Perfect Forward Secrecy in VoIP Networks Through Design a Lightweight and Secure Authenticated Communication Scheme

kazem saedi (ksaedi63@gmail.com)
Greater Amman Municipality https://orcid.org/0000-0002-3669-5508
Mahdi Nikooghadam
Ferdowsi University of Mashhad Faculty of Engineering
Amirhossein Mohajerzadeh
Ferdowsi University of Mashhad Faculty of Engineering

Research

Keywords: VoIP, SIP, Perfect forward secrecy, attacks, security requirements

Posted Date: September 23rd, 2020

DOI: https://doi.org/10.21203/rs.3.rs-78831/v1

Perfect forward secrecy in VoIP networks through design a lightweight and secure authenticated communication scheme

Kazem Saedi[1*], Mahdi Nikooghadam[1], Amirhossein Mohajerzadeh[2]

Abstract

In this research, we have tried to first focus on the previous work and after getting familiar with the base papers, focus on the main paper. In this paper, we try to determine the security problems in the proposed protocols and present appropriate solutions for them. One of the subjects studied by security and encryption researchers is the matter of authentication and key agreement in SIP. Recently, an authentication and key agreement protocol in SIP has been presented in a scheme. In this paper, it was proven that their presented protocol is vulnerable to the replay attack. Such that if an attacker resends the messages sent on the public channel back to the server, the server does not notice the duplicate messages and proceeds with the session process. Also, their protocol is not resistant to the temporary parameter disclosure attack and it is possible for the attacker to discover the session key in case the temporary parameters are disclosed. Furthermore, user anonymity does neither provide re-registration prevention with the real user ID nor early detection. In this paper, we have tried to present a protocol which prevents replay and parameter disclosure attacks.

Keywords: VoIP, SIP, Perfect forward secrecy, attacks, security requirements

[1] Computer Engineering Department,Imam Reza University, Mashhad State, Islamic Republic of Iran, Full list of author information is available at the end of the article

Introduction

What is VoIP[2]?

VoIP stands for Voice over Internet Protocol and is also sometimes called the internet telephone or telephone IP[3]. This technology makes it possible to use the internet for making phone calls and unlike the traditional phones based on wire lines, uses digital technology. In fact, using the VoIP technology the voice is transmitted using IP information packets through the internet. IP-based networks are supported on all networks including organization networks, private networks, wired networks, and also wireless networks. VoIP is, in fact, applicable to all networks. In any case, any technology has its advantages and limitation. We have tried to mention these items briefly in this paper.

Introduction to the operation of SIP in the VoIP protocol

As briefly presented in the following figure, the operation of the SIP protocol in the VoIP system is as follows:

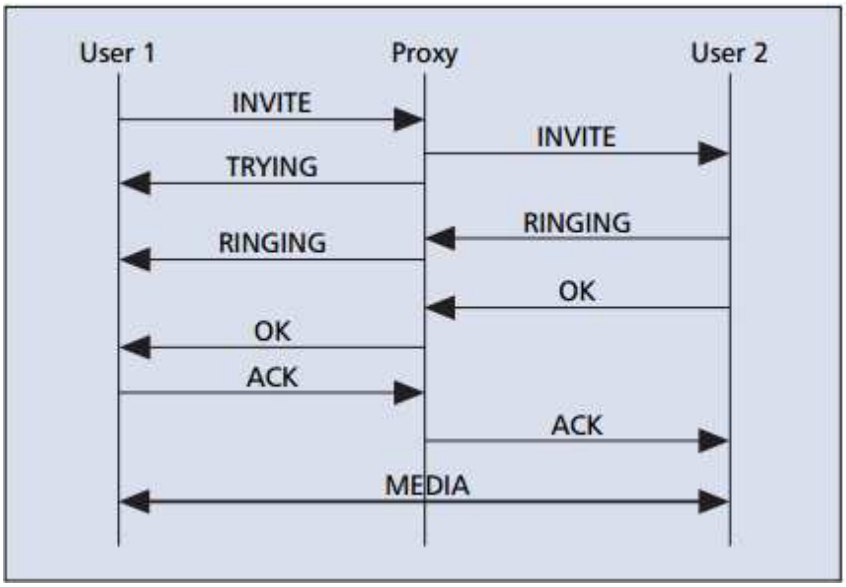


Figure 1 operation of SIP in the VoIP telephone system

- Registration: before starting a session, each one of the parties in the connection, register their current location by sending a REGISTER message to the register server.
- Making the call: in order to start a call, the client starting the connection sends an INVITE message to the server on which it has already registered

[2] Voice over Internet Protocol
[3] Internet Protocol

and declares that it wants to call its intended number. Then, the server finds the location of the user on the other side of the call or the call destination and sends the INVITE message to it. Once the INVITE message reaches the other side of the connection, the OK, RINGING, and TYPING messages will be exchanged and finally, the connection destination returns the ACK message to the client.

- Data transmission: in this stage of the connection, the RTP protocol goes into action and the data get exchanged in an end-to-end manner.
- Ending the call: After transmitting the data, either one of the parties can send the BYE message to the other one in order to request the call to be terminated. Once the OK message is received in response, the connection will end.

[4]SIP Authentication

In the SIP protocol, two-way authentication between the parties in a connection needs to be carried out in the following steps:

- Registration: At the registration step, the authentication mechanism must be used to prevent the registration of illegal users.
- Session setup: When making the call using the INVITE message, both parties need to authenticate the identity of the one on the other side.
- Session termination: When the session is being terminated by either one of the BYE or CANCEL messages, the identity of the party sending these messages should be confirmed for the user receiving the message.

Security requirements in SIP

1. Mutual Authentication: Mutual authentication means that both of the parties in the connection (client and the server) have their identity authenticated by the other party in a single protocol. It can be said that by performing correct and effective authentication in the initial steps on the server, wasting resources is prevented and the attacker will not be able to waste server resources using fake messages.
2. Session key security: This requirement is stated in the sense that at the end of the key exchange step, the session key should only be known to allowed entities, i.e. client and server, and the attacker should not be able to access the session key of either one of the parties in the session.

[4] Session Initiation Protocol

3. Perfect forward secrecy: This is one of the important security requirements in protocols. It is defined in the sense that if any of the long terms (such as the user password or the server private key) are exposed, previous session keys do not end up in the attacker's hands. In order to meet this requirement, the session key should not be solely based on the long terms. Furthermore, it must be noted that the attacker has access to the public channel information as well. Therefore, the public channel parameters should not be directly used for the creation of the session key either.
4. Known key secrecy: This requirement ensures that if the attacker obtains the current session key for both of the parties in a connection, it will not be able to discover previous session keys of the protocol (independence of the session keys from each other). Generally, random numbers are used in security protocols in each session in order to achieve this security features.
5. Fast error detection: This requirement in the two-factor protocols declares that in case the user enters the wrong username and password to the ATM machine, the card can detect this mismatch with the correct username and password and prevent costly operations such as inner product calculation on the server side. With fast detection, wasting the resources on the server can be prevented and it can provide its services to real users.
6. Re-registration on the server: In the protocols where the user ID is sent overtly on the public channel, it is possible that the attacker might re-register on the server with the user's ID and different parameters without the server noticing this re-registration by the attacker using the user's ID.

SIP attacks

In this section, we present the possible attacks on the SIP protocol:

1. Stolen verifier attack: In most of the password-based protocols, the user password gets stored in the server database in order to enable user authentication. On one hand, these databases are the main target of the attackers. If an attacker manages to get its hand on the information stored in these databases, it can forge the identity of the legitimate user or the server. On the other hand, in the protocols which are based on smart cards and password, the smart card might get stolen or lost because of negligence. Therefore, the smart card must not include any important information because it is possible to extract the information stored in the card and the attacker can take action to find the session key using this information.

2. Password guessing attack: In this attack, the attacker records the messages between the server and the legitimate SIP user and then uses the obtained messages and a password dictionary to guess the password. If this process is carried out without the server's knowledge, it is called an offline password guessing attack. The attacker can then fake the user's identity after obtaining the password and receive its intended services from the server as a legitimate user.
3. Denning-Sacco attack: This attack is defined as the attack where the attacker obtains the key of a previous session and uses it to try and find other session keys or the long terms.
4. Modification attack: The attacker tries to modify the authentication variables in this attack. This attack endangers the integrity of the information and leads to the denial of service attack.
5. Man in the middle attack (MITM): In this attack, the attacker acts as the communication interface and gets the information from both sides and changes them however it wants and delivers it to the other side. The connection parties are unaware of the existence of the third party and thing that the information is being exchanged between themselves. It is possible that the attacker merely acts as the listener and only receives the information it needs
6. Replay attack: In this attack, the attacker obtains the transmitted messaged between two entities and sends them at a different time. The connection parties do not notice that the messages are old and treat them as new messages and session key agreement steps go through. By repeatedly sending these expired messages, server resources get used pointlessly and the server becomes unable to serve the legitimate user.
7. Privilege insider attack: In this attack, the attacker is a legitimate person from the server which has obtained the information transmitted on the secure channel. Using this information, the attacker either tries to guess the user's password or fakes its identity.
8. Server spoofing attack: In this attack, the attacker obtains the REQUEST message sent to the server by the user and modifies the calculations carried out on the server side in such a way that the user does not notice the modification made by the attacker on the calculations carried out on the server side. In fact, the user believes that it has connected to the real server while the attacker has placed itself in the server's place.

9. Known session-specific temporary information attack: If the random numbers of a session get disclosed, the key of that session should not be disclosed. In order to prevent this attack, the presented method should not depend merely on the random numbers of each session but also the existence of a parameter in the session key which both the legitimate user and the server are able to obtain is essential.
10. Database injection attack: This type of attack allows the attacker to read or modify the database data by injecting code to the software or installing malicious software on a system and running its desired commands.

Overview of Recent Work

In 2015, Zhang et al. [1] presented a password-based single factor protocol in SIP. However, Lew et al. in 2016 [2] showed that the method proposed by Zhang et al. [1] is vulnerable to privilege insider attack and inappropriate mutual authentication. In the same year, Arshad and Nikooghadam [3] also presented a single factor method. However, in 2016 Lin et al. [4] showed that the method proposed by Arshad and Nikooghadam [3] is vulnerable to privilege insider attack and server spoofing attack and does not provide user anonymity. Therefore, they presented a new single factor method. In 2016, Zhang et al. [5] presented a new two-factor method in SIP for VoIP networks. In the next section, we review this method and explain the possible attacks.

Analysis of the security flaws of the method presented by Nikooghadam et al.

In 2016, Nikooghadam et al. [3] presented a method for key agreement and two-way authentication. In this section, we present the complete steps for registration and key authentication in the proposed protocol. Then, we will show that if for any reason one of the long terms gets leaked then perfect forward secrecy, which according to definition states that if any of the main parameters gets leaked for any reason the attacker should not be able to obtain the session key, is violated and the attacker can get their hand on the session key.

Registration step of the protocol presented by Nikooghadam et al.

user	server
Chooses $\boldsymbol{ID_i}$ and $\boldsymbol{PW_i}$	
Chooses a random number $\boldsymbol{r}$	
Computes	

$\boldsymbol{MPW_i = h(\ ID_i \| r \| PW_i)}$

(secure channel) $\longrightarrow$

$\{\ ID_i, MPW_i\}$

Computes $A_i = h(\ ID_i \| x)$

Computes $B_i = A_i \oplus MPW_i$

Chooses a random number N

Computes $MID_i = E_x(\ ID_i \| N)$

Stores

$\{\ B_i,\ MID_i, E_{key}(.)/\ D_{key}(.), h(.)\}$ into a smart card

$\longleftarrow$ (smart card)

Insert $\boldsymbol{r}$ into the smart card

Smart card

$\{\ \boldsymbol{B_i},\ \boldsymbol{MID_i}, \boldsymbol{r}, \boldsymbol{E_{key}}(.)$

$/\ \boldsymbol{D_{key}}(.), \boldsymbol{h}(.)\}$

Authentication step of the protocol presented by Nikooghadam et al.

User	Server
Inserts smart card	
Inputs $\boldsymbol{ID_i}$and $\boldsymbol{PW_i}$	

Selects a random number $\boldsymbol{RN_i}$ and

current time stamp $\boldsymbol{T_i}$,

Calculates

$$\boldsymbol{A_i} = \boldsymbol{B_i} \oplus \boldsymbol{h}(\boldsymbol{ID_i} \| \boldsymbol{r} \| \boldsymbol{PW_i}) = \boldsymbol{h}(\boldsymbol{ID_i} \| \boldsymbol{x})$$

$$\boldsymbol{M_1} = \boldsymbol{E}_{\boldsymbol{A_i}}(\boldsymbol{ID_i} \| \boldsymbol{RN_i} \| \boldsymbol{T_i} \| \boldsymbol{MID_i})$$

(public channel)

$\longrightarrow$

REQUEST$\{ MID_i, M_1, T_i\}$

Checks $| T_s - T_i| \leq \Delta T$

Decrypts $D_x(MID_i) = (ID_i \| N)$

and $A_i^* = h(ID_i \| x)$

$D_{A_i^*}(M_1) = (ID_i^* \| RN_i^* \| T_i^* \| MID_i^*)$

Selects two random numbers N^{new} and RN_s

CHALLENGE $\{ M_2\}$

$\longleftarrow$

(public channel)

Decrypts $\boldsymbol{D}_{\boldsymbol{A_i}}(\boldsymbol{M_2}) = (\boldsymbol{MID_i^{new}} \| \boldsymbol{RN_s} \| \boldsymbol{ID_i} \| \boldsymbol{RN_i})$

Verifies $\boldsymbol{ID_i}$ and $\boldsymbol{RN_i}$

Calculation $\boldsymbol{M_3 = h(\ RN_s \| MID_i^{new} \| \ RN_i)}$

and session key $\boldsymbol{SK = h(\ RN_i \| \ A_i \| \ RN_s)}$

Replaces

$\boldsymbol{MID_i}$ with $\boldsymbol{MID_i^{new}}$

RESPONSE $\{\boldsymbol{M_3}\}$ ⟶

Calculates

$M_3^* = h(\ RN_s \| MID_i^{new} \| \ RN_i)$

Verifies condition $M_3^* =?\ M_3$

Computes $SK = h(\ RN_i \| \ A_i^* \| \ RN_s)$

Explanation of the protocol presented by Nikooghadam et al. in 2016

In this research, we have tried to explain the protocol proposed by Nikooghadam et al. in 2016 [3] correctly and completely because in the rest of the paper, we will address the problems present in this protocol.

In the registration step, all of the messages exchanged between the user and the server go through the secure channel. This means that the attacker cannot harm the information exchanged between the user and the server in the registration step. However, it should be noted that according to the Kerkhof laws, the encryption algorithm is public and the attacker knows how each parameter is calculated and using what other parameters. Therefore, we could say that the attacker cannot use the parameters sent and received through the secure channel during the registration step and the only thing it can do in the registration step is find out how the formulas are made and calculated. If the connection parties have saved any parameters in their databases, the attacker can infiltrate these databases and access the parameters saved in them. So, the main action by the attacker takes place at the authentication and key selection step.

In the registration step, the main objective of the user is to deliver its information to the server and get the information of the server. The mutual information of the parties is then stored in a smart card. In fact, this smart card acts as a receipt which the server gives to the user after registration. After the registration step, it is time for the authentication and session key agreement step. The parties are trying to agree on a session key so that they can be sure any information exchanged between them cannot be accessed by the attacker.

Any message sent either by the user or the server at the authentication step is first checked by the receiver to make sure if the message was sent by the other real party in the connection. In fact, the message needs to be verified. Once the message is verified (meaning that the message was sent by the other party and hasn't been modified on the way) the required actions and calculations are carried out on the message.

In the rest of this section, in order to help better understand the protocol, some parts are explained briefly. However, while reviewing other papers we will explain them comprehensively, for example:

The equation Checks $|T_s - T_i| \leq \Delta T$ which is seen in the above protocol is meant to stop the replay attack. It, in fact, uses time variables to prevent the attacker from capturing the message from the channel and resend it. Indeed, the message will be checked to see if it is new.

Unlike the registration step, in the authentication step, all of the messages between the user and the server are sent and received through the public channel. This makes it possible for the attacker to access some of the parameters and it is possible that it can work out the session key using these parameters. The main flaws of the main paper are presented in this section.

Explanation of the perfect forward secrecy attack on the Nikooghadam et al. method

We will prove that in the method presented by Nikooghadam et al. if the private key of the server (x) is revealed, according to the premise of this attack, for any reason then the attacker will be able to access the mutual session key of the parties $SK = h(RN_i \| A_i \| RN_s)$.

In the first message sent in the authentication and key agreement phase, parameters M_1 and MID_i are sent on the public channel. The attacker obtains the public channel information and uses the server private key (x) to decrypt MID_i and work out the parameters ID_i and N using equation $D_x(MID_i) = (ID_i \| N)$. Then, the

attacker calculates parameter A_i using equation $A_i = h(ID_i \| x)$. Then, the attacker find gets the value of parameter RN_i by decrypting M_1 and using equation $D_{A_i}(M_1) = (ID_i \| RN_i \| T_i \| MID_i)$.

In the second message transmitted in the authentication and key agreement phase, parameter M_2 is sent to the user through a public channel. Then, the attacker decrypts parameter M_2 and $D_{A_i}(M_2) = (MID_i^{new} \| RN_s \| ID_i \| RN_i)$ and gets the value of parameter RN_s. Considering the session key equation $SK = h(RN_i \| A_i \| RN_s)$, now the attacker can work out the other parameters used to create the session key and obtain the mutual session key used by the parties, provided that it has the private server key (x). Therefore, it was proven that the attacker can obtain the session key of the parties and will have access to all of the messages which are exchanged between the server and the user. This makes the security of the proposed protocol be questioned completely.

So, it was proven in this stage that the idea proposed by Nikooghadam et al. [3] in 2016 is not fully valid and this idea has been unable to meet the perfect forward secrecy security requirement. If we search deeper in the papers in this field, we will realize that meeting the perfect forward secrecy is a difficult task. As mentioned in the paper presented by Nikooghadam et al. in 2018, it has been declared that the perfect forward secrecy security requirement has been met in VoIP networks. However, we will prove in the following sections that this idea has been unable to meet this security requirement.

Review of the paper presented by Zhang et al.

In this section, we review the steps of the two-factor protocol presented by Zhang et al. This protocol, like other protocols presented in the authentication and key agreement in SIP field, includes the following main steps: registration, authentication and key agreement (login), and password change. The steps of this method and details of each step, and also the security flaws of their presented method are reviewed in this section.

The symbols used in Zhang's protocol are presented in the following table

The symbols used in the protocol presented by Zhang et al.

Symbol	Definition
$\boldsymbol{U_i}$	User i
$\boldsymbol{S}$	*SIP* Server
$\boldsymbol{ID_i}$	ID of user i
$\boldsymbol{PW_i}$	Password of user i
$\boldsymbol{s}$	Private key of S
$\boldsymbol{p}$	A prime power
$\boldsymbol{P}$	Critical point on oval curve E
$\boldsymbol{F_p}$	First order field $E_p(a, b)$
$\boldsymbol{E_p(a, b)}$	Equation of the oval curve
$\boldsymbol{r, r_1, r_2, r_3, r_4}$	Random numbers
$\boldsymbol{SK}$	Session key
$\boldsymbol{E_k(.)}$	Symmetrical encryption and decryption with private key k
$\oplus$	(*XOR*)
$\|\|$	Connect operation
$(\boldsymbol{Q})_x/(\boldsymbol{Q})_y$	Values of x and y in the oval curve Q
$\boldsymbol{h(.)}$	Hash function

Initial Step

Part one: The oval curve equation $E_p(a, b): y^2 = x^3 + ax + b(mod\ p)$ on the first order oval curve F_p is selected by server SIP.

Part two: The server selects a random number s as its private key and a hash function. Then, it calculates $P_{pub} = sP$.

Part three: The private key s gets protected by the server and parameters $\{E_p(a, b), P, P_{pub}, h(.)\}$ get published publicly.

Registration Step

Step one: The user sends its ID ID_i and password PW_i and a random number r. Then, it calculates c_1 using equation $c_1 = h(PW_i \oplus r)$. Afterward, the user sends the parameters $\{ID_i, c_1\}$ to the server through the secure channel.

Step two: Once the server has received the information, it calculates c_2 and c_3 using equations $c_2 = h(ID_i \oplus s)$ and $c_3 = c_1 \oplus c_2$. Then, it stores parameter c_3 in the smart card and sends the card to the user through the secure channel.

Step three: Once the user has received the smart card, it saves the random parameter r in the card.

The registration step of Zhang's protocol is presented in the following figure.

The registration step in the protocol presented by Zhang et al.

User		Server
Chooses $\boldsymbol{ID_i}, \boldsymbol{pw_i}, \boldsymbol{r}$		
Calculates $\boldsymbol{c_1 = h(pw_i \oplus r)}$	$\{\boldsymbol{ID_i}, \boldsymbol{c_1}\}$ → (Secure channel)	
		Computes $c_2 = h(ID_i \oplus s)$
		$c_3 = c_1 \oplus c_2$
		Stores c_3 into the smart card
	Smart card ← (Smart card)	
Saves $\boldsymbol{r}$ in to the smart card		

Authentication and key agreement step

At the end of the authentication and key agreement step, connection parties, i.e. the server and the user, get the secure mutual session key.

First step: First, the user inserts its smart card into the ATM machine and enters its ID and password. Then, the smart card selects two random numbers r_1 and r_2 and carries out the following calculations.

$c_2 = c_3 \oplus h(PW_i \oplus r) = h(ID_i \oplus s)$,

$c_4 = rP$,

$c_5 = r_1 c_2 P_{pub}$,

$c_6 = h(c_5) \oplus (h(ID_i \oplus s) \oplus r_2 \| (c_5)_x \| (c_5)_y)$.

$(c_5)_x$ points to point c_5 on dimension x while $(c_5)_y$ points to point c_5 on dimension y. After carrying out the above calculations, message REQUEST $\{ID_i, c_4, c_6\}$ is sent to the server by the user.

Step two: After receiving the REQUEST message, the server calculates the value of c_2 using equation $c_2 = h(ID_i \oplus s)$. Then, it calculates the $\big(h(ID_i \oplus s) \oplus r_2 \| (c_5)_x \| (c_5)_y\big)$ value using equation $h(ID_i \oplus s) \oplus r_2 \| (c_5)_x \| (c_5)_y = h(sc_2c_4) \oplus c_6$. In case the condition is met, the server obtains the value of parameter r_2 using equation $c_2 \oplus h(ID_i \oplus s) \oplus r_2$. Then, the server selects two random numbers and gets the parameters c_7 and SK from equations $c_7 = r_3 P$ and $SK = h(c_4 \| r_3 c_4 \| c_7)$. Afterward, the server calculates the authentication parameter $Auth_s$ using equation $Auth_s = h(h(ID_i \oplus s) \| r_2 \| (SK)_x \| (c_5)_x \| (SK)_y \| (c_5)_y)$ and send the CHALLENGE message containing parameters $\{realm, c_7, Auth_s, r_4\}$ to the user.

Step three: After receiving the CHALLENGE $\{realm, c_7, Auth_s, r_4\}$ message, the smart card calculates the session key using equation $SK = h(c_4 \| r_1 c_7 \| c_7)$. Then, the smart card checks the $h\big(c_2 \| r_2 \| (SK)_x \| (c_5)_x \| (SK)_y \| (c_5)_y\big) =? Auth_s$ equation after calculating $h\big(c_2 \| r_2 \| (SK)_x \| (c_5)_x \| (SK)_y \| (c_5)_y\big)$. In case the equation conditions are met, the parties agree on a session key. Then, the user obtains the authentication parameter $Auth_u$ using equation $Auth_u = h((SK)_x \| (r_4 + 1) \| (SK)_y)$. Otherwise, the session ends. Then, the user sends the RESPONSE $\{realm, Auth_u\}$ message to the server.

Step four: After the RESPONSE $\{realm, Auth_u\}$ message is received by the server, it checks whether the condition $Auth_u = h((SK)_x \| (r_4 + 1) \| (SK)_y)$ is met. In case the condition was not met, the session ends. However, if the condition is satisfied, the session key for the parties will be equal to $SK = r_1 r_3 P$.

The authentication and key agreement step in Zhang's protocol is presented in the following table.

The authentication and key agreement step in the protocol presented by Zhang et al.

User	Server
Inserts smart card	
Inputs $\boldsymbol{ID_i}$ and $\boldsymbol{pw_i}$	
Chooses $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$	
Calculates	
Chooses $\boldsymbol{c_2 = c_3 \oplus h(pw_i \oplus r)}$ $\boldsymbol{= h(ID_i \oplus s)}$	
$\boldsymbol{c_4 = rP}$ and $\boldsymbol{c_5 = r_1 c_2 P_{pub}}$	
$\boldsymbol{c_6 = h(c_5) \oplus (h(ID_i \oplus s) \oplus r_2 \| (c_5)_x \| (c_5)_y)}$	
REQUEST $\{\boldsymbol{ID_i, c_4, c_6}\}$ $\longrightarrow$ (Public channel)	
	Computes $c_2 = h(ID_i \oplus s)$
	$(h(ID_i \oplus s) \oplus r_2 \| (c_5)_x \| (c_5)_y =$ $h(sc_2c_4) \oplus c_6$
	Verifies

$(c_5)_x \| (c_5)_y =? (s\ c_2\ c_4)_x \| (s\ c_2\ c_4)_y$

Computes $r_2 = c_2 \oplus h(\ ID_i \oplus s) \oplus r_2$

Selects $r_3,\ r_4$

Calculates $c_7 = r_3 P$

Session key $SK = h(c_4 \| r_3\ c_4 \| c_7)$

$Auth_s$
$= h(h(ID_i$
$\oplus s) \| r_2 \| (SK)_x \| (c_5)_x \| (SK)_y \| (c_5)_y)$

CHALLENGE $\{\boldsymbol{realm}, \boldsymbol{c_7}, \boldsymbol{Auth_s}, \boldsymbol{r_4}\}$ ←

(Public channel)

Calculates $\boldsymbol{sk = h(c_4 \| r_1\ c_7 \| c_7)}$

Checks

$\boldsymbol{h(c_2 \| r_2 \| (sk)_x \| (c_5)_x \| (sk)_y \| (c_5)_y)}$=?
$\boldsymbol{Auth_s}$

Computes

$\boldsymbol{Auth_u}$=? $\boldsymbol{h((sk)_x \| (r_4 + 1) \| (sk)_y)}$

RESPONSE $\{\boldsymbol{realm}, \boldsymbol{Auth_u}\}$ →

(public channel)
Checks
$Auth_u$ =? $h((SK)_x \| (r_4+1) \| (SK)_y)$

Changing the user password step

In this step, the user can change its password.

Step one: First, the user selects a new password (PW_i^*) and a random number (r^*) and a nonce R. Then, the user enters its previous password (PW_i) and calculates parameters Z and V using equations $h(PW_i \oplus r) \oplus c_3$ and $E_{(SK)_x}(h(PW_i^* \oplus r^*) || ID_i \| R \| Z)$, respectively. Then the user sends parameter V to the smart card.

Step two: After the smart card has received parameter V, the card decrypts this message using the session key ($D_{(SK)_x}(V)$) and calculates ($h(PW_i^* \oplus r^*) \| ID_i \| R \| Z$) and checks if $h(ID_i \oplus s)$ and Z are equal. In case the equality does not stand, the password change request gets rejected and otherwise, the card calculates parameters $c_3^* = h(PW_i^* \oplus r^*) \oplus h(ID_i \oplus s)$ and $W = E_{(SK)_x}(c_3^* \| h(c_3^* \| (R+1)))$. Then, it sends parameter W to the user.

Step three: After the user has received W, in order to verify the tag value ($h(c_3^* \| (R+1))$), W gets decrypted. If the value is correct, the smart card replaces the old value (c_3, r) with the new value (c_3^*, r^*).

The user password changing step of Zhang's protocol is presented in the following table.

User password changing step in the protocol presented by Zhang et al.

User	Smart card
Selects $\boldsymbol{pw_i^*}$, $\boldsymbol{r^*}$and $\boldsymbol{R}$	

Enters old password $\boldsymbol{pw_i}$

Computes $\boldsymbol{Z = h(pw_i \oplus r) \oplus c_3}$

$\boldsymbol{V = E_{(SK)x}\ (h(pw_i^* \oplus r^*) \| ID_i \| R \| Z)}$

$\{\boldsymbol{V}\}$ ⟶

$D_{(SK)_x}(V) = (h(PW_i^* \oplus r^*) \| ID_i \| R \| Z)$

Checks $h(ID_i \oplus s) =? Z$

Calculates

$c_3^* = h(PW_i^* \oplus r^*) \oplus h(ID_i \oplus s)$

$W = E_{(SK)_x}(c_3^* \| h(c_3^* \| (R+1)))$

⟵ $\{\boldsymbol{W}\}$

Verifies authentication tag

$\boldsymbol{h(c_3^* \| (R+1))}$

Updates $(\boldsymbol{c_3, r})$ with $(\boldsymbol{c_3^*, r^*})$

Review of the attacks on the scheme presented by Zhang et al.

- **Replay attack:** While sending the REQUEST message, the server performs calculations for the repetitive requests sent by the attacker too. In fact, we could say that the server does not notice that the messages are duplicate and performs the calculation for these messages as well. As a result of various and repeated messages being sent by the attacker, the resources of the server get wasted and the server becomes unable to serve legitimate users. In fact,

the replay attack leads to the denial of service attack and results in server failure.

- **Known session-specific temporary information attack:** Since C_4 and C_7 are sent on the public channel, which is publicly audible, the attacker can obtain the information transmitted on this channel. Therefore, in case the random variables r_1 and r_3 get leaked, the attacker can obtain the mutual session key used by the parties in the connection.
- **Re-registration:** In protocols where the user ID is sent overtly through the public channel, it is possible for the attacker to re-register on the server using the user ID and reselecting other parameters without the server noticing the registration of the attacker using this duplicate user ID. In the method presented by Zhang et al., because of the user ID being sent overtly and the attacker's knowledge of it, a foreign attacker can re-register by selecting a new pw_i and r and using user's ID_i. The server does not notice the re-registration and issues a smart card for the attacker as well.
- **Anonymity:** Sending the user ID overtly on the public channel eliminates the user anonymity. In fact, by sending the user ID on the public channel, the attacker notices the times at which the user uses the server services and, in a sense, the user's privacy gets violated.
- **Fast error detection:** In the two-factor protocol presented by Zhang et al., the smart card is not able to detect if the user enters the incorrect ID and password. Therefore, the server needs to carry out the costly multiplication on the oval curve for each user in order to verify the validity of the information entered by the user. When the number of users or the number of their request grows, performing this calculation each time gets costly and leads to server failure and denial of service. By creating the denial of service attack, the server becomes unable to provide services to legitimate users.

In this section, we review the proposed two-factor protocol. The proposed protocol, like the other protocols presented in the authentication and key agreement in SID field, consists of the following main steps: registration, authentication and key agreement (login), and password changing. The steps of this method and the details of each step will be reviewed in this section. At the beginning of this section, we present the symbols used in the protocol in the following table. Then, the registration, login, and password changing steps and their details are reviewed [5].

Review of Nikooghadam et al. 2018 paper

The symbols in the proposed protocol

Symbol	Definition
$\boldsymbol{U_i}$	User i
$\boldsymbol{S}$	*Server SIP*
$\boldsymbol{ID_i}$	ID of user i
$\boldsymbol{PW_i}$	Password of user i
$\boldsymbol{ID_{sc}}$	Card ID
$\boldsymbol{x_s}$	Private key *S*
$\boldsymbol{P}$	The critical point on the oval curve *E*
$\boldsymbol{r_i, b_i, r_c, r_s}$	Random numbers
$\boldsymbol{SK}$	Session key
$\boldsymbol{E_k(0)/ D_k(0)}$	Symmetrical encryption and decryption with private key k
$\oplus$	*XOR*
$\|\|$	Connect operation
$\boldsymbol{T_1, T_2, T_3, T_4, T_5}$	The timestamp of the user and the server
$\boldsymbol{h(0)}$	Hash function

Registration step

The user and the server carry out the following steps in a face to face meeting and at the end of this step, the server presents the smart card to the user.

Step one: At first, the user selects a user ID and a password along with two random numbers b_i and r_i. Then, it calculates the values of IP_i and RB with respect to previously selected parameters using their formulas.

$$RB = h(r_i \| b_i)$$

$$IP_i = h(h(ID_i \| PW_i \| RB) \bmod m)$$

m is a number between 2^8 and 2^{16} that makes it impossible for the attacker to guess the user ID and the password simultaneously. Then, the user sends parameters $\{ID_i, IP_i, b_i\}$ to the server through the secure channel.

Step two: After parameters $\{ID_i, IP_i, b_i\}$ are received by the server, it calculates parameters A_i, B_i, and NID_i using the following equations.

$A_i = h(SID_i \| x_s \| ID_{sc} \| ID_i)$

$B_i = A_i \oplus IP_i$

$NID_i = ID_i \oplus IP_i$

Afterward, the server selects timestamp T_1 and calculates parameter RID_i using equation $E_{x_s}(ID_{sc} \| ID_i \| T_1)$. The server puts parameters RID_i and B_i inside the smart card and sends it to the user through the secure channel. The server also saves parameters $\langle NID_i, status, b_i \rangle$ in its database. When the user logs into the server, the $status$ field is set to one and once the user logs out, this field is set to zero on the server's database.

Step three: After the smart card is received by the user, the following calculations are carried out.

$K_i = B_i \oplus r_i$

$W_i = h(h(b_i \| PW_i \| ID_i \| ID_{sc}) \ mod \ m)$

$H_i = W_i \oplus r_i$

Then, the smart card removes parameter B_i and stores parameters $\{b_i, K_i, H_i, ID_{sc}\}$ in the card. In the end, the smart card contains the following values: $\{RID_i, b_i, K_i, H_i, ID_{sc}, E_{key}(.)/ D_{key}(.), h(.)\}$.

The registration step of the proposed method is presented in the table.

The registration step of the proposed protocol

User	Server
Selects $\boldsymbol{ID_i}$, $\boldsymbol{pw_i}$ and random numbers $\boldsymbol{r_i}$ and $\boldsymbol{b_i}$	
Computes $\boldsymbol{RB = h(r_i \| b_i)}$	
$\boldsymbol{IP_i = h(h(ID_i \| pw_i \| RB) \ mod \ m}$	
	$\{ID_i, IP_i, b_i\}$ → (Secure channel)

Calculates $A_i = h(SID_i \| x_s \| ID_{sc} \| ID_i)$

$B_i = A_i \oplus IP_i$

$NID_i = ID_i \oplus IP_i$

Selects time stamp T_1

Computes $RID_i = E_{x_s}(ID_{sc} \| ID_i \| T_1)$

Stores $\{RID_i, B_i\}$ in the smart card

Saves $\langle NID_i, status, b_i \rangle$ in Database

Smart card

(Secure channel)

Computes

$\boldsymbol{K_i = B_i + r_i}$

$\boldsymbol{W_i = h(h(b_i \| pw_i \| ID_i \| ID_{sc}) \bmod m}$

$\boldsymbol{H_i = W_i + r_i}$

Deletes $\boldsymbol{B_i}$

Saves $\{\boldsymbol{b_i, K_i, H_i, ID_{sc}}\}$ in smart card

Smart card

$\{\boldsymbol{RID_i, b_i, K_i, H_i, ID_{sc}, E_{key}(0)/}$
$\boldsymbol{D_{key}(0), h(0)}\}$

Authentication and key agreement step

Step one: First, the user inserts its smart card into the ATM machine and enters its user ID and password. Then, the following calculations are performed by the smart card.

$W_i^* = h(h(b_i \| PW_i^* \| ID_i^* \| ID_{sc}^*) mod\ m)$

$r_i^* = H_i \oplus W_i^*$

$B_i^* = K_i \oplus r_i^*$

$RB^* = h(r_i^* \| b_i)$

$IP_i^* = h(h(ID_i^* \| PW_i^* \| RB^*) mod\ m)$

$A_i^* = B_i^* \oplus IP_i^*$

Then, the smart card selects a timestamp T_2 and a random number r_c and calculates parameter EA_i using equation $E_{A_i^*}(A_i^* \| r_c \| T_2 \| IP_i^*) = EA_i$. Then, it sends the REQUEST $\{EA_i,\ RID_i, T_2\}$ message to the server through the public channel.

Step two: Immediately after the REQUEST $\{EA_i,\ RID_i, T_2\}$ message is received by the server, the freshness of the message is verified using equation $|T_3 - T_2| \leq \Delta T$. In case the message is new, the server decrypts parameter RID_i using its private key x_s according to equation $D_{x_s}(RID_i) = (ID_{sc}^* \| ID_i^* \| T_1)$. Then, it calculates parameter $A_i^* = h(SID_i \| x_s \| ID_{sc} \| ID_i)$. Then, parameter EA_i is decrypted using the calculated A_i^* parameter and parameters A_i ، r_c ، T_2' are obtained. Afterward, the server compares parameter A_i with parameter A_i^* and verifies the validity of equation $|T_3 - T_2'| \leq \Delta T$. In case the parameters are not equal, the session gets terminated. However, if the parameters are equal, the user gets authenticated by the server. Afterward, the server calculates NID_i^* using equation $ID_i \oplus IP_i$ and queries its database for parameter NID_i^*. Then, it extracts $\langle NID_i, status,\ b_i \rangle$ information pertaining to NID_i. If the $status$ field is equal to one, the session gets terminated because it means that the user is currently receiving a service from the server. Otherwise, timestamp T_4 and random number r_s are selected by the server. After performing the above operation, the following calculations are carried out and finally the CHALLENGE $\{RID_i, T_4, EA_2\}$ message is sent.

$SK = h(A_i \| r_s \| r_c \| IP_i \| b_i)$

$m = h(IP_i \| r_c \| r_s \| SK)$

$EA_2 = E_{IP_i}(IP_i \| r_c \| r_s \| T_4)$

$RID_i' = E_{x_s}(ID_{sc} \| ID_i \| T_4)$

Step three: After the CHALLENGE $\{RID_i, T_4, EA_2\}$ message is received by the user, it uses equation $|T_5 - T_4| \leq \Delta T$ to ensure the freshness of the message. In case the equation is not valid, the session gets terminated. However, if the equation

stands, the smart card decrypts parameter EA_2 using parameters IP_i and obtains the $(IP_i^*, r_c^*, r_s^*, T_4)$ values which might have been modified. Then, it checks the validity of equations $IP_i^* = IP_i$ and $r_c^* = r_c$. In case the above equations stand, the server gets authenticated by the smart card and the session key and parameter m are calculated. Then, parameter RID_i replaces parameter RID_i'.

$SK = h(A_i \| r_s \| r_c \| IP_i \| b_i)$,

$m^* = h(IP_i \| r_c \| r_s \| SK)$.

Then, the smart card sends the RESPONSE $\{m^*\}$ message through the public channel.

Step four: After the server receives the RESPONSE $\{m^*\}$ message, it verifies the validity of parameter $m^* =? m$ and sets the value of the *status* field to one. After calculating the session key, *status* resets to zero. The authentication and key agreement step of the proposed method is presented in the table.

The authentication and key agreement step of the proposed protocol

User	Server
Inserts smart card Enters $\boldsymbol{ID_i}$ and $\boldsymbol{pw_i}$ Computes: $\boldsymbol{W_i^* = h(h(b_i \| pw_i^* \| ID_i^* \| ID_{sc}^*) mod\ m}$ $\boldsymbol{r_i^* = H_i \oplus W_i^*}$ $\boldsymbol{B_i^* = K_i \oplus r_i^*}$, $\boldsymbol{RB^* = h(r_i^* \| b_i)}$, $\boldsymbol{IP_i^* = h(h(ID_i^* \| PW_i^* \| RB^*) mod\ m)}$, $\boldsymbol{A_i^* = B_i^* \oplus IP_i^*}$. Selects a time stamp $\boldsymbol{T_2}$ and a random number $\boldsymbol{r_c}$	

Calculates $\boldsymbol{E_{A_i^*}(A_i^* \| r_c \| T_2 \| IP_i^*) = EA_i}$

REQUEST $\{EA_i,\ RID_i, T_2\}$ →

(public channel)

Checks whether $|T_3 - T_2| \leq \Delta T$

Decrypts RID_i

$D_{x_s}(\ RID_i) = (ID_{sc}^* \| ID_i^* \| T_1)$

Calculates

$A_i^* = h(SID_i \| \ x_s \| \ ID_{sc} \| \ ID_i)$

Decrypts EA_i $D_{A_i}(EA_i) = (A_i \| r_c \| \ T_2{}' \| \ IP_i)$

Checks A_i =?A_i^*, and $|T_3 - T_2{}' \ | \leq \Delta T$

Calculates $NID_i{}^* = ID_i \oplus IP_i$,

Searches $NID_i{}^*$ in database , b_i

Chooses a time stamp T_4

and a random number r_s

Computes $SK = h(A_i \| r_s \| r_c \| IP_i \| b_i)$

$EA_2 = E_{IP_i}(IP_i \| r_c \| r_s \| T_4)$

$RID_i{}' = E_{x_s}(ID_{sc} \| ID_i \| T_4)$

← CHALLENGE $\{RID_i, T_4, EA_2\}$

(public channel)

Checks $\boldsymbol{|T_5 - T_4| \leq \Delta T}$

Decrypts $\boldsymbol{EA_2}$

$\boldsymbol{D_{IP_i}(EA_2)= (IP_i^*, r_c^*, r_s^*, T_4)}$

Checks $\boldsymbol{IP_i^* = IP_i}$ *and* $\boldsymbol{r_c^* = r_c}$

Calculates $\boldsymbol{SK = h(A_i \| r_s \| r_c \| IP_i \| b_i)}$,

and $\boldsymbol{m^* = h(IP_i \| r_c \| r_s \| SK)}$

Replaces $\boldsymbol{RID_i}$ with $\boldsymbol{RID_i'}$

(public channel) →

RESPONSE $\{m^*\}$

Checks $m^* =? m$

Updates $status = 1$

$$SK = h(A_i \| r_s \| r_c \| IP_i \| b_i)$$

Password changing step

Step one: First, the user inserts its smart card into the ATM machine and enters its ID and password. Then, the following calculations, which might be fake, are carried out by the smart card.

$W_i^* = h(h(b_i \| PW_i^* \| ID_i^* \| ID_{sc}^*) mod\ m)$

$r_i^* = H_i \oplus W_i^*$

$B_i^* = K_i \oplus r_i^*$

$RB^* = h(r_i^* \| b_i)$

$IP_i^* = h(h(ID_i^* \| PW_i^* \| RB^*) mod\ m)$

$A_i^* = B_i^* \oplus IP_i^*$

Afterward, the smart card selects the timestamp T_2 and a random number r_c and calculates parameter EA_i using equation $E_{A_i^*}(A_i^* \| r_c \| T_2 \| IP_i^*) = EA_i$. Then, it sends the REQUEST $\{EA_i, RID_i, T_2\}$ message to the server through the public channel.

Step two: Immediately after the server receives the REQUEST $\{EA_i,\ RID_i, T_2\}$ message, the freshness of the message is verified using equation $|T_3 - T_2| \leq \Delta T$. Doing so prevents the replay attack and avoids the extra calculations on the server. In case the message is new, the server decrypts parameter RID_i using its private key x_s according to $D_{x_s}(RID_i) = (ID_{sc}^* \| ID_i^* \| T_1)$. Then, it calculates parameter $A_i^* = h(SID_i \| x_s \| ID_{sc} \| ID_i)$. Then, parameter EA_i is decrypted and parameters A_i,r_c, T_2', and IP_i are obtained using A_i^*. Afterward, the server calculates parameter A_i using parameter A_i^* and checks the validity of equation $|T_3 - T_2'| \leq \Delta T$. In case the parameters are not equal, the session gets terminated. However, if the equation stands, the user gets authenticated for the server. Then, the server calculates NID_i^* using equation $ID_i \oplus IP_i$ and searches its database for parameter NID_i^* and extracts the $\langle NID_i, status,\ b_i \rangle$ information according to NID_i. If the $status$ field is equal to one, the session gets terminated. Otherwise, the server selects a timestamp T_4 and a random number r_s. After doing the above operations, the following calculations are carried out. Finally, the CHALLENGE $\{RID_i, T_4, EA_2\}$ message is sent.

$$SK = h(A_i \| r_s \| r_c \| IP_i \| b_i)$$

$$m = h(IP_i \| r_c \| r_s \| SK)$$

$$EA_2 = E_{IP_i}(IP_i \| r_c \| r_s \| T_4)$$

$$RID_i' = E_{x_s}(ID_{sc} \| ID_i \| T_4)$$

Step three: Once the user receives the CHALLENGE message at time T_5, first it checks the freshness of the message using equation $|T_5 - T_4| \leq \Delta T$. If the equation does not stand, the session gets terminated. However, if the equation is valid, the smart card decrypts parameter EA_2 using parameter IP_i and obtains values $(IP_i^*, r_c^*,\ r_s^*, T_4)$ which might have been modified. Then, it verifies the validity of $IP_i^* = IP_i$ and $r_c^* = r_c$ in order to authenticate the server. Afterward, the smart card receives a request for a new PW_i^{new} from the user and calculates the values of $(IP_i^{new}, B_i^{new}, W_i^{new}, K_i^{new}, H_i^{new})$ using the following equations.

$$IP_i^{new} = h(h(ID_i \| PW_i^{new} \| RB) mod\ m)$$

$$B_i^{new} = B_i^{old} \oplus IP_i^{old} \oplus IP_i^{new}$$

$$W_i^{new} = h(h(\ b_i \| PW_i^{new} \| \ ID_i \| \ ID_{sc}) mod\ m)$$

$$K_i^{new} = B_i^{new} \oplus \ r_i$$

$$H_i^{new} = W_i^{new} \oplus \ r_i$$

Finally, the smart card replaces $(\ K_i,\ H_i)$ with $(\ K_i^{new}, H_i^{new})$.

Informal proof of the security of the method proposed in the Nikooghadam 2018 paper

1) User anonymity: The user's ID should not be sent overtly in the REQUEST, CHALLENGE, and RESPONSE messages. Also, in case the attacker steals or finds the smart card and extracts the $\{RID_i, b_i, K_i, H_i, ID_{sc}, E_{key}(.)/D_{key}(.), h(.)\}$ information from it and obtains the $\{EA_i, RID_i, T_2\}$, REQUEST, and CHALLENGE $\{RID_i', T_4, EA_2\}$ messages, it should not be able to find the user's ID.
$RID_i = E_{x_s}(ID_{sc} \| ID_i \| T_1)$
$EA_i = E_{A_i^*}(A_i^* \| r_c \| T_2 \| IP_i)$
$EA_2 = E_{IP_i}(IP_i \| r_c \| r_s \| T_4)$
$RID_i' = E_{x_s}(ID_{sc} \| ID_i \| T_4$
$m^* = h(IP_i \| r_c \| r_s \| SK)$.
In order to obtain the user ID, the attacker needs to know the server's private key x_s while the server's private key is confidential. Therefore, the proposed protocol provides user anonymity.
2) Perfect forward secrecy: If the user's PW_i or the server's private key x_s gets leaked, the user should not be able to access the session key. In this protocol, the attacker cannot obtain the correct session key even if the server's private key x_s gets leaked because only the user and the server are aware of the random number b_i. On the other hand, RB in equation $IP_i = h(h(ID_i \| PW_i \| RB) \, mod \, m)$ is hidden and working out parameter IP_i is not possible provided that the user's PW_i gets leaked. Therefore, the attacker does not obtain the session key.
3) Known-key secrecy: Random parameters r_s and r_c are selected by the parties. Also, they are different for each session and do not depend on previous r_s and r_c. Therefore, leakage of the user's previous session keys does not enable the attacker to obtain the new session key.
4) Session key security: In this protocol, only the parties, i.e. the user and the server, will be able to obtain the session key. Considering the inability of the attacker to obtain parameters A_i, b_i, and IP_i and its lack of access to random parameters r_s and r_c, the attacker is not able to work out the session key.
5) Leakage of the temporary parameters of a session: In case temporary parameters r_s, r_s, and b_i get leaked, the attacker should not be able to obtain the correct session key. Considering that the attacker's

unawareness of parameters x_s, PW_i, and ID_i, it cannot access the right session key.

6) Offline password guessing attack: In case the attacker obtains the REQUEST, CHALLENGE, and RESPONSE messages, it will not be able to guess the user's password because:
$RID_i = E_{x_s}(ID_{sc} \| ID_i \| T_1)$
$EA_i = E_{A_i^*}(A_i^* \| r_c \| T_2 \| IP_i)$
$EA_2 = E_{IP_i}(IP_i \| r_c \| r_s \| T_4)$
$RID_i{}' = E_{x_s}(ID_{sc} \| ID_i \| T_4$
$m^* = h(IP_i \| r_c \| r_s \| SK)$.
Considering each one of the messages, the attacker is unable to guess the user's password. Even if the attacker steals or find the smart card and extract its information, because of $mod\, m$ it will be unable to guess the user's ID and password.

1) Impersonation:
   a. Impersonating the user: In order for the attacker to impersonate the user and its REQUEST messages, the attacker needs to send reliable messages to the RESPONSE server. Therefore, the attacker needs to know m^* and $EA_i{}^*$ values. In order to calculate the valid $EA_i{}^*$ value, RB, ID_i, and PW_i values are needed. But these values are kept confidential by the user. Also, the attacker must know the IP_i ،r_s، r_c, and SK values in order to calculate the valid m^* value. However, as proven in part 4 of section 6-1, the attacker cannot obtain the mutual session key of the user and the server, because it does not know parameters r_s and r_c.
   b. Impersonating the server: In order to impersonate the server, the attacker must be able to create the valid $\{RID_i{}^*, T_4{}^*, EA_2{}^*\}$ CHALLENGE message and send it to the user. To obtain $EA_2{}^*$, the attacker needs to know parameters IP_i, A_i and the random number r_c generated by the user. However, as mentioned in part 6 of section 6-1, the attacker is not able to simultaneously guess the user's ID and password. Moreover, the attacker does not know parameter x_s which is the server's private key and RB. Therefore, the attacker is unable to impersonate the server.

2) Stealing the verifiers: Since hidden parameters, i.e. server's private key and user's password, are not stored in the server's database, the attacker is not able to obtain the session key even if it can access the

server's database. In case the smart card gets stolen or lost and the attacker extracts the information stored inside it, the attacker has the $\{RID_i, b_i, K_i, H_i, ID_{sc}, E_{key}(0)/D_{key}(0), h(0)\}$ information. Due to the random numbers generated by the user and the server, i.e. r_c and r_s, and their influence in the calculation of the session key, $SK = h(A_i \| r_s \| r_c \| IP_i \| b_i)$, the attacker cannot find the session key even if it has the smart card information.

3) Denning-Sacco attack: The attacker cannot access the server's private key parameters or the user's password by having previous session keys. Considering the session key formula in this protocol, i.e. $SK = h(A_i \| r_s \| r_c \| IP_i \| b_i)$, and presence of random numbers r_s and r_c which are selected by the server and the client respectively and are different for each session, it is impossible for the attacker to find the key for new sessions using previous session keys. On the other hand, one can't find the user's password or the server's private key by having the session key since they are not explicitly stated in the session key formula.
4) Replay attack: Imaging the attacker wants to send a duplicate REQUEST$\{EA_i, RID_i, T_2\}$ message to the server. Since equation $|T_3 - T_2| \leq \Delta T$ is verified on the server, the session gets terminated if the message is duplicate. In case the attacker tries to change timestamp T_2 to T_2^* and send the $\{EA_i, RID_i, T_2^*\}$ message to the server, the server calculates parameter A_i^* and calculates the timestamp by decrypting EA_i. By comparing the calculated timestamp to the timestamp T_2^* sent through the public channel, it notices the change to the timestamp. Therefore, the attacker is unable to send duplicate messages or modify the timestamp.

Overview of the avispa software

The Avispa simulation software is a formal security verification tool which is used to show the security or lack of security of a protocol. Avispa provides a language named *HLPSL*[5] for security description of protocols and presenting their security specifications and verifies them as a formal toolkit. Structure and user interface of this tool is presented in the figure. [6]

[5] High Level Protocol Specification Language

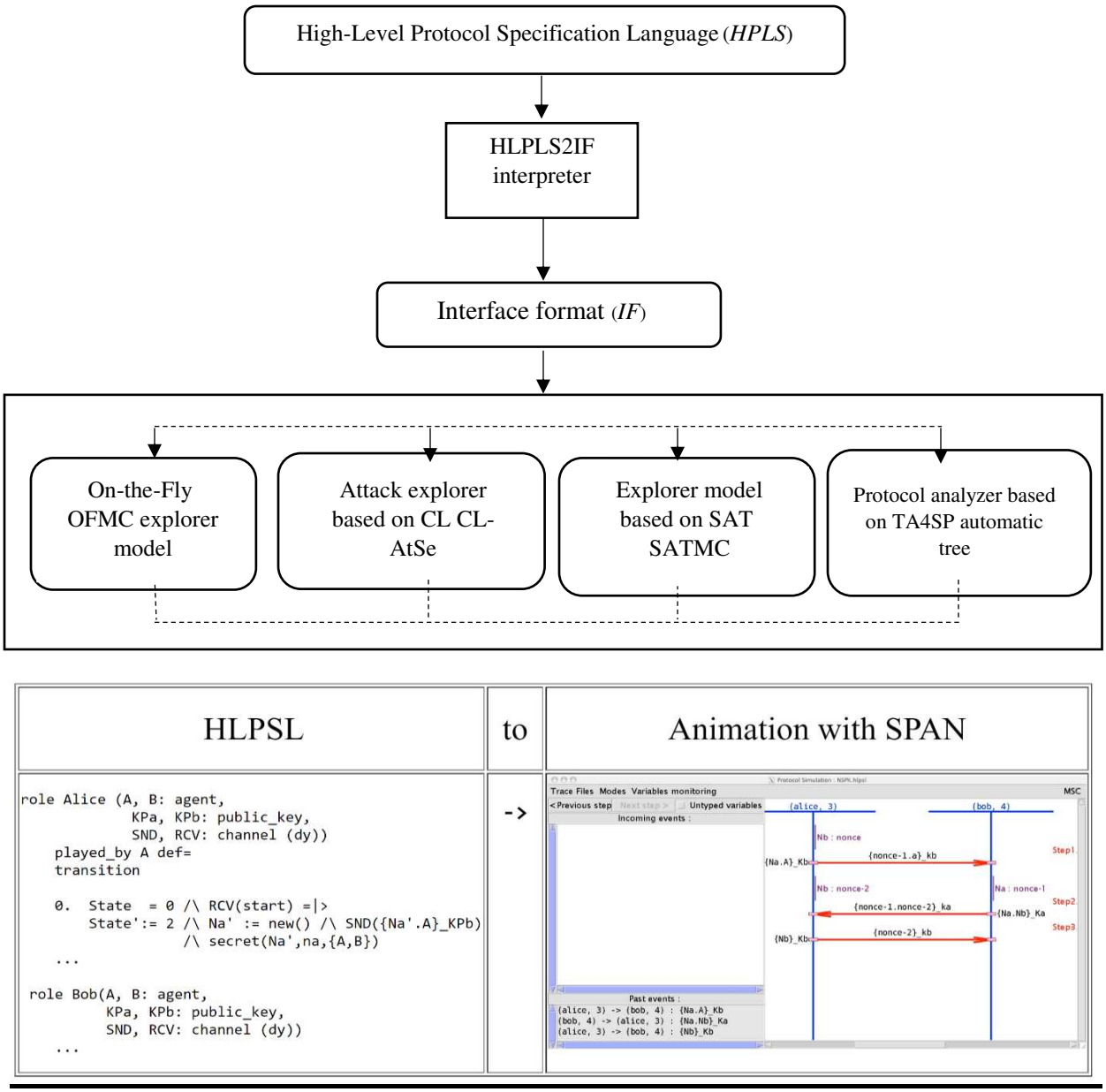


Figure 2 Structure and user interface of Avispa

The flaw of the scheme presented by Nikooghadam et al in 2018

By reviewing the paper presented by Nikooghadam et al. in 2018 we will prove that the Perfect forward secrecy security need will not be satisfied.

The perfect forward secrecy security requirement is defined as: if any of the long terms (such as the user's password or the server's private key) gets leaked, the attacker does not get previous session keys at its disposal. In order to provide this requirement, the session key should not merely depend on the long terms.

Moreover, it should be noted that the attacker has full access to the public channel information. Therefore, the parameters sent through the public channel should not be directly used while creating the session key. As mentioned in the paper's title, this requirement is not provided in the method. This means that if the database on the server is infiltrated, the attacker will access some parameters which make it possible for him to work out the session key and question the Perfect forward secrecy security requirement.

Imagine that the private key x_s, which is a long term, is leaked and the attacker was able to infiltrate the database and obtain parameters b_i and NID_i. Since the attacker has private key x_s and since RID_i was also sent over the public channel, equation $D_{x_s}(RID_i) = (ID_{sc}^* \| ID_i^* \| T_1)$ can be used to obtain ID_i and ID_{sc}. Also, because SID is also public and in fact the server's ID, the attacker has all of the $A_i^* = h(SID_i \| x_s \| ID_{sc} \| ID_i)$ parameters. Therefore, it is able to create A_i. The attacker also has the NID_i and ID_i values and can use $NID_i{}^* = ID_i \oplus IP_i$ to obtain IP_i. Now, it can use $D_{IP_i}(EA_2) = (IP_i{}^*, r_c{}^*, r_s{}^*, T_4)$ to calculate the rest of the values in the session key. This way, we were able to prove that the perfect forward secrecy security requirement is not met.

The proposed solution:

The proposed protocol consists of the following steps: Registration, authentication, and changing the password. We will explain every step at length.

First, the user selects a password, and ID, and three random numbers. Then, in order to prevent the insider attack, the user creates parameter $MPW_i = h(ID_i||PW_i||a_i)$ and carries out the following calculations. Finally, it sends parameters $ID_i, IP_i, b_i, a_i, r_i, pk_{(u)}, TN, MPW_i,$ RB to the server. One of these parameters is its own public key. It is worth noting that the messages exchanged between the user and the server at the registration step are sent through the secure channel.

RB=h($b_i \| r_i$)

$IP_i = h(h(ID_i||MPW_i||\text{RB}) mod\ m)$

TN=h(RB$\|IP_i$)

Immediately after the server receives the user's message, it uses the user ID to create a temporary ID named HID_i using the following operations. It also creates parameter IP_i according to the new temporary ID. Then, the server carries

out the following calculations and finally sends a smart card containing parameters HID_i,F, SID_i, IP_{inew}, (s), and A_i to the user.

MID_i=h ($ID_i \| a_i$)

HID_i=$MID_i \oplus TN$

$IP_{inew} = \square(\square(HID_i || MPW_i || \text{RB}) mod\ m)$

Select ID_{sc}

GF=$E_{pk(u)}(ID_{sc} || bi)$

NID_i=$HID_i \oplus IP_{inew}$

RID_i=h(GF$\|HID_i$)

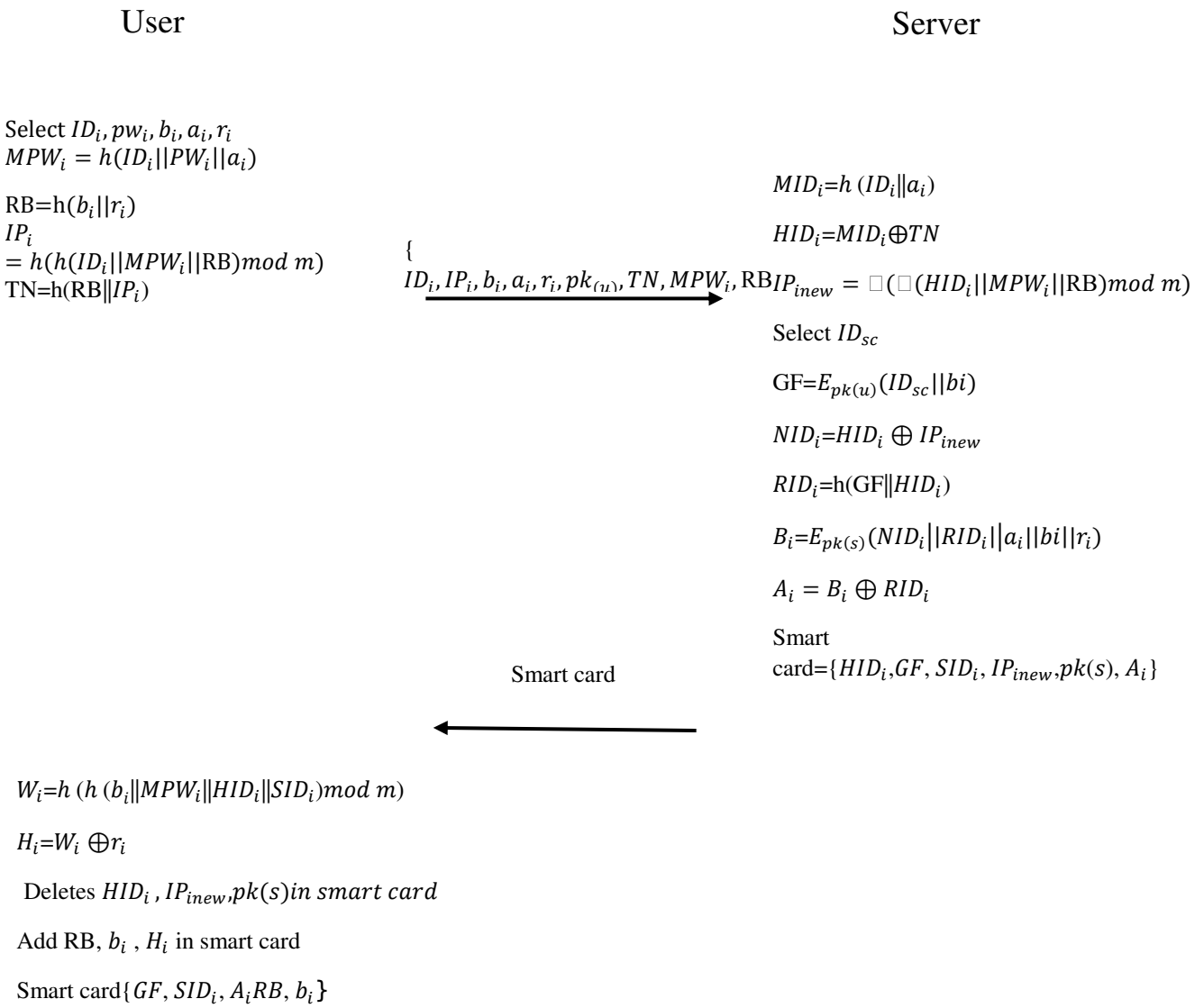
B_i=$E_{pk(s)}(NID_i || RID_i || a_i || bi || r_i)$

$A_i = B_i \oplus RID_i$

Smart card=$\{HID_i, GF, SID_i, IP_{inew}, pk(s), A_i\}$

Upon receiving the smart card, the user will delete some of the parameters in it and add some parameters to it. In the end, the smart card contains parameters GF, SID_i, $A_i RB$, b_i.

Overview of the registration step is presented in the following figure.



Authentication and key agreement step

In the authentication step, first, the user inserts the smart card into the ATM machine and enters its user ID and password. Then, the following calculations are carried out by the smart card and parameters E_1, E_2, T_1 are sent to the user through the public channel.

Inserts smart card

Enters HID_i^*, PW_i^*

$MPW_i^* = h(HID_i^*||PW_i^*||a_i)$

W_i^*=h (h ($b_i‖MPW_i^*‖HID_i^*‖SID_i$)$mod\ m$)

r_i^*=$W_i^* \oplus H_i$

RB^*=h($b_i||r_i^*$)
RB=?RB^* *replace* RB^* *with RB*
$IP_{inew}^* = \square(\square(HID_i^*||MPW_i^*||RB^*) mod\ m)$

$NID_i^* = HID_i^* \oplus IP_{inew}^*$

$RID_i^* = h(GF||HID_i^*)$

B_i^*=$A_i \oplus RID_i^*$

Selects a time stamp T_1 *and a random number* q_i

$D_i = q_i \oplus r_i^*$

$E_1 = E_{PK(s)}(NID_i^*||A_i||RID_i^*||D_i)$

$E_2 = B_i \oplus E_1$

Once the server receives the message, it checks the freshness of the message Check $|T_2 - T_1| \leq \Delta T$. This is done to prevent the replay attack. Then, the server decrypts parameter E_1, which has been encrypted with the server's public key, with its private key and creates $B_i^* = A_i^* \oplus RID_i^*$ and compares it to the B_i' resulting from the encryption of E_1. In fact, this authenticates the received message. Then, since the B_i^* resulting from decryption of E_1 has been encrypted by its own public key, the server decrypts it with its private key and creates the initial session key. Afterward, the server calculates parameter $z_i = E_{pK(u)}(Sk||r_i)$ and finally, it sends T_3, z_i to the other party.

Selects a time stamp T_2

Check $|T_2 - T_1| \leq \Delta T$

Decrypt $E_{1\ sK(s)}$=$(NID_i^*, A_i^*, RID_i^*, D_i^*)$

$B_i^* = A_i^* \oplus RID_i^*$

$B_i' = E_2 \oplus E_1$

$B_i' = ? B_i^*$

Selects a time stamp T_3

B_i=$E_{pk(s)}(NID_i||RID_i||a_i||bi||r_i)$

$r_i = q_i \oplus D_i$

SK=$h(RID_i^*||B_i^*||a_i)$

$z_i = E_{pK(u)}(Sk||r_i)$

Once the other party receives message z_i, it checks the freshness of the message Check $|T_4 - T_3| \leq \Delta T$ to eliminate the possibility of a replay attack. Since z_i has been encrypted with its own public key, the user decrypts it using its private key and obtains the initial session key. Then, it creates parameter D_i and authenticates the received message using equation $D_i^* = D_i$. Since parameter *GF* was encrypted using the user's public key, the user decrypts it with its private key and creates a new session key. Finally, the user calculates the following parameters and also calculates C_i and T_5 and sends them to the other party.

Selects a time stamp T_4

Check $|T_4 - T_3| \leq \Delta T$

Decrypt $Z_{i\,sK(u)}$ (Sk^*, r_i^*)

$D_i = r_i^* \oplus q_i$

$D_i^* = D_i$

Selects a time stamp T_5

Decrypt $GF_{sK(u)}$ (ID_{sc}, b_i)

$SK_{new} = h(RID_i||B_i||a_i||ID_{sc}||q_i)$

$R_i = r_i \oplus b_i$

$C_i = E_{pk(s)}(SK_{new}||R_i)$

The server checks the freshness of the message immediately after receiving it. Then, it decrypts C_i using its private key. Afterward, the server creates parameter R_i and compares it to the R_i resulting from the decryption of C_i. By doing so, the server authenticates the message. At this point, the parties in the connection have agreed on the session key.

Overview of the authentication and key agreement step is presented in the following figure.

User Server

Inserts smart card

Enters HID_i^*, PW_i^*

$MPW_i^* = h(HID_i^* || PW_i^* || a_i)$

W_i^*=$h\ (h\ (b_i \| MPW_i^* \| HID_i^* \| SID_i) mod\ m)$

r_i^*=$W_i^* \oplus H_i$

RB^*=h$(b_i || r_i^*)$

RB=?RB^* *replace* RB^* *with RB*

$IP_{inew}^* = \square(\square(HID_i^* || MPW_i^* || RB^*) mod\ m)$

$NID_i^* = HID_i^* \oplus IP_{inew}^*$

$RID_i^* = h(GF || HID_i^*)$

B_i^*=$A_i \oplus RID_i^*$

Selects a time stamp T_1 *and a random number* q_i

$D_i = q_i \oplus r_i^*$

$E_1 = E_{PK(s)}(NID_i^* || A_i || RID_i^* || D_i)$

$E_2 = B_i \oplus E_1$

E_1, E_2, T_1 →

Selects a time stamp T_2

Check $|T_2 - T_1| \leq \Delta T$

Decrypt $E_{1\ sK(s)}$=$(NID_i^*, A_i^*, RID_i^*, D_i^*)$

$B_i^* = A_i^* \oplus RID_i^*$

$B_i' = E_2 \oplus E_1$

B_i' =? B_i^*

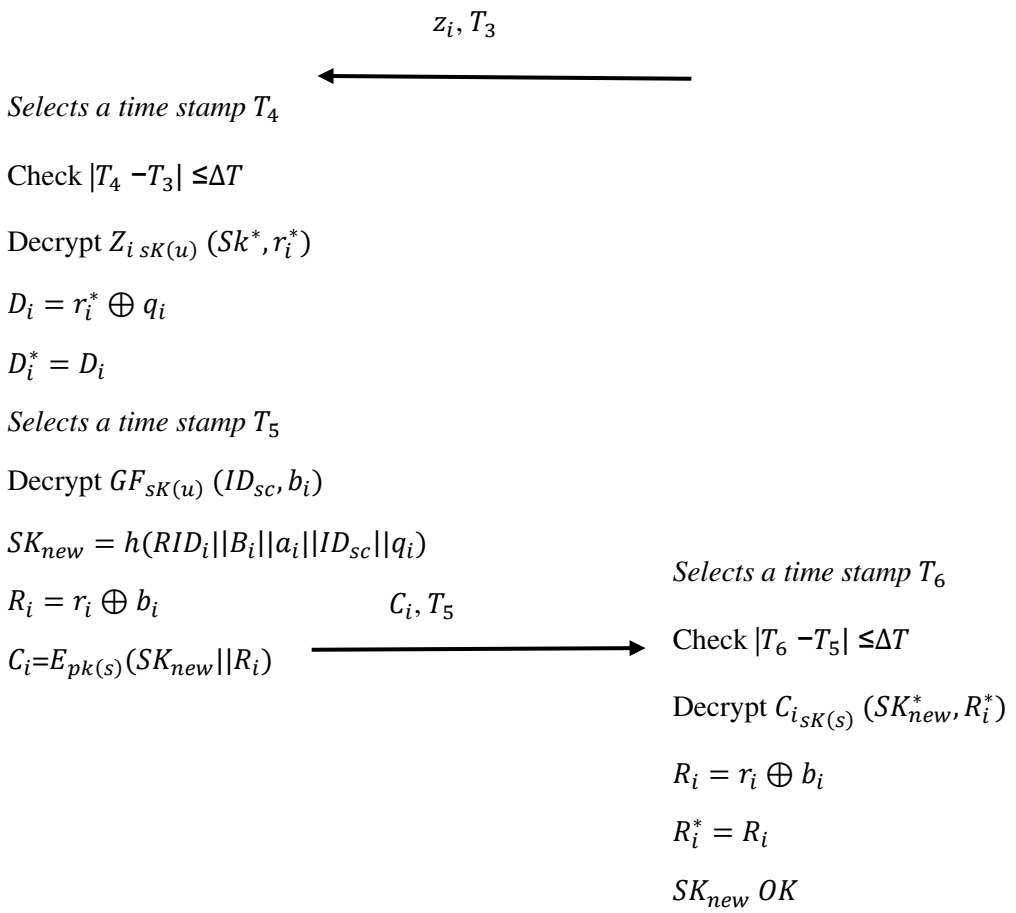
Selects a time stamp T_3

B_i=$E_{pk(s)}(NID_i || RID_i || a_i || bi || r_i)$

$r_i = q_i \oplus D_i$

SK=$h(RID_i^* || B_i^* || a_i)$

$z_i = E_{pK(u)}(Sk || r_i)$



Password changing step

First, the user inserts its smart card to the device and enters its old password, i.e. HID_i^* and PW_i^*. Then, it submits the request to change the password along with the new password.

$MPW_i^* = h(HID_i^*||PW_i^*||a_i)$

$W_i^* = h\,(h\,(b_i||MPW_i^*||HID_i^*||SID_i) mod\ m)$

$r_i^* = W_i^* \oplus H_i$

$RB^* = \mathrm{h}(b_i||r_i^*)$

RB=?RB^*

If the above equation stands, the smart card belongs to the person using it and it has not been stolen. Now, if we imagine the new password to be PW_i^{**}, the

following parameters will be calculated again and finally, RB^{**} will replace RB^{*} on the smart card.

$MPW_i^{**} = h(HID_i^{*}||PW_i^{**}||a_i)$

$W_i^{**}=h\,(h\,(b_i\|MPW_i^{**}\|HID_i^{*}\|SID_i)mod\ m)$

$r_i^{**}=W_i^{**} \oplus H_i$

$RB^{**}=\mathrm{h}(b_i||r_i^{**})$

Security analysis of the proposed method

Anonymity: Since in the registration step, the user sent its ID to the server and the server used it to create a temporary ID which was used from that point on, the attacker cannot obtain the primary ID and anonymity requirement is satisfied.

Perfect forward secrecy: Since the private keys of both parties are used to create the steps of creating the session key, the attacker cannot obtain the mutual session key and this security requirement is satisfied.

Tracking: Once again, since the server creates temporary ID HID_i^{*} and sends it to the user at the registration step which is used at all of the steps, the attacker cannot identify the user by listening to the public channel at the authentication step. Therefore, it will not be able to track the user.

Replay attack: At every step of the authentication process where messages are sent through the public channel, timestamps are used. Once a message is received, its freshness is verified. therefore, the attacker cannot take a message and simply resend it.

Impersonation attack: Since the messages are authenticated at every step of their transmission, as presented in the protocol, the attacker cannot impersonate the user or the server for the other party.

Temporary parameter leakage attack: In this attack, it is assumed that if all of the random session parameters get leaked, the attacker should still be unable to work out the session key. Since parameters $RID_i^{*}\ and\ B_i^{*}$ are involved in the creation of the session key and it does not merely depend on the random parameters, the session key is not vulnerable to the temporary parameter leakage attack.

Insider attack: Since the user sends its password to the other party as $MPW_i = h(ID_i||PW_i||a_i)$ and not directly, insider attack is not possible.

Denning-Sacco attack: This attack expresses that if the attacker obtains session key, it should not be able to use the session key parameters and the parameters sent over the public channel to obtain the user's password. Because of the $MPW_i^* = h(HID_i^* || PW_i^* || a_i)$ structure, this type of attack is impossible for the attacker to carry out.

Analysis of the proposed method using the formal Scyther tool

In this section, the resistance of the proposed method against some of the famous attacks was reviewed. In the following sections, the analysis of the proposed method's correctness using the Scyther tool will also be presented.

Scyther results : verify

Claim				Status	Comments
Drmohajer	U	Drmohajer,U1	Secret ai	Ok	No attacks within bounds.
		Drmohajer,U2	Secret IDsc	Ok	No attacks within bounds.
		Drmohajer,U3	Alive	Ok	No attacks within bounds.
		Drmohajer,U4	Nisynch	Ok	No attacks within bounds.
		Drmohajer,U5	Niagree	Ok	No attacks within bounds.
		Drmohajer,U6	Weakagree	Ok	No attacks within bounds.
	S	Drmohajer,S1	Secret ai	Ok	No attacks within bounds.
		Drmohajer,S2	Secret IDsc	Ok	No attacks within bounds.
		Drmohajer,S3	Alive	Ok	No attacks within bounds.
		Drmohajer,S4	Nisynch	Ok	No attacks within bounds.
		Drmohajer,S5	Niagree	Ok	No attacks within bounds.
		Drmohajer,S6	Weakagree	Ok	No attacks within bounds.

Figure 3 Analysis of the proposed method using the formal Scyther

Conclusion

In the VoIP technology, two types of protocols are used. These protocols are as follows: session initiation protocol (SIP) and real-time transfer protocol.

Considering the importance of correct authentication and key agreement in the SIP field and the connections among two users or more, several protocols have been presented in this field. The recently presented protocols for correct authentication are often in one of the following categories: 1-password-based 2-smart card and password-based. The security protocols in SIP must satisfy security requirements including mutual authentication, session key security, forward secrecy, known key secrecy. They also need to be resistant to various attacks including stolen verifier, password guessing, Denning-Sacco, modification, man in the middle, insider, server spoofing, and temporary session parameters attacks.

As mentioned in the recent work review section, the presented one-factor and two-factor methods have usually been unable to fully satisfy the security aspects and utilize suitable measures to prevent the attacks from happening. Therefore, after a while since presenting the new method, we addressed the security flaws of the proposed method and presented an improved method. Since in addition to the above items, lower execution and connection time are of importance in the communications, the presented protocols reviewed this important factor as well.

In this paper, the method presented by Nikooghadam et al. and the one presented by Zhang were studies comprehensively and all of their flaws were reviewed in length. The outcome was that these papers were unable to prevent replay and temporary parameter leakage attacks. Also, they were unable to provide some security requirements such as anonymity, fast error detection, and some others. In this regard, Zhang et al. solved these problems in a new method but Nikooghadam et al. proved that Zhang was unable to satisfy all of the security requirements. Therefore, in 2018 they presented a new scheme in order to provide every security requirement. However, we proved in this paper that Nikooghadam et al. at 2018 were unable to provide some of the security requirements with their idea. Therefore, some attacks were inflicted on it such as impersonation, denial of service, etc.

Results of informal analysis of the proposed method demonstrated its resistance to various attacks described in SIP. Moreover, security requirements including anonymity, fast error detection, and re-registration are satisfied in the presented method. However, it was informally proven in this paper that these results are faulty and the method cannot satisfy all of the security requirements.

Since the proposed method does not require costly operation such as multiplication on an oval curve, and exponentiation and using symmetrical encryption methods and hash functions, the proposed method is considered a light method compared to other methods. The proposed method has also improved with regard to computational cost and number of transferred messages on the public channel compared to previous methods. Although the connection cost of some presented protocols is lower than the connection cost of the proposed method, none of them are able to provide all of the security aspects. It is worth noting that providing security is far more important than computational and executive costs.

Acknowledgements

The authors thank Dr. Mohajerzadeh for his help in publishing this article.

Authors 'contributions

Authors MN and KS formulated and implemented the methodology and drafted the manuscript. AM adjusted the initial methodology and supervised the work.

Funding

This article was funded by the authors.

Availability of data and materials

Not applicable

Competing interests

The authors declare that there are no competing interests.

Author details

1 Ferdowsi University Main Campus, Mashhad state, Islamic Republic of Iran

2 Engineering Faculty, Ferdowsi University Main Campus, Mashhad state, Islamic Republic of Iran

References

[1] Zhang, Z., Qi, Q., Kumar, N., Chilamkurti, N. and Jeong, H.Y., 2015. A secure authentication scheme with anonymity for session initiation protocol using elliptic curve cryptography. *Multimedia Tools and Applications*

[2] Lu, Y., Li, L., Peng, H. and Yang, Y., 2016. A secure and efficient mutual authentication scheme for session initiation protocol. *Peer-to-Peer Networking and Applications*

[3] Arshad, H. and Nikooghadam, M., 2016. An efficient and secure authentication and key agreement scheme for session initiation protocol using ECC. *Multimedia Tools and Applications.*

[4] Lin, H., Wen, F. and Du, C., 2016. An anonymous and secure authentication and key agreement scheme for session initiation protocol. *Multimedia Tools and Applications.*

[5] Zhang, Liping, Shanyu Tang, and Shaohui Zhu. "An energy efficient authenticated key agreement protocol for SIP-based green VoIP networks." *Journal of Network and Computer Applications.*

[6]*https://www.researchgate.net/publication/228356197_A_security_protocol_animator_tool_for_AVISPA*

Figures

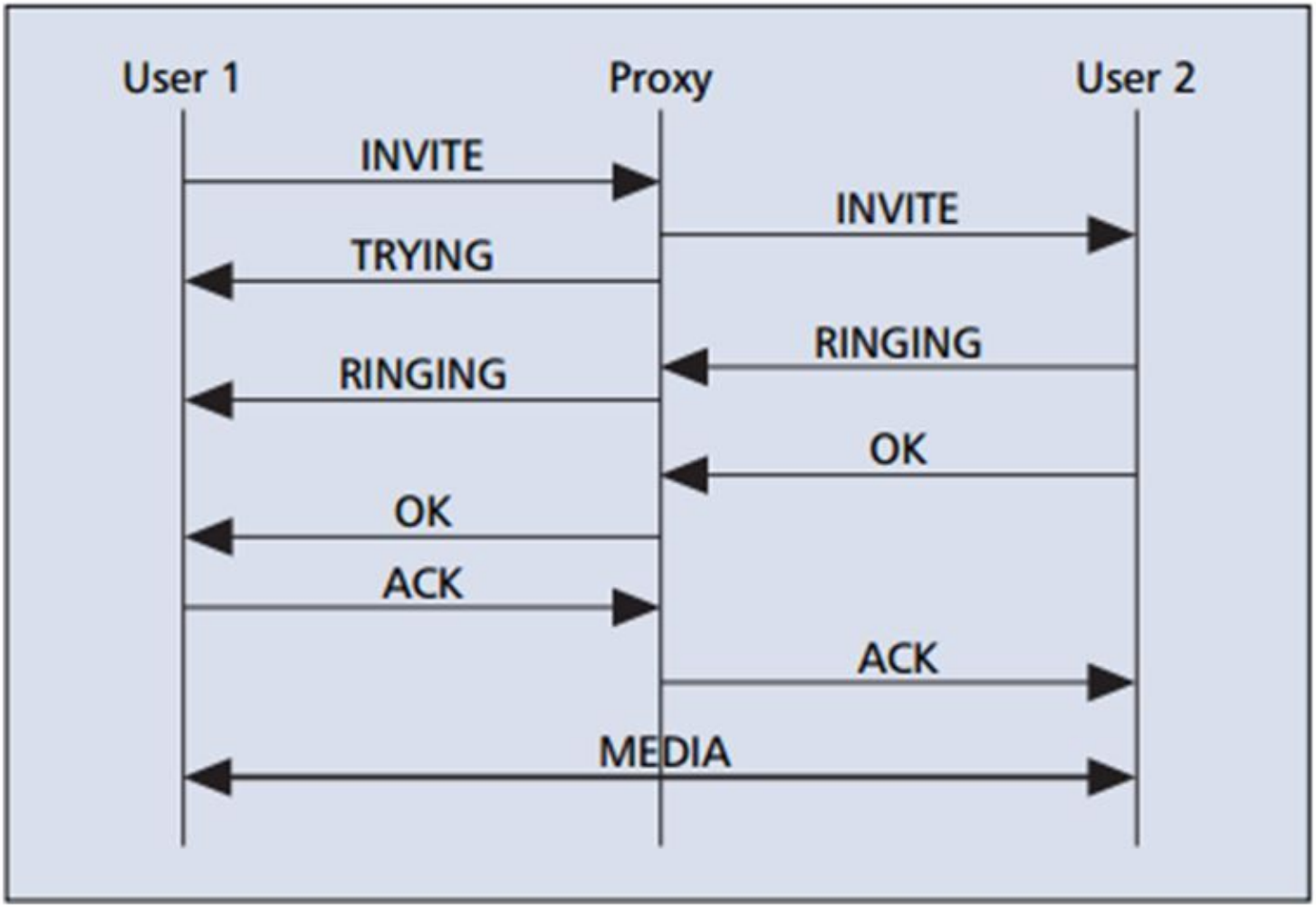


Figure 1

operation of SIP in the VoIP telephone system

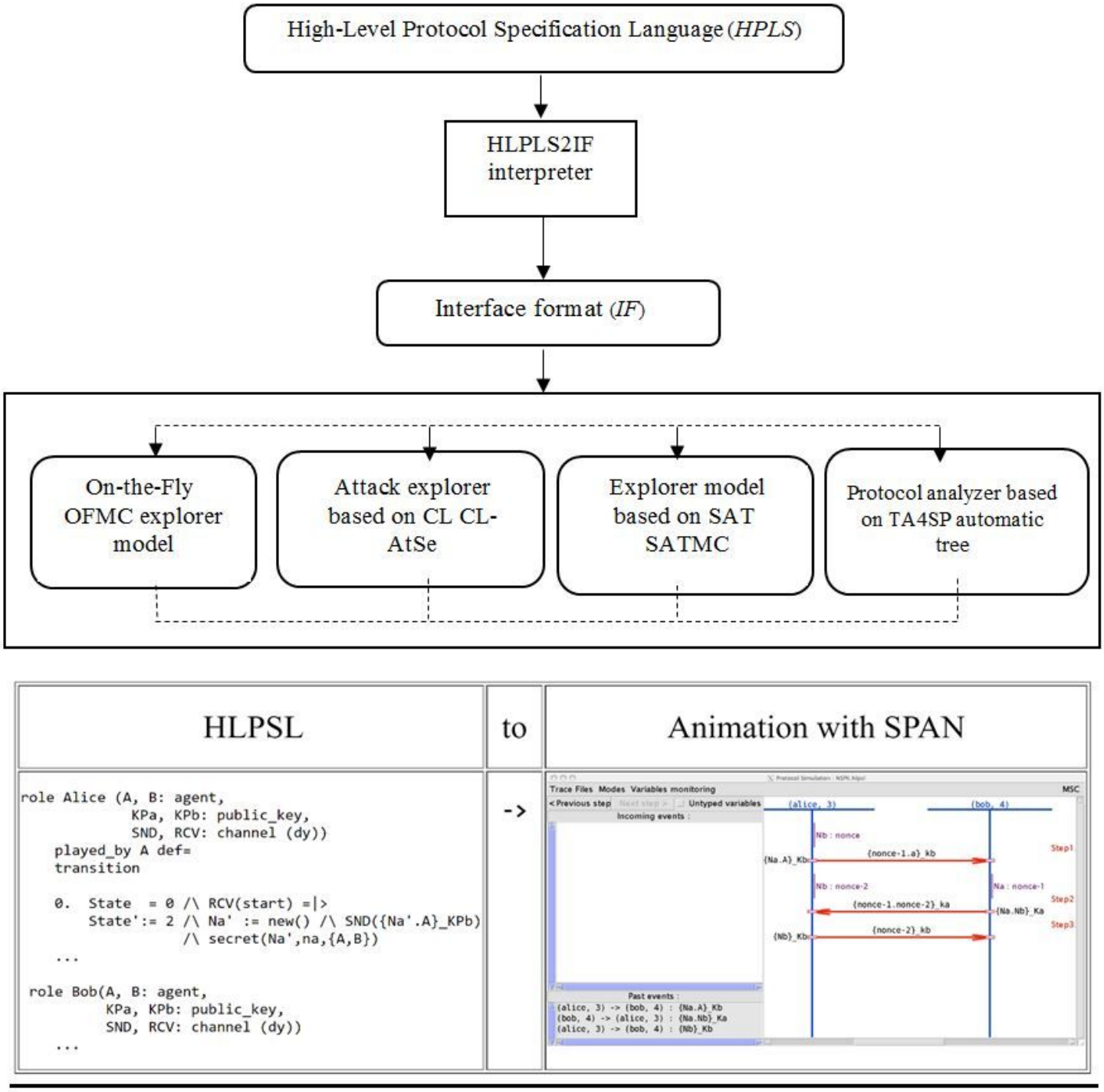


Figure 2

Structure and user interface of Avispa

Claim				Status	Comments
Drmohajer	U	Drmohajer,U1	Secret ai	Ok	No attacks within bounds.
		Drmohajer,U2	Secret IDsc	Ok	No attacks within bounds.
		Drmohajer,U3	Alive	Ok	No attacks within bounds.
		Drmohajer,U4	Nisynch	Ok	No attacks within bounds.
		Drmohajer,U5	Niagree	Ok	No attacks within bounds.
		Drmohajer,U6	Weakagree	Ok	No attacks within bounds.
	S	Drmohajer,S1	Secret ai	Ok	No attacks within bounds.
		Drmohajer,S2	Secret IDsc	Ok	No attacks within bounds.
		Drmohajer,S3	Alive	Ok	No attacks within bounds.
		Drmohajer,S4	Nisynch	Ok	No attacks within bounds.
		Drmohajer,S5	Niagree	Ok	No attacks within bounds.
		Drmohajer,S6	Weakagree	Ok	No attacks within bounds.

Figure 3

Analysis of the proposed method using the formal Scyther