretvalue)}_sk-1
Step3.  {id.hash(id.hash(password)).uid}_sk-2
Step4.  {uid.hash(hash(id.hash(password)).secretvalue)}_sk-3
Step5.  uid.xor(xor(hash(hash(id.hash(password)).secretvalue),hash(secretval
Step6.  ssword)).secretvalue),hash(secretvalue)),nonce-4).xor(hash(sid.secretvalue),nonce-
Step7.  r(hash(hash(id.hash(password)).secretvalue),hash(secretvalue)),nonce-4).hash(secr
Step8.  d)).secretvalue),hash(secretvalue)),nonce-4).xor(hash(secretvalue),hash(xor(sid,non
Step9.  xor(xor(nonce-4,nonce-8),hash(xor(sid,nonce-7)))
Step10. nce-7,nonce-8),hash(xor(xor(hash(hash(id.hash(password)).secretvalue),hash(secre
```

Fig. 5 Execution screen of the AVISPA

transmitted and received through eight main phases, including the registration phase. However, in practice, authentication messages are transmitted and received in 10 phases, because the C_i, D_i, and E_i values sent to S_j from *CS* are divided and transmitted sequentially.

The secret() and witness() functions and the verification properties for the authentication protocol were employed in the goal of the HLPSL created in Section 3.4 for verification. For the secret () and witness () functions shown in Table 7, the specified security properties are as follows:

1. secret(P′,auth_1,{U,S})

    This pertains to the login and authentication phase of STEP 3, and verifies whether the secrecy of P_i in the login request message (A_i, UID_i,Ts) between the user U_i and IoT server S_i is satisfied.

2. secret(H(X),auth_2,{U,S})

    This pertains to the login and authentication phase of STEP 3, and verifies whether the secrecy of $h(X)$ in the login request message (A_i, UID_i,Ts) between the user U_i and IoT server S_i is satisfied.

3. secret(NU’,auth_3,{U,S})

    This pertains to the login and authentication phase of STEP 3, and verifies whether the secrecy of N_{i1} in the login request message (A_i, UID_i,Ts) between the user U_i and IoT server S_i is satisfied.

4. secret(NS',auth_4,{S,CS})

This pertains to the login and authentication phase of STEP 5, and verifies whether the secrecy of N_{i2} in the login request message (UID$_i$, A$_i$, Veru$_i$, B$_i$, Vers$_i$, SID$_j$, Ts) between the IoT server S_j and the authentication server *CS* is satisfied.

5. secret(H(SId.H(X)),auth_5,{S,CS})

This pertains to the login and authentication phase of STEP 5, and verifies whether the secrecy of $Serinfor_j$ in the login request message (UID$_i$, A$_i$, Veru$_i$, B$_i$, Vers$_i$, SID$_j$, Ts) between the IoT server S_j and the authentication server *CS* is satisfied.

6. secret(NCS',auth_6,{CS,S})

This pertains to the login and authentication phase of STEP 11, and verifies whether the secrecy of N_{i3} in the mutual authentication message (C$_i$, D$_i$,E$_i$, Ts) between the authentication server *CS* and the IoT server S_i is satisfied.

7. secret(NS,auth_7,{CS,S})

This pertains to the login and authentication phase of STEP 11, and verifies whether the secrecy of N_{i2} in the mutual authentication message (C$_i$, D$_i$,E$_i$, Ts) between the authentication server *CS* and the IoT server S_i is satisfied.

8. secret(NU,auth_8,{CS,S})

This pertains to the login and authentication phase of STEP 11, and verifies whether the secrecy of N_{i1} in the mutual authentication message (C$_i$, D$_i$,E$_i$, Ts) between the authentication server *CS* and IoT server S_i is satisfied.

9. secret(H(X),auth_9,{CS,S})

This pertains to the login and authentication phase of STEP 11, and verifies whether the secrecy of $h(X)$ in the mutual authentication message (C$_i$, D$_i$,E$_i$, Ts) between the authentication server *CS* and the IoT server S_i is satisfied.

10. secret(NU,auth_10,{S,U})

This pertains to the login and authentication phase of STEP 13, and verifies whether the secrecy of N_{i1} in the login response message (E$_i$, Ts) between the IoT server S_i and the user U_i is satisfied.

11. witness(CS,S,auth_11,xor(H(SId.X),NS'))

This pertains to the login and authentication phase of STEP 8, and the authentication server *CS* verifies whether the IoT server S_j is authenticated through the Vers$_i$ value in the login request message (UID$_i$, A$_i$, Veru$_i$, B$_i$, Vers$_i$, SID$_j$, Ts) sent to the IoT server S_j.

12. witness(CS,U,auth_12,xor(xor(H(H(ID.H(P)).X),H(X)),NU'))

This pertains to the login and authentication phase of STEP 10, and the authentication server *CS* verifies whether the user U_i is authenticated through the $\mathrm{Vers_i}$ value in the login request message ($\mathrm{UID_i}$, $\mathrm{A_i}$, $\mathrm{Veru_i}$, $\mathrm{B_i}$, $\mathrm{Vers_i}$, $\mathrm{SID_j}$, Ts) sent to the IoT server S_j.

13. witness(S,CS,auth_13, xor(xor(NU',NCS'),H(xor(SId,NS'))))

This pertains to the login and authentication phase of STEP 12, and the IoT server S_j verifies whether the authentication server *CS* is authorized through the $C_i$value in the mutual authentication message ($\mathrm{C_i}$, $\mathrm{D_i}$,$\mathrm{E_i}$, Ts) sent to the authentication server *CS*.

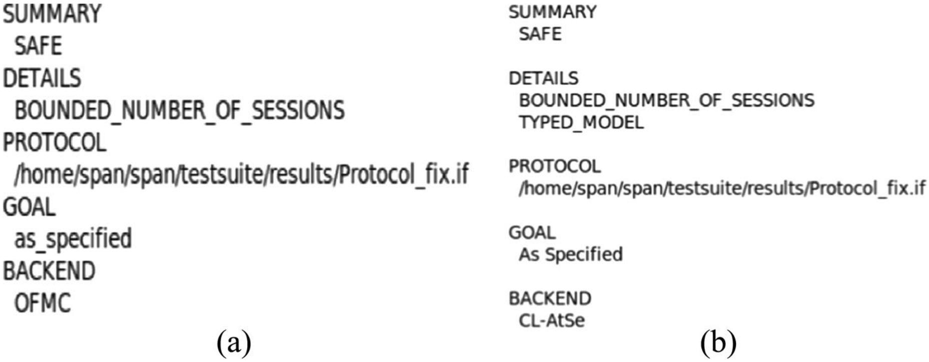
14. witness(U,S,auth_14,xor(xor(NS',NCS'),H(xor(xor(H(H(ID.H(P)).X), H(X)),NU).H(X))))

This pertains to the login and authentication phase of STEP 15, and the user U_i verifies whether the IoT server S_j is authenticated through the E_i value in the mutual authentication message (E_i, *Ts*) sent to the IoT server S_j.

In this study, the formal verification of the authentication protocol is performed through the OFMC and CL-AtSe models, as shown in Figs. 5. and 6. It can be seen that the $h(x)$, N_{i1},N_{i2}, and N_{i3} values, which should be kept secret during the process of sending/receiving messages in the proposed authentication protocol, are not exposed to an attacker. Furthermore, through the verification performed by each entity, it was possible to confirm that the authentication performed between the user U_i, IoT server S_j, and authentication server *CS* was securely authenticated.

6 Conclusion

In recent years, as the scope of applications of IoT has broadened, the amount of data generated in IoT has become enormous, and the multi-server architecture has been utilized to manage this scenario efficiently. In a multi-server IoT environment, a user can manage and receive



```
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/Protocol_fix.if
GOAL
 as_specified
BACKEND
 OFMC
```

(a)

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/Protocol_fix.if

GOAL
 As Specified

BACKEND
 CL-AtSe
```

(b)

Fig. 6 **a** Simulation result for the AVISPA tool under the OFMC model.; **b** Simulation result for the AVISPA tool under the CL-AtSe model information collected by a sensor by connecting to a server via wireless networks from remote locations. However, if a malicious attacker accesses a communication network by exploiting a vulnerable authentication system, the system can be exposed to user impersonation and session key leakage attacks. Thus, a secure authentication protocol is required to prevent this.

Therefore, in this paper, we propose a secure authentication protocol to analyze and respond to security threats that may occur in a multi-server IoT environment. The proposed authentication protocol has been shown to be secure against user impersonations, session key leakage attacks, as well as various other attacks. The verification properties are specified by utilizing the formal specification language HLPSL. By using the formal verification tool AVISPA, the security of the required verification properties has also been verified through the results of our experiments.

The authentication protocol proposed in this paper is expected to be employed in applications such as key exchanges using smart cards, as well as applications in various other fields.

Acknowledgements This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (No. NRF-2014R1A2A1A11050818) and by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2017-2015-0-00403) supervised by the IITP(Institute for Information & communications Technology Promotion).

References

1. Abdellatif R, Aslan HK, Elramly SH (2011) New real time multicast authentication protocol. International Journal of Network Security 12:13–20. https://doi.org/10.1016/j.ejrs.2011.11.003.
2. Armando A, Basin D, Boichut Y (2005) The AVISPA tool for the automated validation of internet security protocols and applications. International Conference on Computer Aided Verification 3576:281–285. https://doi.org/10.1007/11513988_27
3. Chang CC, Wu HL, Wang ZH, Mao Q (2013) An efficient smart card based authentication scheme using image encryption. J Inf Sci Eng 29:1135–1150
4. El-Emam E, Koutb M, Kelash H, Faragallah OS (2011) An authentication protocol based on Kerberos 5. International Journal of Network Security 12:159–170
5. He D, Chen J, Hu J (2011) Weaknesses of a remote user password authentication scheme using smart card. International Journal of Network Security 13:58–60
6. Hwang MS, Chong SK, Chen TY (2010) Dos-resistant ID-based password authentication scheme using smart cards. J Syst Softw 83:163–172. https://doi.org/10.1016/j.jss.2009.07.050
7. Li X, Xiong Y, Ma J, Wang W (2012) An efficient and security dynamic identity based authentication protocol for multi-server architecture using smart cards. J Netw Comput Appl 35:763–769. https://doi.org/10.1016/j.jnca.2011.11.009
8. Mittal H (2014) Diffie-Hellman based smart-card multi-server authentication scheme. Sixth International Conference on Computational Intelligence and Communication Networks. https://doi.org/10.1109/CICN.2014.173.
9. Odelu V, Das AK, Goswami A (2015) A secure biometrics-based multi-server authentication protocol using smart cards. IEEE Trans Inf Forensics Secur 10:1953–1966. https://doi.org/10.1109/TIFS.2015.2439964
10. Ruhul A, Hafizul Islam SK, Karati A (2016) Design of an enhanced authentication protocol and its verification using AVISPA. Recent Advances in Information Technology. https://doi.org/10.1109/RAIT.2016.7507936
11. Sethi P, Sarangi SR (2017) Internet of things: architectures, protocols, and applications. J Electr Comput Eng 2017:25. https://doi.org/10.1155/2017/9324035

12. Sood SK, Sarje AK, Singh K (2011) A secure dynamic identity based authentication protocol for multi-server architecture. J Netw Comput Appl 34:609–618. https://doi.org/10.1016/j.jnca.2010.11.011
13. Tanmoy M, Hafizul Islam SK, Amin R, Giri D, Khan MK, Kumar N (2010) An enhanced multi-server authentication protocol using password and smart-card: cryptanalysis and design. Security and Communication Networks. https://doi.org/10.1002/sec.1653
14. Yoon E-J, Yoo K-Y (2009) Robust multi-server authentication scheme. IFIP International Conference on Network and Parallel Computing. https://doi.org/10.1109/NPC.2009.42
15. Ziauddin S, Martin B (2013) Formal analysis of ISO/IEC 9798-2 authentication standard using AVISPA. Eighth Asia Joint Conference on Information Security. https://doi.org/10.1109/ASIAJCIS.2013.25

Won-il Bae Ms. Candidate at Department of Computer Engineering, Ajou University, Korea B.S degree from Department of Computer Engineering, Mokwon University, Korea. Interest: IoT security, Cryptographic protocols, Cloud security

Jin Kwak Professor, Department of Cyber Security, College of Information and Technology, Ajou University, Korea B.S, M.S., Ph.D. degree from Sungkyunkwan University, Korea. Interest: Cryptographic protocols, Application System Security, Secure Privacy