Practical and Provably Secure Three-Factor Authentication Protocol Based on Extended Chaotic-Maps for Mobile Lightweight Devices

Shuming Qiu, Ding Wang, Guoai Xu, and Saru Kumari

Abstract—Due to the limitations of symmetric-key techniques, authentication and key agreement (AKA) protocols based on public-key techniques have attracted much attention, providing secure access and communication mechanism for various application environments. Among these public-key techniques used for AKA protocols, chaotic-map is more effective than scalar multiplication and modular exponentiation, and it offers a list of desirable cryptographic properties such as un-predictability, un-repeatability, un-certainty, and higher efficiency than scalar multiplication and modular exponentiation. Furthermore, it is usually believed that three-factor AKA protocols can achieve a higher security level than single- and two-factor protocols. However, none of existing three-factor AKA protocols can meet all security requirements. One of the most prevalent problems is how to balance security and usability, and particularly how to achieve truly three-factor security while providing password change friendliness. To deal with this problem, in this article we put forward a provably secure three-factor AKA protocol based on extended chaotic-maps for mobile lightweight devices, by adopting the techniques of “Fuzzy-Verifiers” and “Honeywords”. We prove the security of the proposed protocol in the random oracle model, assuming the intractability of extended chaotic-maps Computational Diffie-Hellman problem. We also simulate the protocol by using the AVISPA tool. The security analysis and simulation results show that our protocol can meet all 13 evaluation criteria regarding security. We also assess the performance of our protocol by comparing with seven other related protocols. The evaluation results demonstrate that our protocol offers better balance between security and usability over state-of-the-art ones.

Index Terms—Extended chaotic-maps, three-factor, authentication and key agreement, guessing attack, perfect forward secrecy

1 INTRODUCTION

With the booming of various sensitive applications over the Internet, user privacy is becoming more and more important, and has attracted widespread concern. For example, in telecare medicine scenario, physicians are inclined to be assisted with health care systems to check the patient’s medical records and diagnose the diseases. In this system, the communication among the physicians, patients and the servers is carried out through an insecure open channel, it is convenient for the intruder to carry out malicious attacks to forge the patients’ records, which is fatal for the patient. Furthermore, with the rapid development of mobile application technologies, affordable and portable mobile lightweight devices are becoming very popular. Mobile lightweight devices (e.g., laptops, personal digital assistants, smartphone, smartwatch, and wearable devices) are able to access cloud servers for online payment, online voice and video chatting, mobile banking interaction, e-commerce, and so on at any time and anywhere. It has been estimated that by 2020, the number of mobile users worldwide will increase to 7.33 billion, 7.8 percent higher than the 6.8 billion users in 2019 [1]. This means that on average everyone in the world has one mobile device. However, due to the openness of wireless network communication, authentication and anonymity issues in mobile environments must be concerned. Otherwise, the security and privacy of the user data will be difficult to guarantee.

The network communication between mobile users and cloud servers may suffer from various attacks, such as impersonation attack and password guessing attack. Moreover, mobile devices are usually resource constrained and vulnerable to special network attacks. Therefore, it is indispensable to establish an authentication and key agreement (AKA) protocol to protect the conversations between the users with lightweight mobile devices and remote servers in various application environments. However, many protocols sacrifice security in order to achieve usability. For example, the schemes in [2], [3] and [4] can’t resist key-compromised user impersonation attack, off-line password guessing attack and can’t provide three-factor security. The reason for these defects lies in that, on the one hand,
their protocols do not follow the design principle of authentication protocol proposed by Ma et al. [5] to achieve robust security, on the other hand, the designers do not employ the state-of-the-art technical means to improve usability. Accordingly, designing an AKA protocol that can balance security and usability remains a challenging problem.

Since the first password authentication and key agreement protocol [6] was presented in 1981, hundreds of AKA protocols have been proposed for Client-Server [7], [8], [9], Multi-Server environment [10], Radio Frequency Identification (RFID) systems [11], etc. For example, Durlanik and Sogukpinar presented an AKA protocol for session initiation protocol (SIP) using elliptic curve cryptography (ECC) [8]. In 2014, Wang-Wang [12] provided the attack model, the design principle and solutions for two-factor AKA protocols for wireless sensor networks. In 2018, Wang et al. [9] analyzed the challenges in designing identity-based privacy-preserving AKA protocols for mobile devices.

In order to improve the computational efficiency of the protocol in the mobile lightweight equipment and reduce the communication cost of the protocol, initially scholars tried to design a practical AKA protocol using symmetric cryptography and hash functions (e.g., [11], [13], [14]). However, it is found that these protocols can not provide forward secrecy, and the public-key cryptography technology is indispensable to achieve forward secrecy according to Ma et al. [5]. Since then, the public key cryptography technology is becoming to be widely used in AKA protocols. At present, many public key cryptography techniques can be utilized to design AKA protocols, for instance, ECC, RSA, bilinear pairings and so on. Notably, another technique, i.e., chaotic-maps [15] (e.g., Logistic maps and Chebyshev-maps), has been intensively studied by scholars. They refer to the random irregular movement in the deterministic system, whose behaviour is uncertain, unrepeatable and unpredictable. Especially, chaotic maps have been widely used in cryptography, such as using chaotic maps to design encryption systems and key agreement protocols.

In 2005, Xiao et al. for the first time presented a deniable AKA protocol for E-Commerce by using chaotic-maps [16]. Later, Alvarez showed that Xiao et al.’s protocol suffers the security and efficiency limitations, for example, they pointed out that this scheme cannot resist counterfeiting attacks [17]. Later, Xiao et al. [18] improved Xiao et al.’s protocol [16] and presented an unconventional AKA protocol based on chaotic-maps. Since then, a large number of chaotic-maps based AKA protocols [19], [20], [21], [22], [23], [24] have been proposed for diverse application scenarios.

The above AKA protocols are mainly based on a single authentication factor (i.e., password) or two authentication factors (i.e., password + smart card). Being faced with demanding security requirements and considering the popularity of mobile devices that supports biometric operations, traditional AKA protocols are integrated with the third authentication factor (i.e., biometric) to achieve higher security [20], [25], [26], [27], [28]. Biometrics belong to "something the user is" and are the invariable physiological characteristics (such as fingerprint, face and iris) that people have. It is usually called three-factor AKA protocol when it is used together with password and smart card. Before the password is verified, the user is required to provide her biological information. Compared with single-factor AKA protocols and two-factor AKA protocols, three-factor AKA protocols are considered to have certain advantages in security. Recently, AKA protocols based on three-factor have become a hot research topic. However, in many three-factor AKA protocols [2], [3], [4], [22], [29], three-factor security and other security goals actually cannot be fulfilled. Therefore, to make better usage of the advantage of biometrics, it is very important to construct an AKA protocol for mobile environments that can provide truly three-factor security while maintaining desirable usability properties.

1.1 Related Work

In 2013, Guo and Chang [26] put forward a two-factor AKA protocol by employing the chaos theory. But in 2015, Lin [30] found that Guo and Chang’s protocol can not provide anonymity and session-key security. Further, Lin presented an improved protocol to address these limitations [30]. After two years, Lee et al. [20] argued that Lin’s protocol can not achieve the security requirements as claimed: Lin’s protocol is vulnerable to DoS attacks, internal enemy attacks and cannot ensure secure key agreement. To eliminate the defects of Lin’s protocol, Lee et al. put forward an enhanced two-factor AKA protocol while employing extended chaotic-maps that is the extended version of chaotic-map in interval (−∞, +∞). Lee et al. [20] argued that their solution is secured against all known attacks. But, we find that Lee et al.’s protocol cannot resist the temporary information leakage attack, key compromise impersonation attack and does not provide important usability properties such as local password update (i.e., the criterion C2 in [31]) and timely false password detection (i.e., the criterion C9 in [31]).

In 2014, Islam [32] designed a three-factor AKA protocol based on extended chaotic-mapping and remarked that his protocol provides security against various known attacks including smart card loss attacks and upholds the three-factor security characteristics. Despite of such claims, in 2016, Jiang et al. [23] revealed that Islam’s proposal can not provide false password timely inspection functions and is prone to off-line password guessing attack, smart card loss attack, and biological sample leaks. To cope the mentioned limitations, Jiang et al. [23] presented a new three-factor AKA protocol by using the fuzzy verification method and proved that the scheme does not only resist all kinds of attacks but also holds essential security features. Nevertheless, we remark that Jiang et al.’s proposal is prone to key compromise user impersonation attack and clock-synchronization attack.

In 2015, on the basis of Lee’s work [33], Liu et al. [21] analyzed the AKA protocol using chaotic-maps that are presented by Guo et al. [34]. The authors revealed various security limitations of Guo et al.’s protocol: it cannot provide mutual authentication and is vulnerable to denial of service (DoS) attack, replay attack and guessing attack. Later, Liu et al. [21] put forward an improved two-party AKA protocol. Liu et al. remarked that their presented protocol could resist all known attacks. On the contrary, in 2016, Chen et al. [35] pointed out that the proposal of Liu et al. [21] is unable to resist the off-line password-guessing-attack, failing to achieve truly three-factor security.
In 2017, Wazid et al. [2] proposed a three-factor user authentication protocol for renewable-energy-based smart grid environment. We observed that, this protocol only uses the lightweight cryptographic computations to improve efficiency and the verifier parameter $CT_i$ can be used to perform off-line password guessing. Therefore, this protocol can neither resist off-line password guessing attack nor provide perfect forward secrecy and three-factor security. In 2018, Roy et al. [3] designed a chaotic map-based anonymous authentication protocol with fuzzy extractor for crowdsourcing Internet of Things whose verifier parameter $f_i$ is also vulnerable to be performed off-line password guessing. In the same year, Islam et al. [4] also proposed a provably secure three-factor protocol for multimedia big data communications. Similar defects make the protocol of Islam et al. [4] vulnerable to off-line guessing attack and key compromised user impersonation attack and unable to provide functionality and security features such as three-factor security.

In 2018, Chatterjee et al. [10] used the Chebyshev chaotic map, cryptographic hash function and symmetric key encryption/decryption to construct a three-factor AKA protocol for multi-server environments. Nevertheless, it can be observed that their protocol cannot achieve truly three-factor security: the explicit password verifier $A_i$ enables guessing attackers to off-line guess passwords and identities when the other two factors (i.e., biometric and smart card) are leaked. Note that, truly three-factor security is the most essential goal of a three-factor AKA protocol.

### 1.2 Motivations and Contribution

To be practical, a three-factor AKA protocol based on chaotic-maps should be able to provide comprehensive security goals and various desirable properties. Most essentially, a three-factor AKA protocol must satisfy at least five security goals: (1) Truly three-factor security, that is, if an attacker gains any two of the three authentication factors (i.e., password, smart card and biometric), the attacker can not successfully figure out the third factor; (2) Anonymity and un-traceability, including identity protection and user untraceability; (3) Resistance against password guessing attacks, including off-line guessing attacks and on-line guessing attacks; (4) Session-key security, that is, the attacker can not steal or calculate the session key negotiated between the user and the server, which includes that the protocol should provide perfect forward security, and can resist session-specific temporary information leakage attacks; (5) Resistance to impersonation attacks, including user impersonation attacks, server impersonation attacks, and key-compromise impersonation attacks.

However, existing state-of-the-art three-factor AKA protocols can not satisfy at least one of the above five security goals. For example, the protocols in [2] and [29] can not provide truly three-factor security, the protocol in [21] can not provide user un-traceability, the protocols in [29] and [3] can not resist off-line guessing attacks and session-key security, and the protocol in [22] can not resist counterfeiting attacks. Shin and Kobara [36] pointed out that any passive/active attacker can find out the client’s password and the static key by utilizing off-line guessing attacks against the IEEE 1363.2 Standard [37] and its multi-server system [38].

Another defect of existing state-of-the-art three-factor AKA protocols is that most of them assume that passwords follow the uniform distribution when formulating the advantages of the attacker. However, Wang et al. [31, 39] have proved that user-chosen passwords follow the Zipf’s law, a vastly different distribution from the widely used (but unrealistic) uniform distribution. It illustrates that an AKA protocol with passwords of the Zipf’s distribution is inherently difficult to resist both online and offline password guessing attacks, and a proper security formulation for the advantages of the attacker shall be used. To solve the common security problems in existing three-factor AKA protocols, we put forward a secure three-factor AKA protocol by combining biometric and chaotic-maps for mobile lightweight devices to secure data communications.

In all, this paper makes the following key contributions:

1. In order to resist password guessing attacks, existing three-factor AKA protocols generally adopt two ways. The first way is that in the login phase, the protocol does not provide the usability property of timely password typo detection. The other way is that password verification parameters are stored in smart cards. Following the first way, the protocol can avoid the password guessing attacks caused by the leakage of the password verifier, but it brings additional computational burden and the risk of denial of service attacks to the server. Most protocols follows the second way, but they cannot effectively resist password guessing attacks once the password verifier is leaked. Fortunately, in this paper, we use the technique of “Fuzzy-Verifiers” [39] to construct a fuzzy password verifier, and then we combine the notion of “Honeymords” [39] to design a robust three-factor AKA protocol based on extended chaotic-maps and biometrics, which can effectively resist password guessing attack.

2. We prove that the proposed protocol is semantically secure under the random oracle model based on chaotic-maps computational Diffie-Hellman problem, demonstrate how the protocol satisfies the 13 security evaluation criteria and simulate the protocol by making use of the AVISPA automated tool.

3. We perform a comparative analysis of the security, computation and communication cost of the proposed protocol with seven state-of-the-art related protocols, e.g., Wazid et al. [2] (IEEE Trans. II’17), Roy et al. [3] (IEEE IoT’18) and Islam et al. [4] (IEEE IoT’18). Comparison results demonstrate that the proposed protocol for mobile lightweight devices can achieve a better balance between security and usability.

### 1.3 Roadmap of This Paper

The roadmap of this paper is as follows: Section 2 introduces necessary preliminaries, including the adversary model and the 13 evaluation criteria for three-factor AKA protocols. In Section 3, we present our new protocol. The formal security proof and heuristic security arguments are presented in Sections 4.1 and 4.2, respectively. Section 5 summarizes the comparison of security, communication and computation cost. Section 6 concludes the paper.
Given a biometric input $BIO$, the fuzzy extractor extracts a nearly random string $\sigma$ in an error-tolerant manner. If $BIO'$ can be considered as the change, but closely related to $BIO$, then the extracted $\sigma$ remains the same due to auxiliary string $t$. A fuzzy extractor comprises the following two phases [23], [43]: (1). $GEN(BIO) = (\sigma, t)$. $GEN$ refers to a probabilistic generator that outputs an auxiliary string $t$ and an extracted string $\sigma$, when given a biometric input $BIO$. (2). $REP(BIO', t) = \sigma$. If $BIO'$ is reasonably close to $BIO$, then $REP$ refers to the deterministic reproduction procedure that allows one to recover $\sigma$ from the corresponding auxiliary string $t$ and $BIO'$. 

### Table 1: Notations and abbreviations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>Server</td>
</tr>
<tr>
<td>$U_i$</td>
<td>User</td>
</tr>
<tr>
<td>$ID_i, PW_i$</td>
<td>Identity, password of $U_i$</td>
</tr>
<tr>
<td>$p$</td>
<td>Large prime</td>
</tr>
<tr>
<td>$T_s(x)$</td>
<td>Public key of $S$</td>
</tr>
<tr>
<td>$v$</td>
<td>Random number of $S$</td>
</tr>
<tr>
<td>$u$</td>
<td>Random numbers of $U_i$</td>
</tr>
<tr>
<td>$H(\cdot)$, $H_0(\cdot)$</td>
<td>One-way hash-function</td>
</tr>
<tr>
<td>$SK$</td>
<td>Session key between $U_i$ and $S$</td>
</tr>
<tr>
<td>$G(t)$</td>
<td>Biometric key extraction algorithm</td>
</tr>
<tr>
<td>$REP(\cdot)$</td>
<td>Biometric key reproduction algorithm</td>
</tr>
</tbody>
</table>

### 2.2 Fuzzy Extractor of Biometrics

Given a biometric input $BIO$, the fuzzy extractor extracts a nearly random string $\sigma$ in an error-tolerant manner. If $BIO'$
TABLE 2
Security Evaluation Criteria for Three-Factor AKA Protocols (Adapted From [39], [53])

<table>
<thead>
<tr>
<th>Notation</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC1</td>
<td>User Anonymity and Un-Traceability</td>
</tr>
<tr>
<td>EC2</td>
<td>Password-Verifier-Table is unwanted</td>
</tr>
<tr>
<td>EC3</td>
<td>Password and Biometric Friendliness</td>
</tr>
<tr>
<td>EC4</td>
<td>Password Exposure is avoidable</td>
</tr>
<tr>
<td>EC5</td>
<td>Timely Typo Detection</td>
</tr>
<tr>
<td>EC6</td>
<td>No smart-card Loss Attack</td>
</tr>
<tr>
<td>EC7</td>
<td>Resistance to Known Attacks</td>
</tr>
<tr>
<td>EC8</td>
<td>Key-Agreement provision</td>
</tr>
<tr>
<td>EC9</td>
<td>Mutual-Authentication verification</td>
</tr>
<tr>
<td>EC10</td>
<td>No Clock Synchronization</td>
</tr>
<tr>
<td>EC11</td>
<td>Sound-Repairability</td>
</tr>
<tr>
<td>EC12</td>
<td>Three-Factor Security</td>
</tr>
<tr>
<td>EC13</td>
<td>Perfect Forward Secrecy</td>
</tr>
</tbody>
</table>

objectives of AKA protocols are different in literature [27], [39], [44], [52], [53], [54]. For example, HMQV [52] pointed out that key-compromised user impersonation attacks should be considered.

Wang et al. [39] proposed 12 security indicators for two-factor AKA protocols, and Wang et al. [53] summarized the security indicators for AKA protocols in wireless sensor networks (WSN). Table 2 (adapted from [39] and [53]) refers to the security indicators proposed by these protocols [27], [39], [44], [52], [53], [54] to cover the security requirements of single-factor, two-factor ([14], [28]) and three-factor AKA protocols. Nevertheless, when analyzing specific AKA protocols, we should treat them differently according to the number of factors, the characteristics of cryptography and the application environment. For example, when analyzing the security of a single-factor AKA protocol or a double-factor AKA protocol, we do not consider three-factor security, which is the inherent security characteristic of three-factor AKA protocols.

It should be noted that in three-factor AKA protocols, password and biometric friendliness (EC3) means that users are allowed to choose passwords and biometric features freely and update these factors efficiently and locally. No smart-card loss attack (EC6) means that the protocol should be able to resist the smart card loss attack, that is, anyone who obtains a legitimate user’s smart card should not be able to change or restore the password through online, offline or hybrid guessing techniques within polynomial time, or imitate the user to login and execute the protocol, even if the user’s biometrics is compromised. It should be noted that resistance to known attacks (EC5) should include session-specific temporary information attacks and (key-compromised) user impersonation attacks [52]. Moreover, EC7 also includes off/on-line password guessing attack, privileged insider attack, de-synchronization attack, replay attack, server impersonation attack and man-in-the-middle attack, etc. Three-factor security (EC12) means that for any three-factor AKA protocol, even if any two of the three factors are leaked, but not all, it should resist all known attacks. Undetectable online guessing attack refers to cases where the server can find out who the communicating party is. In order to resist detectable online guessing attacks, it is recommended to set a limit on the times of user login request message validation errors (preset threshold value) on the server side. When analyzing online guessing attacks, we assume that the server is legal and reliable, acting as a password verification oracle. It should be noted that the online guessing attack in this paper are the detectable online guessing attack.

3 PROPOSED PROTOCOL

To meet security needs of AKA protocol for mobile users in Client-Server environment, in our protocol we adopt at least the following six approaches:

1) In order to prevent attacks from privileged insiders, in the registration phase, we only pass IDi to the server, construct password login verifiers by using the fuzzy verification technology [39], and produce random numbers of biometrics by using the fuzzy extractor [43].

2) To resist password guessing attack, on the one hand, we use “Fuzzy-Verifiers” to provide the resilience to offline password guessing attack [39] and to ensure timely password typo detection, efficient password update and three-factor security, where $W_i = H((H(ID_i) \oplus VP_{Wi}) \mod n_0)$ is the key parameter. On the other hand, we set “Honeynets” [39] to resist online password guessing attack.

3) According to [25], to ensure perfect forward secrecy and efficiency, we employ Chaotic-maps (from the public key cryptosystem domain).

4) In order to provide security against key compromise user impersonation attacks, we propose storing a secret parameter $r_i$ securely at the server side (e.g., stored in an auxiliary server as with [9]).

5) We present a dynamic identity approach by employing a public key algorithm to provide user un-traceability, i.e., $C_3 = (ID_i||B_0) \oplus H_0(C_2)$.

6) The session key will be computed as $SK = H((1||CID_i||B_0||P_{0}\oplus||T_{Wi}(T_i(x))||T_{Wi}(T_i(x))))$, instead of $T_{Wi}(T_i(x))$ to resist session-specific temporary permanent information attacks. In addition, our scheme provides comprehensive features including revocation and re-registration.

In all, the proposed three-factor AKA protocol meets all 13 evaluation indicators $EC_1$–$EC_{13}$ outlined in Table 2. The protocol comprises six phases, namely: setup, registration (as depicted in Fig. 1), login and authentication (Fig. 2), password or biometrics updating (Fig. 3), revocation and re-registration.

3.1 System Setup Phase

The server $S$ randomly selects a number $s \in Z^*_n$ as well as two one-way hash-functions $H(\cdot)$ (SHA-160) and $H_0(\cdot)$ (SHA-320). Then, $S$ calculates the public key $T_i(x)$, publicizes these parameters $(T_i(x), x, H(\cdot), H_0(\cdot)$, and keeps a long private key $s$ as a secret.

3.2 Registration Phase

In order to resist privileged insider attack, we only send identity $ID_i$ to the server $S$. 
Step 1. The user $U_i$ selects an $ID_i$ and sends it to the server $S$.

Step 2. Upon getting $\{ID_i\}$, $S$ picks $e_i, r_i \in Z_n^*$ and computes $CID_i = H(ID_i||e_i)$, $VPWI = H(PWI||ID_i||CID_i||e_i)$, and stores $\{ID_i, r_i, Honey\_List = Null\}$ in its database, inputs $T_i(x), x, CID_i$ to a new smart card $SC_i$, and finally sends $SC_i$ to $U_i$.

Step 3. Upon receiving the smart card $SC_i$ from the server $S$, the user $U_i$ inputs her new password and fingerprints $BIO_i$ into $SC_i$. Then, smart card $SC_i$ randomly generates a number $2^4 \leq n_0 \leq 2^8$ and calculates some important parameters: $GEN(BIO_i) = (\sigma_i, T_i)$, $VPWI = H(PWI||ID_i||CID_i||e_i)$, $W_i = H((H(ID_i) \oplus VPWI) \mod n_0)$, $B_i = B_0 \oplus VPWI \oplus \sigma_i$. Finally, the smart card $SC_i$ contains the following parameters: $\{CID_i, B_i, W_i, T_i, T_s(x), x, n_0\}$ and $\{H(\cdot), H_0(\cdot), GEN(\cdot), REP(\cdot)\}$.

Remark 1. Traditionally, the $Honey\_List$ contains all the honeypots (seemingly like the real password but not correct) that have been tried by the attacker $[55, 56]$. In this work, the $Honey\_List$ contains all the honey parameters $B_0'$ (seemingly like the real parameters $B_0 = H(\cdot)$) that have been tried by the attacker $[55, 56]$.

<table>
<thead>
<tr>
<th>User ($U_i$)</th>
<th>Secure Channel</th>
<th>Server ($S$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Login and Authentication Phase:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inserts $SC_i$ and inputs $ID_i, PW, BIO_i$</td>
<td>Computes $\sigma_i = REP(BIO_i, T_i)$</td>
<td>$VPWI = H(PWI</td>
</tr>
<tr>
<td>Generates a random number $\nu$</td>
<td>Computes $C_2 = T_i(C_1)$</td>
<td>$CID^{new} = H(CID</td>
</tr>
<tr>
<td>Computes $C_1 = T_s(x), C_3 = T_s(T_i(x))$</td>
<td>Generates a random number $\nu^{new}$</td>
<td>$RP^{new} = H(ID_i</td>
</tr>
<tr>
<td>Generates $C_2 = T_s(C_1)$</td>
<td>Computes $C_2 = T_s(C_1)$</td>
<td>$RP^{new} = H(ID_i</td>
</tr>
<tr>
<td>${C_2, C_3, C_4}$</td>
<td>Computes $C_3 = T_s(C_2)$</td>
<td>$RP^{new} = H(ID_i</td>
</tr>
<tr>
<td>Computes $C_3 = T_s(C_2)$</td>
<td>Generates a random number $\nu^{new}$</td>
<td>$RP^{new} = H(ID_i</td>
</tr>
<tr>
<td>${C_2, C_3, C_4}$</td>
<td>The common session key: $SK = SK^{new} = SK^{new}$</td>
<td>$RP^{new} = H(ID_i</td>
</tr>
<tr>
<td>$VPWI^{new} = H(PWI</td>
<td></td>
<td>ID_i</td>
</tr>
<tr>
<td>$W_i^{new} = H((H(ID_i) \oplus VPWI^{new}) \mod n_0)$</td>
<td>Updates ${ID_i, Honey_List}$</td>
<td>$B_0 = (B_i \oplus VPWI \oplus \sigma_i)$</td>
</tr>
<tr>
<td><strong>Update Phase:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Computes $\sigma_i = REP(BIO_i, T_i)$</td>
<td>$B_0 = (B_i \oplus VPWI \oplus \sigma_i)$</td>
</tr>
<tr>
<td>${ID_i, PW, BIO_i, Updating_Request}$</td>
<td>Generates two random numbers $e_i, r_i \in Z_n^*$</td>
<td>$B_0 = (B_i \oplus VPWI \oplus \sigma_i)$</td>
</tr>
<tr>
<td>Computes $CID_i = H(ID_i</td>
<td></td>
<td>e_i)$</td>
</tr>
<tr>
<td>$B_0 = H(ID_i</td>
<td></td>
<td>e_i</td>
</tr>
</tbody>
</table>
| $SC_i, B_0$ | New smart card: $SC_i := \{T_s(x), x, CID_i, H(\cdot)\}$ | Otherwise, rejects.

**Fig. 1. User registration.**

**Fig. 2. Login and authentication.**

**Fig. 3. Password or biometrics updating.**

$H(ID_i||\nu^{new})$ but not correct) that have been tried by the attacker whenever the honey parameters $B_0'$ is being tried, it indicates that the attacker has extracted the password verifier $W_i$ stored in the user’s smart card. See more detailed analysis in Section 4.2.6.

### 3.3 Login and Mutual Authentication Phase

After the user $U_i$ is registered with the server $S$ successfully, she transmits the login request to $S$ when she wishes to obtain some service – see below:

Step 1. $U_i$ inputs her smart card $SC_i$ into a card reader, and provides $ID_i, PW, and BIO_i$ to $SC_i$. Then, $SC_i$ computes $\sigma_i = REP(BIO_i, T_i)$, $VPWI = H(PWI||ID_i||CID_i||e_i)$, and checks whether $W_i = H((H(ID_i) \oplus VPWI) \mod n_0)$. If not, $SC_i$ rejects the login request. Otherwise, $SC_i$ computes $B_0 = B_i \oplus VPWI \oplus \sigma_i$. Subsequently, $SC_i$ picks $\nu \in Z_n^*$ and computes $C_1 = T_s(x)$, $C_2 = T_s(T_i(x))$, $C_3 = (ID_i||B_0)$, $H_0(C_2)$, $C_4 = H(ID_i||CID_i||B_0||C_2||C_3)$. Finally, $SC_i$ sends $\{CID_i, C_1, C_2, C_3\}$ to $S$.

Step 2. After obtaining $\{CID_i, C_1, C_2, C_3\}$, $S$ calculates $C_2 = T_s(C_1), (ID_i||B_0') = C_0 \oplus H_0(C_2), C_3 = (ID_i||B_0')$ and searches whether $C_1 = H(ID_i||CID_i||B_0||C_2||C_3)$. If they are unequal, $S$ terminates this session. Otherwise, $S$ checks $\nu$ whether the derived $B_0'$ equals the computed $B_0$. If they are equal, $S$ proceeds to the next step. If they are unequal, $S$ knows that $U_i'$'s smart card has been corrupted and the adversary did not get the real password. Accordingly, $S$ inserts the honeyword $B_0'$ into $Honey\_List$ and wraps this login request. Moreover, if $[Honey\_List] > N_0$ (Such as $N_0 = 5$, where $N_0$ is a threshold value, $S$ suspends the use of $SC_i$ until $U_i$ re-registers and requests to restore $SC_i$. Otherwise, $S$ picks $\nu^{new} \in Z_n^*$ and calculates $CD^{new} = H(ID_i||\nu^{new})$, $B_0^{new} = H(ID_i||\nu^{new})||CD^{new}$. Subsequently, $S$ updates $\{ID_i, Honey\_List = Null\}$ in its backend database. Moreover, $S$ picks $\nu \in Z_n^*$ and computes $\nu^{new}$.
C_3 = T_3(x), C_0 = T_0(C_1), C_2 = (CID_1^{\text{pre}} \parallel B_0^{\text{pre}}) \oplus H(C_0 \parallel B_0), SK_u = H(ID_1 || CID || B_0^{\text{pre}} || C_2 || C_0) \text{ and } C_5 = H(ID_1 || CID || B_0^{\text{pre}} || C_2 || C_0). Lastly, S sends \{C_3, C_7, C_8\} to SC_i.

Step 3. Upon receiving the message \{C_3, C_7, C_8\}, SC_i computes \(\text{C}_4 = T_4(C_3), (\text{CID}_i^{\text{pre}} \parallel \text{B}_0^{\text{pre}}) = C_7 \oplus H(C_0 || B_0),\) \(\text{SK}_u = H(ID_1 || CID || B_0^{\text{pre}} || C_2 || C_0),\) and \(C_5 = H(ID_1 || CID || B_0^{\text{pre}} || C_2 || C_0).\) If not, SC_i aborts this session. Otherwise, SC_i considers a shared key \(SK = SK_u = SK_s\) is being shared with S. Subsequently, SC_i randomly generates a number \(2^t \leq n_0^{\text{new}} \leq 2^s\) and calculates \(VPW^{\text{new}} = H(PW || ID_1 || CID || \text{sk}_i), W^{\text{new}} = H(H(ID_1) \oplus VPW^{\text{new}} \mod n_0^{\text{new}}),\) and \(B_1^{\text{new}} = B_0^{\text{pre}} \oplus VPW^{\text{new}} \oplus \text{sk}_i.\) Finally, the smart card SC_i replaces \{CID_i, W_i, B_1, n_0\} with \{CID_i, W_i, B_1^{\text{new}}, n_0^{\text{new}}\}.

### 3.4 Password or Biometrics Update Phase

U_i injects her smart card into the card reader, and provides \(ID_u, PW_i,\) and \(BIO_i\). Then, SC_i calculates \(\sigma_i = REP (BIO_i, \tau_i), VPW_i = H(PW_i || ID_1 || CID_i || \sigma_i)\) and checks whether \(W_i = H(H(ID_1) \oplus VPW_i) \mod n_0.\) If not, SC_i declines this update request. Otherwise, SC_i acknowledges this update request.

Case I. If \(U_i\) only wants to change the password, then she inputs a new password \(PW^{\text{new}},\) SC_i will then randomly generate \(2^t \leq n_0^{\text{new}} \leq 2^s\) and calculates \(VPW^{\text{new}} = H(PW^{\text{new}} || ID_1 || CID_i || \sigma_i), W^{\text{new}} = H(H(ID_1) \oplus VPW^{\text{new}} \mod n_0^{\text{new}}),\) and \(B_1^{\text{new}} = (B_1 \oplus VPW_i || \text{sk}_i) \oplus VPW^{\text{new}} \oplus \text{sk}_i.\) Finally, SC_i replaces \{W_i, B_1, n_0\} with \{W_i^{\text{new}}, B_1^{\text{new}}, n_0^{\text{new}}\}.

Case II. If \(U_i\) only wants to change her biometrics, then she inputs a new biometrics \(BIO^{\text{new}},\) then \(SC_i\) randomly generates a number \(2^t \leq n_0^{\text{new}} \leq 2^s\) and calculates \(GEN (BIO^{\text{new}} = (\sigma_i^{\text{new}}, t_i^{\text{new}}), VPW^{\text{new}} = H(PW^{\text{new}}, ID_1 || CID_i || \sigma_i^{\text{new}}), W^{\text{new}} = H(H(ID_1) \oplus VPW^{\text{new}}) \mod n_0^{\text{new}},\) and \(B_1^{\text{new}} = (B_1 \oplus VPW_i || \text{sk}_i) \oplus VPW^{\text{new}} \oplus \text{sk}_i.\) Finally, SC_i replaces \{W_i, B_1, t_i, n_0\} with \{W_i^{\text{new}}, B_1^{\text{new}}, t_i^{\text{new}}, n_0^{\text{new}}\}.

### 3.5 Revalidation Phase

In this subsection, we provide a feature that allows the user to block a stolen or misplaced smart card:

Step 1. \(U_i\) verifies the authentication of smart card is similar to the login phase. If \(SC_i\) authenticates \(U_i\), as a legitimate user, \(SC_i\) sends the revocation request \{CID_i, C_3, C_1, \text{Revoke_request}\} to the server.

Step 2. Upon getting the revocation request, \(SC_i\) authenticates \(SC_i\) by verifying \(C_3,\) if it is found to be invalid, \(S\) denies this revocation request. Otherwise, \(S\) sets \(|\text{Honey_List}| > N_0\) such that \(SC_i\) is revoked. Finally, \(SC_i\) is suspended until \(U_i\) re-registers.

### 3.6 Re-Registration Phase

If a legitimate user’s smart-card is revoked or the number of times that the user cannot be authenticated by the server \(S\) exceeds the maximum threshold value, then \(U_i\) is required to re-register. In this case, \(U_i\) will not be able to log into the system even though she inputs the correct values \{ID, PW, BIO\}. However, \(U_i\) can re-register by performing the following steps:

Step 1. \(U_i\) submits the re-registration request \{ID, \text{Register_request}\} to \(S\) through a private channel.

Step 2. Upon receiving the request message from \(U_i\), the server \(S\) uniquely identifies the user by checking her identity information such as her SSN, national card number, or some other relevant legal identity documents issued by the government. Afterwards, if \(ID_i\) is found in the database, \(S\) confirms whether \(|\text{Honey_List}| > N_0\). And if \(SC_i\) is found to be revoked, then \(S\) accepts this request and performs Steps 2 and 3 of the Registration phase to complete re-registration. Otherwise, \(S\) rejects this request.

### 4 Security analysis

In this section, we provide the formal security proof, heuristic security analysis and security simulation with AVISP for the proposed protocol. Among them, the security simulation based on AVISPA software can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2020.3022797 and also at http://wang.dingg.weebly.com/publications.html. For ease of description, we abbreviate the proposed protocol to CM3FAKAP.

#### 4.1 Formal Security Proof

We will now formally prove the security of CM3FAKAP under the CMCDH assumption in the random oracle model.

#### 4.1.1 Security Model

The security of CM3FAKAP relies on the BPR2000 [57] and Bresson [58] models, and we also use the technique of Wang et al. [39] in our proof.

**Participants.** A three-factor AKAP involves two participants, namely: \(U\) and \(S\). Each participant has a number of distinct instances called oracles. The \(i\)th instance of \(U\) and the \(j\)th instance of \(S\) are denoted by \(U^i\) and \(S^j\), respectively. Besides, any instance is denoted as \(I\).

**Queries.** The interaction between an adversary \(A\) and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack. The types of queries that can be used by \(A\) are as follows:
that \( A \) asks \( q_4 \) times Hash-queries, \( q_3 \) times Execute-queries and \( q_4 \) times Send-queries. Then,

\[
\text{Adv}_{\text{CM3FAKAP},\text{P}}^{\text{AKA}}(A) \leq 2C^e \cdot q_4^t + \frac{(q_3 + q_3 + q_4^2)}{2^{t-1}} + \frac{2(q_4 + q_4^3)}{p} + 2q_4 \text{Adv}_{\text{CMCDH}}(t),
\]

where \( \mathcal{D} \) refers to password space that follows a Zipf’s law \( [31] \) in terms of frequency distribution, \( s' \) and \( C' \) denote the Zipf parameters, \( t \) is the bit length of the output of the Hash function \( (e.g., l = 160) \), \( t' \leq t + (q_4 + q_4 + 1)T_c \). \( T_c \) is the computational time required for the extended chaotic-maps.

**Proof.** Suppose that the adversary \( A \) is able to break the security of the CM3FAKAP, in this case, we design an algorithm \( B \) that solves the Chaotic-maps computational Diffie-Hellman problem (CMCDHP). More precisely, \( B \) responds \( T_{\mathcal{E}}(x) \) against the instance \((x, T_{\mathcal{E}}(x), T_{\mathcal{E}}(x))\) of CMCDHP, where \( \theta, \varphi \) are two integers. The proof is comprised of \( 7 \) sequence of games: \( E_0, E_1, \ldots, E_5 \). Let \( E_i \) refer to the event that \( A \) outputs the correct \( b \) in \( E_i \) where \((i = 0, 1, 2, 3, 4, 5) \).

**Game E_0.** In this experiment, the simulation of real attack is executed in the random oracle model. \( A \) has the access to all the oracles. Thus, we have

\[
\text{Adv}_{\text{CM3FAKAP},\text{P}}^{\text{AKA}}(A) = |2Pr[E_0] - 1|.
\]

**Game E_1.** This experiment simulates the random oracle \( H \) by managing a hash list \( \Lambda_{\mathcal{H}} \) and \( \Lambda_{\mathcal{A}} \). Since all oracles are simulated as the real attack, therefore, this experiment cannot be distinguished from the actual execution of the protocol. Fig. 4 depicts all queries asked by \( A \). Thus, we have

\[
|Pr[E_1] - Pr[E_0]| = 0.
\]

**Game E_2.** This experiment simulates all kinds of queries just like in game \( E_1 \), except that the simulation is aborted in the following two cases: (1). Collisions on the output of hash queries, (2). Collisions on the partial transcripts: \((C_{\mathcal{E}}(1, C_1, C_4), (C_5, C_7, C_8))\). According to the birthday-paradox, we have

\[
|Pr[E_2] - Pr[E_1]| \leq \frac{q_3^2}{2^{t+1}} + \frac{(q_3 + q_3^2)^2}{2p}.
\]

**Game E_3.** We simulate this experiment just like the prevalent game, the only difference is the termination of protocol in case of \( A \) correctly guesses the authentication elements \( C_1 \) and \( C_8 \) without querying the random oracle. This experiment is indistinguishable from the previous experiment except that the instance \( U' \) (or \( S' \)) refuses a correct authentication element. Therefore, we have

\[
|Pr[E_3] - Pr[E_2]| \leq \frac{q_3}{2}.
\]

**Game E_4.** In this experiment, the \( S' \) is guessed without querying the corresponding random oracle, i.e., the session key is independent from \( K = T_{\mathcal{E}}(x) \) and \( \mathcal{H} \). Accordingly, this experiment is indistinguishable from
The ability to obtain $H(q)(H_0(q))$ or $H(q')(H_0(q'))$, if the record $(q, r)$ or $(0, q, r)$ belongs to $\Lambda_H$. Otherwise, $r$ as an answer is defined as: randomly select an element $r \in \{0, 1\}$, and $(q, r)$ is added to $\Lambda_H$. If the adversary directly asks the query, $(q, r)$ or $(0, q, r)$ will be added to $\Lambda_H$.

**Send(U', Start)-query**: Randomly select a number $n \in Z_{2^n}$ and calculate $C_1 = T_0(z), C_2 = T_2(T_0(z)), N_0 = B_1 \lor V_{PW_1} \lor \pi_1, C_3 = (I_D(B_1) \lor H_0(C_2))$ and $C_4 = H(1D) || C_1 || |B_0| || C_2 || C_3$. Afterwards, the query is answered with $(C_1, C_2, C_3, C_4)$.

**Send(S', (C1D, C3S, C4S))-query**: Compute $C_5 = T_0(C_1), (I_D (B_1) = C_5 \lor H_0(C_2)), B_0 = H(1D) || |r_1| || C_1 || C_4 = H(1D) || C_1 || |B_0| || C_2 || C_3)$. If:
- Reject if $I_D$ cannot be searched.
- Reject if $C_5$ is not equal to $C_4$.
- Reject if $B_0$ is not equal to $B_0$.
- Inserts $B_1$ into $\text{Honey_Li}$ if $\text{Honey_Li} < N_0$.
- Suspend $SC_i$ if $\text{Honey_Li} > N_0 \land B_0_0$.

Aftetwards, generate a random number $e_{new}$ and compute $CIDS_{new} = H(ID) || |e_{new}|, B_{new}^R = H(ID) || |r_1| || CIDS_{new}$. Choose a random number $v$ and compute $C_5 = T_0(x), C_6 = T_0(C_1), C_7 = (CIDS_{new} || B_{new}^R) \lor H(C_6)$. $SK_x = H(ID) || |CIDS_{new}| || |B_0| || C_5 || C_7$. Thus, the query is answered with $(C_5, C_6, C_7)$.

**Send(U', (C_5, C_6, C_7))-query**: Compute $C_6 = T_0(C_6), (CIDS_{new} || B_{new}^R) = C_5 \oplus H(C_6) \land B_0, SK_x = H(ID) || |CIDS_{new}| || |B_0| || C_5 || C_7).$ If $C_5$ and $C_7$ are not equal. Otherwise, generate a new integer $2^l \leq l' \leq 2^l$ and compute $B_{new}^R = H(PW_1 || ID) || |CIDS_{new}^l| \oplus v$ and $B_{new}^R = B_{new}^R \lor V_{PW_1} \lor \pi_1, \text{Replace} (CIDS_{new}, B_{new}^R, B_{new}^R, B_{new}^R)$, respectively.

**Execute(U', S')-query**: The query employs the successive simulations of the Send-queries to process: $(CIDS, C_1, C_3, C_4) \rightarrow \text{Send}(U', S')$, $(C_5, C_6, C_7) \rightarrow \text{Send} (S', (C_1D, C_3S, C_4S))$, and yields the transcript $(CIDS, C_1D, C_3S, C_4S)$.

**Corrupt(U)-query**: The query responds with $PW$, and $(CIDS, C_1D, C_3S, C_4S)$, $\pi_0, H(I_D), GEN, REP, \text{stored} in$ the smart-card in case if $s = 1$; returns the biometrics $BO_0$, and $(CIDS, C_1D, C_3S, C_4S)$, $\pi_0, H(I_D), GEN, REP, \text{stored} in$ the smart-card in case if $s = 2$; returns the password $PW$, and biometrics $BO_0$ if $s = 3$.

**Corrupt(S)-query**: Return the long-term private $s$ of $S$ if $s = 1$.

**Reveal(I)-query**: The query responds with $(SK_x)$ or $(SK_x)$ calculated by the instance $I$ (if the latter has accepted).

**Test(I)-query**: The query first obtains $SK$ from Reveal(I). Afterwards, a coin $6$ is flipped. If $b = 0$, return a random number of the same length as $SK$ from $(0, 1)^b$, otherwise, return $SK$.

If $A$ fails to ask any $H$ query with the valid input, then there is no advantage for this experiment to distinguish the real $SK$ with the session key of the same size made from the random value. Therefore, we have $|Pr[E_s] - Pr[E_s]| \leq C' + q_h^2/2^t$. Thus, our protocol reduces the communication and computation cost in the event of incorrect inputs or the attack triggered by an illegitimate user. Accordingly, the probability of getting $T_{w}(x)$ is: $q_h^2/2^t$ at most. Therefore, we have

$$|Pr[E_s] - Pr[E_s]| \leq C' + q_h^2/2^t. (4)$$

**Remark 3.** The Canetti and Krawczyk (CK) model [60] allows partial/empheral information leakage. CK model is suitable for authentication and key exchange (AKE) protocols, but not suitable for analyzing password-based authentication protocols. In AKE protocols, it means that both sides of the communication have a digital certificate (i.e., a private key with high entropy), while password-based authentication schemes like ours are cryptographic protocols that use memorable low entropy password in order to eliminate the management problem of digital certificates. If password-based authentication protocols are analyzed with the CK model (i.e., temporary key information can be disclosed, partial information leakage), they cannot resist offline password guessing attack. Therefore, the BPR2000 model [57], Bresson et al.’s model [58], and the model of Wang et al. [39] are widely used to analyze the password-based authentication schemes [2], [3], [39] (and our proposed protocol).

### 4.2 Security Analysis Based on Evaluation Criteria

We will now evaluate the security of the proposed protocol, based on the evaluation criteria and adversary model presented in Section 2.

#### 4.2.1 Timely Password Typo Detection

In the login phase of both protocols of Li et al. and Zhu et al., $SC_i$ has no capability to verify the login activity of any user until it is detected by $S$. However, the login phase of our protocol checks the password’s validity by verifying whether $W_i = W_i$ after inputting $(ID, PW_i, BO_i)$. If these are found to be valid, then $SC_i$ submits the request message to the server. Otherwise, it aborts this session. By doing so, our protocol reduces the communication and computation cost in the event of incorrect inputs or the attack triggered by an illegitimate user. Accordingly, the timely password typo detection is successfully handled by the proposed protocol.
4.2.2 User-Anonymity and Un-Traceability

In the proposed protocol, there is no identity information to be transmitted through the public channel or to be stored in the user's smart-card. Therefore, the adversary may only obtain user's identity during the communication session.

Suppose that A extracts all parameters \(\{CID_i, B_t, W_t, r_t, T_e(x), x, n_0, H(\cdot), H_0(\cdot)\}\) stored in \(SC_t\), and obtains \(\{CID_t, C_1, C_2, C_3, C_4, C_5, C_7, C_8\}\) from U, and S by eavesdropping. \(CID_t\) can be calculated by a one-way function and is updated during every session. \(C_3 = (ID_t||B_t) \oplus B_t||C_2\) is variable because \(C_2\) is random. \(C_4 = H(ID_t||CID_t||B_t)\) is also variable and is secure due to the use of a one-way function. Therefore, A is unable to derive or to trace the user's identity \(ID_t\). Thus, the proposed protocol ensures user anonymity and un-traceability.

4.2.3 Privileged Insider Attack

During the registration step of the proposed protocol, \(U_i\) only sends \(ID_t\) to \(S\) without any information relating to the password or biometrics. Then, \(S\) sends \(B_0\) and a new smartcard \(SC_t\) to \(U_i\). After receiving \(SC_t\) and \(B_0\), \(U_i\) activates \(SC_t\) by providing \(PW_t\) and \(BIO\), that are only known to \(U_i\). Finally, \(U_i\) has \(SC_t\) with the new parameters. Observing all parameters stored in \(SC_t\), we find that \(PW_t\) is not available in plaintext but is secure due to the use of a one-way function. Moreover, it is also hard for \(A\) to retrieve the user's biometrics. Therefore, the presented protocol provides resistance to privileged insider attacks.

4.2.4 Key-Compromise User Impersonation Attack

In the proposed protocol, even if the long-term private key \(s\) of \(S\) is revealed to \(A\), \(A\) will not be able to impersonate the legitimate user \(U_i\) to \(S\). This is because \(A\) is unable to obtain the value of \(B_0\) to forge the login request message \(C_t\). \(B_0\) can be computed by two ways: (1) The legitimate user can calculate it because she knows \((ID_t, PW_t, B_t, CID_t)\) and can retrieve \(s_t\). (2) The server is able to compute it due to knowledge of \(\{c_t, r_t\}\) and \(s\). However, it is computationally challenging for \(A\) to obtain these key parameters. Thus, \(A\) has no capability to impersonate the user to \(S\). Accordingly, the presented protocol is resilient to such attack.

4.2.5 Server Impersonation Attack

To impersonate the server \(S\), \(A\) has to compute valid \(\{C_5, C_7, C_8\}\). Since \(C_7 = (CID_t\text{enc}||B_0\text{enc}) \oplus H(C_0||B_0)\) and \(C_8 = H(ID_t||CID_t||B_t||B_0\text{enc}||C_2||SK_t)\) are computed by \(B_0, ID_t, CID_t\text{enc}, B_0\text{enc}, C_2, SK_t\) and protected by a one-way function, \(A\) must know these key parameters or to guess correct value within polynomial time. To obtain these key parameters, \(A\) needs to have \(\{ID_t, s, r_t\}\). However, it is computationally infeasible for \(A\) to guess these private parameters within polynomial time, and the proposed protocol preserves user anonymity according to Section 4.2.2. Therefore, \(A\) is unable to compute valid response message, and hence the proposed protocol is resilient to server impersonation attacks.

4.2.6 Password-Guessing-Attack

In the presented CM3FAKAP, we suppose that \(A\) eavesdrops all messages over the public channel and is capable of stealing biometrics of the user and extracting all parameters from the user’s smart card. Therefore, we analyze the password-guessing-attack from two aspects: (i) Off-line guessing attack: On the one hand, we assume that the user's identity \(ID_t\) is anonymous. Then, \(A\) retrieves \(s_t\) and guesses \(\{ID_t, PW_t\}\) from dictionary guessing space. Afterwards, \(A\) will be able to compute related parameters \(VPW_t = H(PW_t||ID_t||CID_t||s_t)\) and \(VPW_0 = H((H(ID_t) \oplus VPW_t) \text{mod } n_0)\). Subsequently, \(A\) checks whether \(VPW_t = VPW_0\). Finally, \(A\) repeats the above steps until she correctly guesses the identity and password. Clearly, \(A\) can guess the related \(ID_t\) and \(PW_t\), which satisfy \(VPW_t = VPW_0\) within a polynomial time. The reduced guessing size of identity and password space is presented as \(\frac{|PWI_0|}{|PW_t|}\), where \(2^4 \leq |n_0| < 2^5\). During the registration step of the proposed protocol, the login request message \(Sc(i)\) is revealed to \(A\), and obtained \(PID_t\) from \(S\). Thus, the proposed protocol ensures user anonymity and un-traceability.

4.2.7 De-Synchronization Attack

During the update phase, as long as the identity, password and biometric pass the verification of the smart card, the user is allowed to change the password or biometrics. Thus, the adversary remains incapable of performing the underlying attack. In the login and authentication phase, even if the attacker blocks the first message flow, the server is not required to update key parameters at the backend database. Thus, this action does not influence the consistency of next communications. Although the attacker blocks the second messages flow, it also cannot influence the consistency of communications between user and server, because only when \(C_8 = C_8^t\) holds, the smart card will update the data of smart card. Thus, the proposed protocol can resist de-synchronization attacks.

4.2.8 Replay Attack

Assume that \(A\) has obtained the login request message \(\left\{CID, C_1, C_3, C_4\right\}\) of the previous session through some means. To impersonate the server, \(A\) needs to have \(\{ID_t, PW_t, C_1, C_2, C_3, C_4\}\). Therefore, the proposed protocol ensures user anonymity and un-traceability.
public channel. If A replays \(\{CID_i, C_1, C_3, C_4\}\) to S, then S verifies \(C_4\). Since \(\{CID_i, B_0\}\) are updated in every successive session, \(C_1\) is also changed to \(C_2^{new}\) each time. Therefore, the other \(C_4\) cannot be verified by the server during the current session. Otherwise, if A replays \(\{C_5, C_7, C_8\}\) to \(U_i\), then the older message \(C_4\) also cannot be verified by \(U_i\) during the current session. Therefore, the proposed protocol can resist replay attacks.

4.2.9 Session-Specific Temporary Information Attack

In the proposed protocol, if all temporary information \(u, v, e_1\) are compromised, then A can compute \(C_2, C_6\). To reveal \(SK = H(\{ID_i\}||\{CID_i\}||\{B_0\}||B_0^{new}||\{C_2\}||C_6)\), A needs to calculate \(\{B_0, B_0^{new}\}\) and to know the correct \(ID_i\). Since A has no way of figuring out the correct \(B_0, B_0^{new}\) computed by using the long-term private key \(s\) of the server. Therefore, the proposed protocol is resilient to session-specific temporary information attacks.

4.2.10 Man-in-the-Middle Attack

In the presented protocol, we suppose that A intercepts and blocks the login request message \(\{CID_i, C_1, C_3, C_4\}\), the challenge message \(\{C_5, C_7, C_8\}\), and extracts all parameters of \(SC_i\). To be successful in the man-in-middle attack, A has to forge the new message flow \(\{\{CID_i, C_1^*, C_3, C_4\}\}, \{C_5^*, C_7, C_8\}\) or to replay the previous message flow. As discussed earlier, the proposed protocol can resist impersonation attacks and replay attacks. Therefore, A is not capable of being authenticated by both user and server. Hence, the proposed protocol is resilient to man-in-middle attacks.

4.2.11 Mutual Authentication

In the proposed protocol, S authenticates \(U_i\) by verifying whether \(C_4^* = C_4\), while \(U_i\) authenticates \(S\) by checking whether \(C_4^* = C_8\). After mutual authentication, \(U_i\) and \(S\) negotiate a common session key \(SK\). Thus, the proposed protocol achieves secure mutual authentication.

4.2.12 Perfect Forward Secrecy

This feature states that even if \(PW_i\) and \(BIO_i\) of \(U_i\) and all secret keys of \(S\) are revealed to \(A\), all prior session keys must remain secret. Let us suppose that all private keys \(\{s, r_1, c_1\}\) of \(S\) and \(\{PW_i, BIO_i\}\) of \(U_i\) are compromised, and \(A\) extracts \(\{CID_i, B_1, W_i, \tau, T_s(x), x, n_0, H(\cdot), H_0(\cdot)\}\) stored in the smart card and eavesdrops and \(\{\{CID_i, C_1, C_3, C_4\}, \{C_5, C_7, C_8\}\}\). Then, \(A\) can figure out \(C_2 = T_s(C_1)\) and \(\{ID_i\}||B_0 = C_3 + H_0(C_2)\). However, to compute the previous session key \(SK = H(\{ID_i\}||\{CID_i\}||B_0||B_0^{new}||\{C_2\}||C_6)\), \(A\) needs to know \(C_6 = T_s(C_3) = T_s(C_1)\). However, it is computationally infeasible for \(A\) to extract random number \(u\) from \(C_1\) or \(r\) from \(C_3\) and to calculate \(C_6\) due to the intractability of CMDLP and CMCDHP. Thus, even if all secret parameters are compromised, \(A\) is not capable of computing \(SK\). Therefore, the proposed protocol achieves perfect forward secrecy.

4.2.13 Efficient Update Phase

In both protocols of Zhu et al. [22] and Li et al. [24], to update the user’s password or biometric, the user is required to communicate with the server. This has implications on the communication and computation cost. However, in the proposed protocol, \(U_i\) can update her password or biometrics only by interacting with \(SC_i\). Moreover, \(S\) does not need to participate in update phase. Hence, this reduces the associated communication and computation cost.

4.2.14 Three-Factor Security

It is known that any three-factor AKA protocol \(P\) incorporates three features including password, smart-card and biometric. This implies that \(P\) is different from the single-factor and two-factor AKA protocol. We now let \(A\) obtain any two of the three factors. According to the previous analysis, \(B_0 = B_1 \oplus VPW_i \oplus \sigma_i\) is the key parameter to launch any attack and compute the session key. Thus, in this section, we will explain why \(A\) cannot compute \(B_0\) in any of these combinations of knowledge: password and smart-card, password and biometric, biometric and smart-card.

- **Case I**: Although \(A\) knows the password and can extract all parameters in the smart card, she still cannot compute \(B_0 = B_1 \oplus VPW_i \oplus \sigma_i\), to forge the valid login message without having access to biometric information \(\sigma_i\).

- **Case II**: Even if \(A\) obtains password and biometric information without the smart card, she cannot compute \(B_0\) without \(B_1\), because \(B_0 = B_1 \oplus VPW_i \oplus \sigma_i\), and \(B_1\) are stored in the smart-card.

- **Case III**: Although \(A\) has both biometric information and the smart card, she cannot compute \(VPW_i\), without knowing the correct \(PW_i\). So \(A\) is unable to compute \(B_0 = B_1 \oplus VPW_i \oplus \sigma_i\).

Therefore, the proposed protocol achieves three-factor security, which is very vital for the three-factor authentication protocols.

5 COMPARATIVE SUMMARY: SECURITY AND PERFORMANCE

To demonstrate the good balance of the proposed AKA protocol in security and usability, this section provides a comparative measurement of the security, computation and communication cost of Islam et al. [4], Wazid et al. [2], Tsai et al. [29], Roy et al. [3], Liu et al. [21], Zhu et al. [22], Jiang et al. [23]’s AKA protocols and the proposed protocol.

5.1 Security Comparison

Now we use the evaluation metric introduced in Section 2.4 to perform an objective comparison of our new protocol with seven state-of-the-art protocols. The comparison results of [2], [3], [4], [21], [22], [23], [29] and the presented protocol are depicted in Table 3.

In schemes in [2], [3], [4] and [22], smart cards store an explicit password validation parameter, these protocols cannot resist off-line password guessing attacks. While there is no password validation parameters in [21] and [28], this will lead to denial of service attack. The scheme in [3] uses only three chaotic operations, unfortunately, it still fails to provide perfect forward security when long-term private key leaks. Despite the use of elliptic curve cryptography in [29], this protocol stores a key in storage device, which directly results in the failure of forward...
only one that is immune to various known attacks and from Table 3, we can deduce that the presented protocol is the EC col [29] does not provide smartcard revocation function. That is, [2] does not achieve sessionspecific temporary information attack and clock synchronization attack, nor provide three-factor security and smartcard revocation function. That is, [2] does not achieve EC5, EC7, EC10, EC11 and EC12. Similarly, Tsai et al.’s protocol [29] does not provide EC5, EC6, EC11, EC12 and EC13, and Roy et al.’s protocol [3] does not satisfy EC6, EC7, EC10, EC12 and EC13. In a nutshell, by observing Table 3, we can deduce that the presented protocol is the only one that is immune to various known attacks and achieves the desirable security and usability goals.

5.2 Computation and Communication Cost Comparison
To analyze the computational complexity of login and authentication step, the lightweight operations including exclusive-OR and string concatenation operation are omitted. For better presentation, the following additional notations are used, i.e., Tt is the computational time for attacking the extended chaotic-maps, Tm is the computational time for executing an elliptic curve point multiplication, Ta is the computational time for executing an elliptic curve point addition, Tp is the computational time for symmetric cryptogra phy operation, Tc is the computational time for operating a one-way hash operation.

The cost of server-end and user-end required for the protocols in login and authentication phase is presented in Table 4. The total computational cost required in the proposed protocol during login and authentication phase is 18Tt + 6Tc. Also, we can observe that the proposed protocol requires slightly more computational cost, as compared to the other protocols. This is because of the additional functionalities and better security (i.e. trade-off between security and usability). To measure the communication cost in the login and authentication phase, we use the length of the security parameter as follows: the length of each component of the function GEN(·) output is 80 bits, the length of the user identity is 160 bits, the bit length of the prime number p is 256 bits, the bit length of the random number is 128 bits, the length of an elliptic curve point is 160 bits, the output length of the hash function (SHA-160) is 160 bits, the length

<p>| TABLE 3 | Security Comparison Among Relevant Three-Factor AKA Protocols |</p>
<table>
<thead>
<tr>
<th>Protocols</th>
<th>EC1</th>
<th>EC2</th>
<th>EC3</th>
<th>EC4</th>
<th>EC5</th>
<th>EC6</th>
<th>EC7</th>
<th>EC8</th>
<th>EC9</th>
<th>EC10</th>
<th>EC11</th>
<th>EC12</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roy et al. (2018)[3]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
<td>CMCDHP</td>
</tr>
<tr>
<td>Islam et al. (2018)[4]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
<td>ECDDHP</td>
</tr>
<tr>
<td>Wazid et al. (2017)[2]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
<td>ECDDHP</td>
</tr>
<tr>
<td>Liu et al. (2016)[21]</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>CMCDHP</td>
</tr>
<tr>
<td>Jiang et al. (2016)[23]</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>CMCDHP</td>
</tr>
<tr>
<td>Zhu et al. (2015)[22]</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>CMCDHP</td>
</tr>
<tr>
<td>Tsai et al. (2013)[29]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ECDDHP</td>
</tr>
<tr>
<td>Our protocol</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>CMCDHP</td>
</tr>
</tbody>
</table>

✓ means the property is satisfied; ❌ means the property is not satisfied; – means the property is not considered; ECDDHP means Elliptic Curve Computational Diffie-Hellman Problem; CMCDHP means Chaotic-Maps Computational Diffie-Hellman Problem.

<p>| TABLE 4 | Summary of Computational Cost in the Login and Authentication Phase |</p>
<table>
<thead>
<tr>
<th>Protocols</th>
<th>User(smart-card)</th>
<th>Server</th>
<th>Total</th>
<th>CC</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roy et al. (2018)[3]</td>
<td>9Tt + 2Tc</td>
<td>6Tt + Tc</td>
<td>15Tt + 3Tc</td>
<td>960 bits</td>
<td>2</td>
</tr>
<tr>
<td>Islam et al. (2018)[4]</td>
<td>7Tt + 2Tm + Tc</td>
<td>5Tt + 2Tm + Tc</td>
<td>12Tt + 4Tm + 2Tc</td>
<td>768 bits</td>
<td>3</td>
</tr>
<tr>
<td>Wazid et al. (2017)[2]</td>
<td>8Tt + 2Tm</td>
<td>5Tt + 4Tm + 2Tc</td>
<td>13Tt + 6Tm + 2Tc</td>
<td>1140 bits</td>
<td>3</td>
</tr>
<tr>
<td>Liu et al. (2016)[21]</td>
<td>6Tt + 3Tc</td>
<td>6Tt + 3Tc</td>
<td>12Tt + 6Tc</td>
<td>1280 bits</td>
<td>3</td>
</tr>
<tr>
<td>Jiang et al. (2016)[23]</td>
<td>6Tt + 3Tc</td>
<td>5Tt + 3Tc</td>
<td>11Tt + 6Tc</td>
<td>768 bits</td>
<td>2</td>
</tr>
<tr>
<td>Zhu et al. (2015)[22]</td>
<td>4Tt + 2Tm</td>
<td>6Tt + 2Tc</td>
<td>10Tt + 4Tc</td>
<td>736 bits</td>
<td>2</td>
</tr>
<tr>
<td>Tsai et al. (2013)[29]</td>
<td>5Tt + Tm</td>
<td>5Tt + 3Tm</td>
<td>10Tt + 4Tm</td>
<td>960 bits</td>
<td>3</td>
</tr>
<tr>
<td>Our protocol</td>
<td>10Tt + 3Tc</td>
<td>8Tt + 3Tc</td>
<td>18Tt + 6Tc</td>
<td>1376 bits</td>
<td>2</td>
</tr>
</tbody>
</table>

CC means Communication Cost; CR means Number of Communication Message Flows. Some lightweight operations like exclusive-OR and “||” are ignored.
of the ciphertext of the symmetric encryption/decryption algorithm is 128 bits and the bit length of the timestamp is 16 bits. The total communication cost of our protocol is (160 +128+320+160)+(128+160×2+160)=1376 bits. From Table 4, we can observe that existing protocols cost slightly less communication than our protocol. But our protocol only needs two communication message flows, while [2], [4], [21] and [29] need three flows.

However, in summary, the proposed AKA protocol is best suitable for three-factor authentication and key agreement with regard of the trade-off between security and usability for mobile lightweight devices.

6 CONCLUSION

In this paper, we have designed a new chaotic-maps-based AKA protocol by adopting the techniques of “Fuzzy-Verifiers” and “Honeywords”, and then prove its security from formal and evaluation criteria-based security analysis. The security analysis results indicate that the proposed AKA protocol is capable to achieve semantic security and meet the 13 evaluation criteria (EC1-EC13). By comparing the security, computation and communication costs of our protocol with the state-of-the-art protocols, we show that our new protocol is more practical for mobile lightweight devices. Its design ideas are generic and can be used as a guideline to build AKA protocols with a good balance of security and usability. Our future work is to develop concrete three-factor AKA protocols with robust security and high efficiency for scenarios where more than three or four parties are involved (e.g., cloud computing and edge computing environments).

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their invaluable comments. This work was supported by the National Natural Science Foundation of China (No. 61802006, 61897069), National Key Research and Development Program of China (No. 2018YFB0803605), Science and Technology Research Project of Education Department of Jiangxi Province (No. GJJ191680), and Doctoral Foundation of Jiangxi Normal University.

REFERENCES


Shuming Qiu received the PhD degree from the Beijing University of Posts and Telecommunications, China, in 2019. He won the 2018 China Internet Development Foundation Network Security Scholarship. He is currently a lecturer with Jiangxi Normal University, China. His research interests include authentication protocol, lattice cryptography, fully homomorphic encryption, non-communicative cryptography, (semi-)group algebra, etc.

Ding Wang received the PhD degree in information security from Peking University, in 2017. He is currently a professor with the College of Cyber Science, National University of Defense Technology, and also serves as the deputy director of the Tianjin key laboratory of Network and Data Security Technology. As the first author, he has published more than 40 papers at venues like ACM CCS, Usenix Security, NDSS, IEEE DSN, ESORICS, the ACM Transactions on Cyber-Physical Systems, IEEE Transactions on Dependable and Secure Computing, and IEEE Transactions on Information Forensics and Security. Seven of them are recognized as “ESI highly cited papers”. His PhD thesis receives the “ACM China Doctoral Dissertation Award” and “China Computer Federation (CCF) Outstanding Doctoral Dissertation Award”. He has been involved in the community as a TPC member, AEC member, or PC Chair for more than 50 international conferences such as Usenix Security, ACSAC, ACM AsiaCCS, SEC, ISC, and ACISP. His research interests include password, multi-factor authentication, and provable security.

Guoai Xu received the PhD degree in signal and information processing from the Beijing University of Posts and Telecommunications, China, in 2002. He was awarded the Title of Professor in 2011. He is currently an associate director with the National Engineering Laboratory of Security Technology for Mobile Internet, School of Cyberspace Security, Beijing University of Posts and Telecommunications. His research interests include the areas of software security and data analysis.

Saru Kumari received the PhD degree in mathematics from Chaudhary Charan Singh University, Meerut, Uttar Pradesh, India, in 2012. She is currently an assistant professor with the Department of Mathematics, Chaudhary Charan Singh University, Meerut, Uttar Pradesh, India. She has published 190 papers in international journals and conferences including 170 research publications in SCI indexed journals. Her current research interests include information security, digital authentication, security of wireless sensor networks, and applied mathematics.
Practical and Provably Secure Three-Factor Authentication Protocol Based on Extended Chaotic-Maps for Mobile Lightweight Devices (Appendix File)

Shuming Qiu, Ding Wang, Guoai Xu, and Saru Kumari

Abstract—This appendix file consists of one part. In the Appendix A, we simulate the proposed protocol using AVISPA. The simulation results show that the proposed protocol is secure against the active and passive attacks under the Dolev-Yao model.

Index Terms—Appendix file.

APPENDIX A: SECURITY VERIFICATION OF THE PROPOSED PROTOCOL USING AVISPA

In this Appendix, we simulate the proposed protocol using AVISPA. AVISPA (Automated Validation of Internet Security Protocols and Applications) is a button software tool and widely accepted for automatically validating Internet security sensitive protocols and applications [1]. AVISPA supports high-level protocol specification language called HLPSL, which is usually used to provide formal security verification for the simulated protocol. The simulation results of AVISPA can verify whether the simulated protocol is secure against active and passive attacks. Now, we first describe the proposed protocol using HLPSL, and then execute the HLPSL specifications using SPAN. Lastly, the simulation results are presented.

HLPSL Specification of the Proposed Protocol

In this section, we translate the protocol into HLPSL language, and provide the standardized descriptions of user, server, session, environment and goal roles under HLPSL language (See Section I, Section II and Section for details). In the simulation, the following eight security goals and two authentication properties were verified.

Goal 1: The secrecy_of subs1 means that ID_i is kept secret to (U_i, S).

Goal 2: The secrecy_of subs2 means that PW_i, u and \sigma_i are kept secret to U_i only.

Goal 3: The secrecy_of subs3 means that B_0 is kept secret to (U_i, S).

Goal 4: The secrecy_of subs4 means that random key parameters \{C_2, C_6\} are kept secret to (U_i, S).

Goal 5: The secrecy_of subs5 means that the negotiated session key SK_u is known by (U_i, S).

Goal 6: The secrecy_of subs6 means that the secret key s and the secret parameter r_i are permanently kept secret, known to only S.

Goal 7: The secrecy_of subs7 means that the the random number v is only known to S.

Goal 8: The secrecy_of subs8 means that the negotiated session key SK_v is known by (U_i, S).

Authentication Property 1: The authentication_on alice_server_c2 means that U_i generates u and compute C_2. If S can computes C_2 by making use of the login request message C_1 and successfully verify C_i, then it authenticates U_i.

Authentication Property 2: The authentication_on server_alice_c6 means that S generates v. If U_i can compute C_6 by making use of the respond message C_5 and successfully verify C_6, then it authenticates S.

Role specification of U_i in HLPSL

\[
\text{role alice(U_i,S:agent,SKa:symmetric_key,}
\begin{align*}
\text{SKa:}&\text{symmetric_key},H,Chao,REP:hash\_func,} \\
\text{Snd, Rcv: channel(dy)}
\end{align*}
\]

played_by U_i
def- local State: nat, ID_i, PW_i, VPW_i, B0, SIG_i, TAUi, N0, CID_i, B0, Bi, TS_i, Wi, UI_i, C2, C6, CIDIN, BON: text, C1, C3, C4, C5, C7, C8, SKu: message, Inc: hash\_func \const \text{alice_server, server_alice, subs1, subs2, subs3, subs4, subs5, subs6, subs7, subs8: protocol_id}

init State=0

transition 1. State=0/Rcv\(\text{start})=\rightarrow

State’:=1

/ID_i’:=new()/Snd\{(ID_i’, SKas)

/secret\{(ID_i), subs1, UI_i, S)\}

2. State=1/Rcv\{(TS_i’, X_i’.CID_i’, B0’) _SKsa)=\rightarrow
Role server(S, Ui:agent, SKas:symmetric_key, SKsa:symmetric_key, H, Chao, GEN, REP:hash_func) played_by S def= local State:nat, SS,Ri,CIDi,Idi,Ei,B0,Ts,X,C2,C6,ElN,CIDiN, B0N:test,C1,C3,C4,C5,C7,C8,SKs:message, Inc:hash_func const alice_server,server_alice,subs1,subs2, subs3,subs4,subs5,subs6,subs7,subs8:protocol_id init State:=0 transition 1.State=0/Rcv({Idi'}_SKas)=|> State'=1 /\Ei':=new() /\Ri':=new() /\CIDi':=H(Idi'.Ei') /\B0':=H(Idi'.SS.Ri'.CIDi') /\Ts':=new() /\X':=new() /\Snd({Ts'.X'.CIDi'.B0'}_SKsa) /\secret({SS,Ri'},subs6,S) 2.State=1/Rcv(CIDi'.C1'.C3'.C4')=|> State'=2 /\C2':=Chao(Ts'.C1')/\CIDi':=xor(C3',C2') /\CIDiN':=H(Idi'.ElN') /\B0N':=H(Idi'.SS.Ri'.CIDiN') /\VS':=new() /\C5':=Chao(VS'.X')/\C6':=Chao(C5'.C1') /\C7':=xor(B0N',H(C6' .B0)) /\SKs':=H(Idi'.CIDi'.B0'.B0N'.C2'.C6') /\C8':=H(Idi'.CIDi'.B0'.B0N'.C2'.SKs') /\witness(S,Ui,server_alice_c6,C6') /\secret({VS'},subs7,S) /\secret({SKs'},subs8,{Ui,S}) /\Snd({C5'.C7'.C8'}) end role

Simultaneous公社 of S in HLPSL

Role specification of S in HLPSL

Role server(S, Ui:agent, SKas:symmetric_key, SKsa:symmetric_key, H, Chao, GEN, REP:hash_func) played_by S def= local State:nat, SS,Ri,CIDi,Idi,Ei,B0,Ts,X,C2,C6,ElN,CIDiN, B0N:test,C1,C3,C4,C5,C7,C8,SKs:message, Inc:hash_func const alice_server,server_alice,subs1,subs2, subs3,subs4,subs5,subs6,subs7,subs8:protocol_id init State:=0 transition 1.State=0/Rcv({Idi'}_SKas)=|> State'=1 /\Ei':=new() /\Ri':=new() /\CIDi':=H(Idi'.Ei') /\B0':=H(Idi'.SS.Ri'.CIDi') /\Ts':=new() /\X':=new() /\Snd({Ts'.X'.CIDi'.B0'}_SKsa) /\secret({SS,Ri'},subs6,S) 2.State=1/Rcv(CIDi'.C1'.C3'.C4')=|> State'=2 /\C2':=Chao(Ts'.C1')/\CIDi':=xor(C3',C2') /\CIDiN':=H(Idi'.ElN') /\B0N':=H(Idi'.SS.Ri'.CIDiN') /\VS':=new() /\C5':=Chao(VS'.X')/\C6':=Chao(C5'.C1') /\C7':=xor(B0N',H(C6' .B0)) /\SKs':=H(Idi'.CIDi'.B0'.B0N'.C2'.C6') /\C8':=H(Idi'.CIDi'.B0'.B0N'.C2'.SKs') /\witness(S,Ui,server_alice_c6,C6') /\secret({VS'},subs7,S) /\secret({SKs'},subs8,{Ui,S}) /\Snd({C5'.C7'.C8'}) end role

Specification of the session, environment, and goal in HLPSL

Role session(S, Ui:agent, SKas:symmetric_key, SKsa:symmetric_key, H, Chao, GEN, REP:hash_func) def=local SI,SJ,RI,RJ:channel(dy) composition alice(Ui,S,SKas,SKsa,H,Chao,GEN,REP,SI,RI) /\server(Ui,S,SKas,SKsa,H,Chao,GEN,REP,SI,RI) end role

Simulation Results

In order to test the security of the proposed protocol by using of SPAN (Security Protocol ANimator for AVISPA) [2], our experimental environment is Inter(R) Core(TM) i5-5200U CPU@2.20GHz, RAM8.00+4.00GB, SPAN-Ubuntu 10.10-light, Ubuntu (32-bit), virtual machine memory 2GB. Since the OFMC (On-the-Fly-Model-Checker) and CL-ATSe (Constraint-Logic-based Attack Searcher) models accept Diffie Hellman and XOR operations, we use them to simulate the proposed protocol. After the implementation of OFMC and CL-ATSe, the simulation results of the proposed protocol are shown in Figs. 1 and 2. It can be observe that in the OFMC model, the search depth is 4, 18
nodes are accessed, the analysis time is almost 0 second, the search time is 0.63 second, and the simulation result shows SAFE; in the CL-ATSE model, two states are analyzed, where the translation takes 0.16 second and the computation time is almost 0 second, and the simulation result shows SAFE. Therefore, the simulation results show that the proposed protocol is secure against the active and passive attacks under the Dolev-Yao model [3].

REFERENCES

