A privacy-preserving multi-factor authenticated key exchange protocol with provable security for cloud computing

Fushan Wei¹ ², Qi Jiang³, Jian Shen⁴, Ruijie Zhang¹, Chuangui Ma⁵

¹ State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China
² State Key Laboratory of Cryptology, P. O. Box 5159, Beijing, China
³ School of Computer Science and Technology, Xidian University, Xi’an, China
⁴ School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China
⁵ Department of Basic, Army Aviation Institution, Beijing, China

Abstract. In the past decade, cloud computing has grown from being a promising business concept to one of the fast growing segments of the information technology’s industry. More and more information of individuals and companies are stored in the cloud. However, the openness of the network and the uncontrollability of the cloud computing environment bring serious threats to user’s sensitive data and personal privacy. Security and privacy issues pose as the key roadblock to its fast adoption. In order to address these issues, we present an efficient MFAKE protocol based on fuzzy extractors in this paper. The novel protocol can be proven secure in the random oracle model and achieves truly multi-factor security. Compared with other related MFAKE protocols, our protocol not only enjoys stronger security, but also has greater efficiency both in terms of computation and communication. The proposed MFAKE protocol provides high-level security and protects user’s privacy, so it is particular suitable for cloud computing environments.

Key words: Multi-Factor Key Exchange; Password; Cloud Computing; Biometrics; Fuzzy Extractors.

1 Introduction

1.1 Backgrounds

With the rapid development of the cloud computing technologies, users can conveniently enjoy the ubiquitous services via mobile devices [1–4]. However, the openness of the network and the uncontrollability of the computing environment bring serious threats to user’s sensitive data and personal privacy [5]. Access control and user privacy become main concerns in cloud computing environments. In order to protect user’s data from unauthorized access, user authentication is a pre-requisite for almost all secure communication systems. The adversary is
rather powerful in the cloud computing environment. Consequently, strongly secure authentication mechanism is needed under this circumstances [6–8]. Meanwhile, it is also desirable to protect user’s privacy during the authentication process.

Generally speaking, user authentication is based on three types of authentication factors which are related to the user. The first type is something the user knows (typically a human memorizable password); The second type is something the user has (e.g., a smart card or a secure token that stores a private key); The third type is something the user is (e.g., biometric characteristics). Unfortunately, each authentication factor has its own inherent weaknesses which cannot be solved using cryptographic techniques only [9]. The adversary can steal a user’s password through phishing attacks or spywares. Ordinary users are totally unaware of these attacks even though their passwords are stolen; A smart card or a secure token can be lost or stolen. The adversary can open and clone the device of the victim user unless the device uses expensive tamper-resistant techniques; Biometric characteristics may be easily copied and they are usually unchangeable. Since the biometric capture devices are remotely located, the authentication server cannot correctly detect whether the biometric characteristics come from an alive user or not.

To enhance the security of the authentication schemes, many multi-factor authenticated key exchange (MFAKE) protocols, which combines two or even more authentication factors, are proposed in recent years [10–20]. The basic idea of MFAKE protocols is that the adversary has to compromise all the authentication factors in order to breach the security of the MFAKE protocol. However, as is noted in [21], it is a non-trivial hard work to design even a secure single-factor based authenticated key exchange (AKE) protocol. Designing a truly secure MFAKE protocol can only be harder and is incredibly difficult. Great caution should be taken when designing MFAKE protocols.

After twenty years of research, the literature about two-factor AKE protocols is rich of many results. However, three-factor AKE protocols has so far received scant attention. The approach of designing three-factor AKE protocols is far from maturity and satisfactory. Much research is needed to improve the security of three-factor AKE protocols. In this paper, we focus on the design of three-factor AKE protocols.

1.2 Related work

In 2006, Spantzel et al. [22] proposed an elegant privacy-preserving three-factor authentication scheme based on a password, a secure token and a biometric key. They used a zero-knowledge proof scheme to prove the possession of biometric key. However, their protocol does not have rigorous security proof. In 2008, Pointcheval et al. [10] defined the first formal security model for MFAKE protocols. They also presented a protocol which is provably secure in the random oracle model. Their work is the basis of many subsequent research. In 2009, Fan et al. [23] provided a three-factor authentication scheme which combines biometrics with passwords and smart cards. Their protocol heavily relies on public
key encryption and thus is inefficient. In 2010, Stebila et al. [24] introduced a
security model for MFAKE protocols and proposed an efficient MFAKE protocol
which combines passwords, one-time passwords and biometric keys. Meanwhile,
Liu et al. [25] extended Pointcheval et al.’s MFAKE protocol to the three-party
setting. In 2011, Huang et al. [26] proposed a generic framework which can com-
bine a secure two-factor smart-card-based password authentication scheme with
biometrics to produce a three-factor authentication scheme. Their main techni-
cal tool is the fuzzy extractor introduced in [27]. In 2012, Hao et al. [21] showed
that Pointcheval et al.’s MFAKE protocol is vulnerable to an adversary who only
compromises the password factor. In other words, the adversary only needs to
compromise a single password factor in order to break down the entire system.
Hao et al.’s attack also applies to Liu et al.’s MFAKE protocol [25], which is the
three-party extension of Pointcheval et al.’s MFAKE protocol. In 2014, Yu et
al. [28] introduced an efficient generic framework for three-factor authentication
using fuzzy vaults. Their generic framework can be viewed as an improvement
of Huang et al.’s work [26]. Huang et al. [29] presented a robust generic multi-
factor authentication protocol for fragile communications. They also put forward
the notion of stand-alone authentication which is of independent interest. He et
al. [30] proposed a MFAKE protocol for multi-server environment using elliptic
curve cryptography. Their protocol is the first three-factor authenticated scheme
for multi-server environment. Recently, Fleischhacker et al. [31] propose a mod-
ular framework for MFAKE protocols by mixing multiple types and quantities
of authentication factors in a secure way. Although their modular framework is
attractive, it is complex and very inefficient.

1.3 Motivation and contribution

Most of the existing MFAKE protocols are based on passwords, smart cards
and biometric keys, Pointcheval et al.’s protocol [10] is the only provably secure
MFAKE protocol which combines a password, a high-entropy cryptographic key,
and a biometric key. Their MFAKE protocol is very famous and lays the basis for
many MFAKE protocols. Unfortunately, Pointcheval et al.’s MFAKE protocol
has the following disadvantages. First of all, Hao et al. [21] showed Pointcheval
et al.’s MFAKE protocol is insecure against an adversary that only corrupts
a user’s password factor. The adversary can impersonate the server with the
user’s password and compromise all other authentication factors. Hao et al.
found that the attack is a fundamental problem of the protocol, which seems
not easily fixable within the current structural design of the protocol. Secondly,
Pointcheval et al.’s protocol only considers unilateral client-authentication and
does not achieve mutual authentication. Lack of mutual authentication opens
the door for Hao et al.’s attack. Thirdly, Pointcheval et al. use the ElGamal
encryption scheme to encrypt each bit of user’s biometric template. Usually,
user’s biometric template is encoded on a 1024-bit string. As a result, their
protocol is very inefficient in terms of both computation and communication.
Last but not least, Pointcheval et al.’s protocol needs 4 rounds though it does
not provide mutual authentication. However, mutual authentication actually can be achieved by 3 rounds communications.

In order to overcome the disadvantages of Pointcheval et al.’s MFAKE protocol, we propose a novel three-factor authenticated key exchange protocol using fuzzy extractors. Our basic idea is as follows. The user generates a biometric key $R_C$ by the generation algorithm of the fuzzy extractor using his biometric template $W_C$ as input. The biometric key $R_C$ is stored in the server’s database along with a public string $P_C$. The public string can be used to reconstruct $R_C$ using another template $W_C'$, as long as $W_C'$ and $W_C$ are close enough. The privacy of user’s biometric is preserved because the authentication server does not have to know the user’s real biometric template. In the authentication phase, the user and the server run a two-party EKE protocol \cite{32} to exchange the Diffie-Hellman key materials. The session key is computed by using a special version of the HMQV protocol \cite{33} in which a party wants to establish a secure channel with itself. In such a case, the Diffie-Hellman key materials can be viewed as ephemeral keys and and $R_C$ can be viewed as the long-term key. The session key is established by combining the password factor and the biometric factor. The public-private key pairs are used to sign a key confirmation value of the session key. In this way, the signature can only be verified by someone who knows the session key. This trick strengthens the security of the protocol. In summary, we use the password factor and the biometric factor to establish the session key. The high-entropy cryptographic key is used to sign the authentication value of the session key. Compared with Pointcheval et al.’s MFAKE protocol, our protocol provides truly three-factor security and can resist Hao et al.’s attack. Most notably, our protocol is very efficient in terms of computation and communication. Moreover, our protocol achieves mutual authentication with 3 rounds communications. Consequently, our protocol not only enjoys stronger security, but also has higher efficiency.

The rest of the paper is organized as follows. We recall some building blocks and the security model for MFAKE protocol in Section 2. In Section 3, we present the proposed MFAKE protocol. We compare the efficiency and security features of our protocol with Pointcheval et al.’s MFAKE protocol in Section 4. We discuss some design rationales in Section 5. We conclude our paper in Section 6. The security proof of the proposed protocol is conducted in the random oracle model in Appendix A.

2 Preliminaries

2.1 Fuzzy extractors

The notion of fuzzy extractors was introduced by Dodis et al. \cite{27} in 2004 as a mechanism to extract a biometric key from a biometric template $W$ and later reproduce it using another biometric template $W'$ close to the original $W$. We review some basic definitions related to fuzzy extractors from \cite{34}. 

4
Definition 1 (metric space) A metric space \((M, d)\) is a finite set \(M\) equipped with a symmetric distance function \(d : M \times M \rightarrow \mathbb{Z}^+ \cup \{0\}\) such that the triangle inequality holds and \(d(x, y) = 0 \iff x = y\).

The format of the biometric data is assumed to form a metric space under some appropriate distance function. The hamming distance is the most commonly used distance function. There is no need to specify any particular metric space because a variety of metrics are good enough for our purpose.

Definition 2 (statistical distance) Let \(X_1\) and \(X_2\) be two probability distributions over some set \(S\). The statistical distance between \(X_1\) and \(X_2\) is defined as \(SD(X_1, X_2) = \frac{1}{2} \sum_{s \in S} |Pr[X_1 = s] - Pr[X_2 = s]|\). If two distributions \(X_1\) and \(X_2\) have statistical distance at most \(\varepsilon\), we say they are \(\varepsilon\)-close.

Definition 3 (min-entropy) The min-entropy of a probability distribution \(X\) is defined as \(H_\infty(X) = -\log(\max_x Pr[X = x])\).

Definition 4 (fuzzy extractors) An \((m, l, t, \varepsilon)\) fuzzy extractor over a metric space \((M, d)\) is composed of a pair of efficient randomized algorithms \((\text{Gen}, \text{Rep})\) such that:

1. The generation algorithm \(\text{Gen}\) takes as input \(W \in M\) and outputs an extracted string \(R \in \{0, 1\}^l\) and a helper string \(P \in \{0, 1\}^*.\) The reproduction algorithm \(\text{Rep}\) takes as inputs an element \(W' \in M\) and a string \(P \in \{0, 1\}^*\), and returns a string in \(\{0, 1\}^l\).
2. Correctness: If \(d(W, W') \leq t\) and \((R, P)\) is generated by \(\text{Gen}(W)\), then \(\text{Rep}(W', P) = R\).
3. Security: For all distributions over \(M\) with min-entropy \(m\), the statistical distance between \(R\) and uniform distribution is at most \(\varepsilon\) even conditioned on \(P\).

2.2 Security model

In this subsection, we extend the security model for MFAKE protocol proposed in [10], which in turn builds upon the models from [35, 36].

Protocol participants. Participants in an MFAKE protocol are either clients \(C\) or a unique authentication server \(S\). Each participant can activate several instances at a time and run several sessions concurrently. An instance of participant \(U\) in session \(i\) is denoted as \(\Pi_i^U\). We define the session identity \(sid_i^U\) as the transcript of all messages sent and received by the instance \(\Pi_i^U\), except for the last protocol message. By \(\text{pid}_i^U\), we denote the partner identity with which the instance \(\Pi_i^U\) is interacting in the protocol session. We also define a boolean variable \(\text{acc}_i^U\), which is determined at the end of the session and denotes whether the instance \(\Pi_i^U\) accepts the session or not.

Partnering. The two instances \(\Pi_i^U\) and \(\Pi_i^{U'}\) are said to be partners if the following conditions are fulfilled: (1) \(\text{acc}_i^U = \text{acc}_{i'}^{U'} = 1\); (2) \(\text{pid}_i^U = U'\) and \(\text{pid}_{i'}^{U'} = U\); (3) \(\text{sid}_i^U \neq \text{sid}_{i'}^{U'}\).
**Long-lived keys.** Each client $C$ has a tuple $t_C = (W_C, sk_C, pw_C)$, where $W_C$ is a probability distribution for his biometric, while $sk_C$ and $pw_C$ are a high-entropy cryptographic key and a low-entropy password respectively. The authentication server holds a list of tuples $t_S = \langle t_S[C] \rangle$ with an entry for each client, where $t_S[C]$ is a transformed-tuple of $t_C$. We also assume the authentication server has its own high-entropy cryptographic key $sk_S$ with the corresponding system-wide known public key $pk_S$.

**Adversarial queries.** The adversary $A$ is modeled as a probabilistic polynomial time (PPT) adversary. We assume $A$ controls the communication channel between the client and the authentication server. The abilities of $A$ are modeled through oracle queries:

- $\text{Execute}(\Pi^C, \Pi^S)$: This query models passive eavesdropping of a protocol execution between a client instance $\Pi^C$ and a server instance $\Pi^S$. At the end of the execution, a transcript is given to the adversary, which logs everything an adversary could see during the execution.
- $\text{Send}(\Pi^U, m)$: This query models an active attack against instance $\Pi^U$. The adversary $A$ sends message $m$ to the protocol instance $\Pi^U$, $\Pi^U$ executes as specified by the protocol and sends back its response to the adversary.
- $\text{Reveal}(\Pi^U)$: This query causes the output of session key held by the instance $\Pi^U$. This query allows the adversary learn past session keys from previous executions, modeling the known key attack.
- $\text{Corruption}(S)$: This query allows the adversary learn the server’s secret key $sk_S$ and the list of tuples $t_S = \langle t_S[C] \rangle$. The corrupted server $S$ is under the control of the adversary.
- $\text{Corruption}(C)$: there are three different corruption queries to the client:
  (a) $\text{Corruption}(C, pw_C, sk_C)$: This query enables the adversary to get the client $C$’s password $pw_C$ and secret key $sk_C$;
  (b) $\text{Corruption}(C, pw_C, W_C)$: This query enables the adversary to get the client $C$’s password $pw_C$ and the biometric template $W_C$;
  (c) $\text{Corruption}(C, sk_C, W_C)$: This query enables the adversary to get the client $C$’s secret key $sk_C$ and the biometric template $W_C$;
- $\text{Test}(\Pi^U)$: This query does not captures $A$’s real attack ability. It measures the session key security of instance $\Pi^U$. The simulator flips a coin, if the random bit is $b = 1$, then he sends the real session key to the $A$. Otherwise, the simulator sends a random key of the same size to $A$. $A$ will guess the value of $b$ to win the game.

**Freshness.** The freshness notion captures the intuitive fact that a session key is not trivially known to the adversary. A client $C$ is fully corrupted if and only if all the authentication factors of $C$ have been corrupted. For example, a client $C$ is fully corrupted if an adversary asks more than one corruption query to the client. A server $S$ is fully corrupted if and only if a $\text{Corruption}(S)$ query has been asked. We say that the session key of an instance $\Pi^U$ is fresh if the following conditions hold:

1. Upon acceptance of the instance $\Pi^U$, neither the participant $U$ nor its intended partner is fully corrupted;
(2) No Reveal query has been sent to either $\Pi_i$ or its partner (if such instance exists).

Note that the above definition of freshness captures the security property of forward security. Because after the instance accepts, even if the client or the server is fully corrupted, the instance is still fresh.

**Session key security.** Suppose $\mathcal{P}$ is an MFAKE protocol and $\mathcal{A}$ is a probabilistic polynomial time (PPT) adversary against session key security of $\mathcal{P}$. The adversary $\mathcal{A}$ is given oracle access to $\text{Execute, Send, Reveal, Corrupt and Test}$ oracles. However, $\mathcal{A}$ can only make Test query on fresh instances. If the adversary can correctly guess the random bit $b$ used by the Test oracle, then we say the adversary succeeds in this attack game. We denote this event by $\text{Succ}$.

**Definition 5** The advantage of an adversary $\mathcal{A}$ in violating the MFAKE semantic security of the protocol $\mathcal{P}$, is defined as

$$\text{Adv}_{\mathcal{P}}^{\text{mFAKE}}(\mathcal{A}) = 2 \cdot \Pr[\text{Succ}] - 1.$$ 

The advantage function of the protocol $\mathcal{P}$ is defined as

$$\text{Adv}_{\mathcal{P}}^{\text{mFAKE}}(t, R) = \max\{\text{Adv}_{\mathcal{P}}^{\text{mFAKE}}(\mathcal{A})\}$$

where maximum is over all $\mathcal{A}$ with time-complexity at most $t$ and using resources at most $R$ (such as the number of oracle queries).

**Definition 6** A MFAKE protocol $\mathcal{P}$ is said to be semantically secure if the advantage $\text{Adv}_{\mathcal{P}}^{\text{mFAKE}}(t, R)$ is only negligibly larger than $kq_{\text{send}}/|\mathcal{D}|$, where $q_{\text{send}}$ is the number of Send queries, $\mathcal{D}$ is the dictionary space and $k$ is a constant.

**Authentication security.** A MFAKE protocol should provide authentication security to verify the validity of a participant. Informally speaking, authentication security ensures that an adversary should not be able to impersonate a participant unless the participant is fully corrupted.

We say an adversary $\mathcal{A}$ breaks client authentication if there exists a server instance $\Pi_i^S$ that has accepted with partner identity $C$, but there exists no client instance $\Pi_j^C$ that is partnered with $\Pi_i^S$ and the client $C$ is not fully corrupted. Let $\text{Succ}_{\text{MFAKE,CAuth}}(\mathcal{A})$ denote the adversary $\mathcal{A}$’s success probability in breaking client authentication.

**Definition 7** A MFAKE protocol achieves client authentication security if for all PPT adversary $\mathcal{A}$, the success probability $\text{Succ}_{\text{MFAKE,CAuth}}(\mathcal{A})$ is only negligibly larger than $kq_{\text{send}}/|\mathcal{D}|$, where $q_{\text{send}}$ is the number of Send queries, $\mathcal{D}$ is the dictionary space and $k$ is a constant.

Likewise, we can define server authentication security. We omit it for simplicity. We say a MFAKE protocol provide mutual authentication if it achieves both client authentication security and server authentication security. Note that the definition of authentication security in our extended model is a strong one.
because KCI (Key Compromise Impersonation) attack is captured by our definition. More precisely, the adversary is allowed to fully corrupt the server and then impersonate a client to the server. The only restriction is that the client is not fully corrupted. If a server instance $H_S^k$ accepts in this case, the adversary is successful in breaking client authentication.

3 Description of the Protocol

In this section, we propose an efficient multi-factor authenticated key exchange protocol using fuzzy extractors. Our protocol is based on a cyclic group with parameters $(p, q, g)$. $p$ and $q$ are two big primes such that $q \mid p - 1$. Let $G_q$ be a subgroup in $Z_p^*$ with prime order $q$, and $g$ be a generator of $G_q$. Let $u$ and $v$ be two random elements in $G_q$.

In the registration phase, a client $C$ chooses his password $pw_C$ from the dictionary space $D$. $C$ also imprints his biometric template $W_C$ and computes $Gen(W_C) = (R_C, P_C)$ using the generation algorithm of the fuzzy extractor, where $R_C$ is a private string and $P_C$ is a public string. $R_C$ can be treated as a biometric key. Without loss of generality, we assume $pw_C, R_C \in Z_q^*$. $C$ sends $(pw_C, R_C, P_C)$ to the server $S$ through a secure channel. $C$ has his private-public key pair $(sk_C, pk_C)$ for signature and $S$ also has its private-public key pair $(sk_S, pk_S)$ for signature. For simplicity, we assume $S$ stores all clients’ public keys in its database and the public key of the server is known system-wide. Thus, we can omit public key credential exchange and verification in the key exchange phase.

Detailed steps of the key exchange phase of the proposed protocol, as shown in Fig. 1, are described as follows:

**Round 1.** The client $C$ randomly chooses $x \in Z_q^*$, compute $X = g^x$ and $X^\ast = X \cdot u^{pw_C}$. Finally, $C$ sends $(C, X^\ast)$ to the server $S$.

**Round 2.** Upon receiving the message $(C, X^\ast)$, $S$ chooses a random number $y \in Z_q^*$, then computes $Y = g^y$, $Y^\ast = Y \cdot v^{pw_C}$ and $X' = X^\ast/u^{pw_C}$. $S$ also computes a secret value $K_S = (X^\ast \cdot g^{dR_C})(y + dR_C)$, where $e = H(X^\ast, S)$ and $d = H(Y^\ast, C)$. Next, $S$ computes $\alpha = H_1(C, S, X^\ast, Y^\ast, P_C, R_C, K_S)$ and signs $\alpha$ using his private key $sk_S$ to generate the signature $\sigma_S$. Finally, $S$ sends the message $(S, Y^\ast, P_C, \sigma_S)$ back to the client $C$.

**Round 3.** Upon receiving the message $(S, Y^\ast, P_C, \sigma_S)$, $C$ imprints another biometric template $W_C$ and computes $R_C = Rep(W_C, P_C)$ using the reproduction algorithm of the fuzzy extractor. $C$ also computes the secret value $K_C = (Y^\ast \cdot g^{dR_C})(x + eR_C)$, where $e = H(X^\ast, S)$ and $d = H(Y^\ast, C)$. $C$ computes $\alpha = H_1(C, S, X^\ast, Y^\ast, P_C, R_C, K_C)$ and verifies the validity of the signature $\sigma_S$. If it is valid, then $C$ computes $\beta = H_2(C, S, X^\ast, Y^\ast, P_C, \sigma_S, R_C, K_C)$ and signs $\beta$ using his private key $sk_C$ to generate the signature $\sigma_C$. Finally, $C$ computes the common session key $sk = H_3(C, S, X^\ast, Y^\ast, P_C, \sigma_S, R_C, K_C)$, sends the message $(C, \sigma_C)$ to $S$ and accepts the session.

Upon receiving the message $(C, \sigma_C)$, $S$ computes $\beta = H_2(C, S, X^\ast, Y^\ast, P_C, \sigma_S, R_C, K_S)$ and verifies the validity of the signature $\sigma_C$. If the verification is success-
Fig. 1. Our Multi-Factor Authenticated Key Exchange Protocol

ful, then $S$ also computes the session key $sk = H_3(C, S, X^*, Y^*, P_C, \sigma_S, \sigma_C, R_C, K_S)$ and accepts the session.

4 Performance Analysis

To the best of knowledge, Pointcheval et al.’s MFAKE protocol [10] is the only MFAKE protocol in the literature which combines passwords, high-entropy keys and biometrics. So we compare the efficiency and security features of the proposed protocol with Pointcheval et al.’s MFAKE protocol in this section.

With respect to computation, we only consider the number of modular exponentiations since these are the most expensive type of computation. Note that the product of two modular exponentiations in the formulation of $g^x h^y$ can be computed at the cost of a single modular exponentiation (see Alg 15.2 in [39]).

The computation cost of the fuzzy extractor [27] is no more than the cost of several hash function operations, so we omit it for simplicity. We instantiate the signature scheme used in our protocol using the well-known Schnorr signature scheme [37]. In terms of communication, the length of the identifications is assumed to be 32 bits, an element in $G_q$ is assumed to be 160 bits, the output size of secure hash functions is 160 bits and the length of the helping data $P_C$ in the fuzzy extractor is about 62464 bits [38]. According to [37], the length of the signature is roughly 320 bits (the length of the hash function is 160 bits and $q \geq 2^{160}$). The length of the biometric template is 1024 bits and the length of the authentication tag is assumed to be 4 bits in Pointcheval et al.’s MFAKE protocol. The comparison of computation cost and communication cost of the proposed protocol and Pointcheval et al.’s protocol is presented in Table 1. We
can see from Table 1 that our protocol is very efficient both in terms of computation and communication. Particularly, the computation cost of protocol is negligible compared with that of Pointcheval et al.’s protocol.

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Computation Cost</th>
<th>Bandwidth</th>
<th>Message Rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>PZ08 [10]</td>
<td>1026E 2050E</td>
<td>172416bits</td>
<td>4</td>
</tr>
<tr>
<td>Our protocol</td>
<td>4E 4E</td>
<td>63520bits</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 2 summarizes security features of the proposed protocol compared with Pointcheval et al.’s protocol [10]. Both protocols are proven secure in the random oracle model under the CDH assumption. However, we can see from Table 2 that our protocol provides more security features than Pointcheval et al.’s protocol does. Their protocol does not provide mutual authentication. Due to Hao et al.’s attack [21], their protocol does not achieve three-factor security either. It is claimed that biometric privacy is preserved in Pointcheval et al.’s protocol because the server only stores an ElGamal encryption of the reference biometric template. Unfortunately, Hao et al.’s attack showed that the server is able to discover the biometric template with the password factor. Moreover, our protocol is secure against key compromise impersonation (KCI) attack. The only disadvantage of our protocol is that we assume the biometric template is private. However, the biometric template is assumed to be public in Pointcheval et al.’s protocol, which is more general and realistic. Nevertheless, it should be noted that it is more difficult to design a MFAKE protocol when the biometric data is assumed to be public.

5 Discussion

Previous solutions of MFAKE protocols combine all the factors to generate a strongly secure session key and then use the session key to realize mutual authentication between the client and the server. All the factors are aiming at computing a common secret. However, it turns out this is not a good way to provide three-factor security. In the proposed protocol, we use a different paradigm. Only the password factor and the biometric factor are used to generate the session
key, while the high-entropy cryptographic key is used to sign the key confirmation value. From the performance analysis, we can see this technique greatly strengthens the security of the protocol. We believe it is a good idea that different factors should play different roles in MFAKE protocol.

We use a variant of the famous HMQV protocol [33] to compute the secret value $K_S$. The biometric key $R_C$ is treated as the long-term key while the Diffie-Hellman key materials are viewed as the ephemeral keys. Note that the knowledge of the biometric key is proven by the input of the hash function $H_1$ rather than the computing of the session key. However, we still compute the session key in this way because we think the password factor is better protected without bringing much computation cost in this manner. Unlike previous solutions, the participants in our protocol sign on the authenticators $\alpha, \beta$ rather than on some public message. This trick kills two birds with one stone because it not only proves the identity of the signer but also provides key confirmation.

Note that biometrics are assumed to be fully public in Pointcheval et al.’s protocol because they think it is not reasonable in practice to treat the biometric data as private [10]. As a result, Their protocol resorts to the liveness assumption, which basically states that all computations involving the biometric template are done honestly. To model the liveness assumption, they introduced a computation oracle $\text{Compute}$ which requires the adversary to query the computation oracle whenever computing biometric-related messages. However, the liveness assumption is controversial. The adversary is allowed to control the communicating channel in authenticated key exchange protocols. Even if it is required by the liveness assumption that all biometric-related messages are computed by the computation oracle, the adversary still can modify the biometric-related messages during transmission. This actually invalidates the liveness assumption. Nevertheless, it should be noted that it is more difficult to design a MFAKE protocol when the biometric data is assumed to be public. How to design a secure MFAKE protocol when the biometric data is assumed to be public is still an open problem. More research on this problem is needed.

6 Conclusion

In this paper, we design a simple and privacy-preserving MFAKE protocol using fuzzy extractors. The proposed protocol is proven secure in the random oracle model based on the hardness of the CDH problem. Security and performance comparisons show that our protocol achieves both higher efficiency and stronger security. To the best of our knowledge, this is the most efficient MFAKE protocol with provable security which combines passwords, high-entropy cryptographic keys and biometric templates.

7 Acknowledgements

The authors would like to thank the anonymous referees for their helpful comments. the National Natural Science Foundation of China (Nos. 61309016, 61379150,
61501515), Postdoctoral Science Foundation of China (No. 2014M562493), Postdoctoral Science Foundation of Shanxi Province, Natural Science Basic Research Plan in Shanxi Province of China (Program No.2016IM6005), Fundamental Research Funds for the Central Universities (Program No. JB161501), and Key Scientific Technological Project of Henan Province (No. 122102210126,092101210502).

References


A Security Proof

In this section, we present the security proof of our protocol within the security model given in subsection 2.2.

**Theorem 1.** Let $\mathcal{P}$ be the MFAKE protocol in Figure 1. Assume the signature scheme used in our protocol is existentially unforgeable against adaptive chosen message attacks and the fuzzy extractor in our protocol is an $(m, l, t, \varepsilon)$ fuzzy extractor. Let $\mathcal{A}$ be an adversary makes $q_{\text{send}}$, $q_{\text{send}} \leq |\mathcal{P}|$, queries of type Send to different instances. Then under the CDH assumption, the adversary’s advantage in attacking the semantic security of the proposed protocol is bounded by

$$\text{Adv}^{\text{ake}}_{\mathcal{P}, \mathcal{D}}(\mathcal{A}) \leq \frac{q_{\text{send}}}{|\mathcal{P}|} + \operatorname{negl}(l).$$

**Proof.** We prove the theorem through a sequence of experiments. These experiments begins from the real attack. We gradually change the simulation rules over these experiments. In the last experiment, the session keys of accepted sessions are randomly chosen and all the active attack sessions are terminated without accepting (except for the sessions in which the adversary only faithfully relays the messages). Thus, the adversary cannot distinguish the session keys from random numbers in the last experiment. Moreover, the adversary has no advantage in breaking client authentication or server authentication because all these impersonated sessions are terminated without accepting. For each experiment $Exp_i$, we define the following events:

1. $S^{\text{ake}}_i$ (for session key security): this event occurs if the adversary correctly guesses the hidden bit $b$ used by the Test oracle;
- $A_i^{CAuth}$ (for client authentication): this event occurs if the adversary impersonates a client who is not fully corrupted and makes a server instance accepts a session with the impersonated client, but there is no partner instance of the client.

- $A_i^{SAuth}$ (for server authentication): this event occurs if the adversary impersonates the server which is uncorrupted and makes a client instance accepts a session with the server, but there is no partner server instance.

We also denote by $\Delta_i$ the distance between experiments $Exp_i$ and $Exp_{i+1}$.

**Experiment $Exp_0$.** This is the real attack experiment against the MFAKE protocol. According to the definitions, we have:

$$
\begin{align*}
Adv_{MFAKE}^P(A) &= 2 \cdot Pr[Sake_0] - 1 \\
Succ_{MFAKE}^{CAuth}(A) &= Pr[A_0^{CAuth}] \\
Succ_{MFAKE}^{SAuth}(A) &= Pr[A_0^{SAuth}]
\end{align*}
$$

(1)

**Experiment $Exp_1$.** We simulate the random oracles $H_i (i = 1, 2, 3)$ in this experiment as usual by maintaining hash lists $\wedge H_i (i = 1, 2, 3)$. We additionally simulate three private random oracles $H_i' (i = 1, 2, 3)$ which will be used in later experiments by maintaining hash lists $\wedge H_i' (i = 1, 2, 3)$. We also simulate all the instances, as the real players would do, for the Execute-queries, the Send-queries, the Reveal-queries, the Corruption-queries and the Test-queries. From this simulation, we see that this experiment is indistinguishable from the previous experiment. Thus we have:

$$\Delta_0 = 0$$

(2)

**Experiment $Exp_2$.** For an easier analysis, we cancel the sessions in which some unlikely collision appear in this experiment. We first cancel the sessions in which some collision on the session transcripts $((C, X), (S, Y, P_C, \sigma_S))$ occur, and we also cancel the sessions in which some collision appear on the output of the hash functions. Since transcripts involve at least one honest party and hash functions are modeled as random oracles, thus we have:

$$\Delta_1 \leq \frac{(q_{exe} + q_{send})^2}{2q} + \frac{3q^2_{h}}{2^{l-1}}$$

(3)

**Experiment $Exp_3$.** In this experiment, we begin to modify the simulation rules for passive sessions. It should be noted that if the adversary simply forwards the messages it receives from the oracle instances, such sessions are viewed as passive sessions although the adversary use Send queries in these sessions. For passive sessions in which one of the participants is fully corrupted by the adversary, we compute the authenticators $\alpha, \beta$ and the session key $sk$ using the corresponding private random oracles $H_i' (i = 1, 2, 3)$ simulated in $Exp_1$. More precisely, we compute $\alpha = H_1'(C, S, X^*, Y^*, P_C, R_C)$, $\beta = H_2'(C, S, X^*, Y^*, P_C, \sigma_S, R_C)$ and $sk = H_3'(C, S, X^*, Y^*, P_C, \sigma_S, \sigma_C, R_C)$ without using the Diffie-Hellman key $K_C/K_S$. This experiment actually deals with forward security.
The experiments $Exp_3$ and $Exp_2$ are indistinguishable unless $A$ queries the hash function $H_i(i = 1, 2, 3)$ using the messages $(C, S, X^*, Y^*, P_C, R_C, K_S)$, $(C, S, X^*, Y^*, P_C, \sigma_S, R_C, K_C)$ and $(C, S, X^*, Y^*, P_C, \sigma_S, \alpha_C, R_C, K_C)$, respectively. We denote such an event by $\text{PassiveAskH}$. To upper bound the probability of this event, we consider an auxiliary experiment $Exp_3$. The simulation of the participants changes in the experiment $Exp_3$, but the distributions remain the same. Let us be given an instance of the CDH problem $(U, V)$. We simulate the above-mentioned passive sessions using the self-reducibility of the CDH problem. To simulate these passive sessions, we first choose $a_0, a_1, b_0, b_1$ from $Z_q^*$. We compute $X^* = U^{a_0} g^{a_1} \cdot G^{pw}$ and $Y^* = V^{b_0} g^{b_1} \cdot G^{pw}$. All other simulation is the same as the one in $Exp_3$, which means we use private hash oracles to compute $\alpha, \beta$ and $sk$ in these sessions. If the event $\text{PassiveAskH}$ occurs, we can extract $CDH(U^{a_0} g^{a_1}, V^{b_0} g^{b_1}) = CDH(U^{a_0}, V^{b_0}) \cdot CDH(U^{a_0}, g^{b_1}) \cdot CDH(g^{a_1}, V^{b_0}) \cdot CDH(g^{a_1}, g^{b_1}) = CDH(U, V) g^{a_0 b_1} \cdot U^{a_0 b_1} \cdot V^{a_0 b_1} \cdot g^{a_1 b_1}$ from $\wedge H_i(i = 1, 2, 3)$. If an adversary can distinguish between the experiments $Exp_2$ and $Exp_3$, then we can solve the CDH problem with exactly the same advantage. Hence we have, thus we have:

$$\Delta_3 \leq q_h \cdot Adv^{\text{edh}}_{G} (O(t)).$$

**Experiment $Exp_4$.** In this experiment, we again modify the simulation rules for passive sessions. For passive sessions in which the password factor of the client is uncorrupted by the adversary, we simply choose $X^*$ and $Y^*$ randomly from $G$ without using the password. Moreover, we compute the authenticators $\alpha, \beta$ and the session key $sk$ of these sessions using the private random oracles $H_i(i = 1, 2, 3)$ as we did in the previous experiment. With a same analysis with the previous experiment, we have

$$\Delta_3 \leq q_h \cdot Adv^{\text{edh}}_{G} (O(t)).$$

**Experiment $Exp_5$.** In this experiment, we begin to modify the simulation rules for $Send$-queries. If the adversary asks $\text{Corruption}(C_i, pw_{C_i}, W_{C_i})$ query to a client $C_i$, we change the simulation rules of the server with respect to client $C_i$. The adversary knows the password and the biometric template of the client $C_i$, the only uncorrupted factor is the signature key $sk_{C_i}$. The simulation rules are modified as follows: when the server receives a $Send(S_i, (C_i, X^*))$ query, it sends back the message $(S, Y^*, P_{C_i}, \sigma_S)$ according to the description of the protocol. However, when the adversary sends back a $Send(S_i, (C_i, \sigma_{C_i}))$ to the server, the server simply rejects and terminate the session without verifying the validity of the signature $\sigma_{C_i}$. The experiments $Exp_2$ and $Exp_5$ are undistinguishable unless the adversary successfully forges a valid signature $\sigma_{C_i}$ without the knowledge of $sk_{C_i}$. Since we cancel the collisions of the transcript as well as the output of hash functions in $Exp_5$, the value $\beta$ will not appear in previous sessions. If the adversary forges a valid signature $\sigma_{C_i}$ on the value $\beta$, which means it breaks the existentially unforgeable security of the signature scheme. Thus, we have

$$\Delta_4 \leq Adv_{\text{sig}} (O(t)).$$
Experiment $Exp_6$. In this experiment, we again modify the simulation rules for $Send$-queries. If the adversary asks $Corruption(C_i, pw_{C_i}, sk_{C_i})$ query to a client $C_i$, we change the simulation rules of the server with respect to client $C_i$. More precisely, when the server receives a $Send(S, (C_i, X^*))$ query from the adversary, it randomly chooses $Y^*$ from $G_q$ without using the password. Moreover, the server computes $\alpha = H'_1(C, S, X^*, Y^*, P_{C_i})$ using the private hash function $H'_1$ without using the biometric key $R_{C_i}$ and the Diffie-Hellman value $K_S$. The server generates a signature $\sigma_S$ of the value $\alpha$ and sends the message $(S, Y^*, P_{C_i}, \sigma_S)$ to the adversary. When the adversary sends back a $Send(S, (C_i, \sigma_{C_i}))$ query to the server, the latter simply rejects and terminates the session without verifying the validity of the signature $\sigma_{C_i}$. The experiments $Exp_6$ and $Exp_7$ are undistinguishable unless the adversary queries the hash function $H_1$ on message $(C_i, S, X^*, Y^*, P_{C_i}, R_{C_i}, K_{C_i})$ or queries the hash function $H_2$ on message $(C_i, S, X^*, Y^*, P_{C_i}, \sigma_S, R_{C_i}, K_{C_i})$, where $K_{C_i} = CDH(X^*/u_{w_{C_i}}, g^{R_{C_i}}, Y^*/v_{w_{C_i}}, g^{d_{R_{C_i}}})$. If the adversary makes such queries to hash functions, we can recover the secret biometric key $R_{C_i}$ from the hash lists with the help of the adversary, which violates the security of the fuzzy extractor. Thus, we have

$$\Delta_5 \leq \frac{\varepsilon}{q_H}.$$  \hspace{1cm} (7)

Experiment $Exp_7$. In this experiment, we continue to modify the simulation rules for $Send$-queries. If the adversary asks $Corruption(C_i, w_{C_i}, sk_{C_i})$ query to a client $C_i$, we change the simulation rules of the server with respect to client $C_i$. More precisely, when the server receives a $Send(S, (C_i, X^*))$ query from the adversary, it randomly chooses $Y^*$ from $G_q$ without using the password. Moreover, the server computes $\alpha = H'_1(C, S, X^*, Y^*, P_{C_i})$ using the private hash function $H'_1$ without computing the Diffie-Hellman value $K_S$. The server generates a signature $\sigma_S$ of the value $\alpha$ and sends the message $(S, Y^*, P_{C_i}, \sigma_S)$ to the server. When the adversary sends back a $Send(S, (C_i, \sigma_{C_i}))$ query to the server, the server simply rejects and terminates the session without verifying the validity of the signature $\sigma_{C_i}$. The experiments $Exp_6$ and $Exp_7$ are undistinguishable unless the adversary queries the hash function $H_1$ on message $(C_i, S, X^*, Y^*, P_{C_i}, R_{C_i}, K_{C_i})$ or queries the hash function $H_2$ on message $(C_i, S, X^*, Y^*, P_{C_i}, \sigma_S, R_{C_i}, K_{C_i})$, in which $K_{C_i} = CDH(X^*/u_{w_{C_i}}, g^{R_{C_i}}, Y^*/v_{w_{C_i}}, g^{d_{R_{C_i}}})$. We denote this bad event by $ActiveAskH$. Thus, we have

$$\Delta_6 \leq Pr[ActiveAskH_7].$$  \hspace{1cm} (8)

Experiment $Exp_8$. In this experiment, we again modify the simulation rules for $Send$-queries. If the adversary fully corrupts a client $C_i$, it is obvious that the adversary can impersonate $C_i$. However, it is desirable that the adversary cannot impersonate the server with $C_i$’s secret information to fool $C_i$. We change the simulation rules of the client $C_i$. More precisely, when $C_i$ receives a $Send(C_i, Start)$ query, he responds according to the description of the protocol and sends $(C_i, X^*)$ to the adversary. Moreover, when $C_i$ receives
a $\text{Send}(C_i, (S,Y^*, PC_i, \sigma_S))$ query, it simply rejects and terminates the session without verifying the validity of $\sigma_S$. It should be noted that the server is uncorrupted. As a result, $sk_S$ is unknown to the adversary. The experiments $\text{Exp}_7$ and $\text{Exp}_8$ are undistinguishable unless the adversary forges a valid signature $\sigma_S$. With a similar analysis with $\text{Exp}_5$, we have

$$\Delta_7 \leq \text{Adv}_{\text{sig}}(O(t)). \tag{9}$$

**Experiment $\text{Exp}_9$.** In this experiment, we modify the simulation rules for $\text{Send}$-queries for the last time. If the adversary corrupts the server, it is obvious that the adversary can impersonate the server and knows the passwords and biometric keys of all clients. However, it is desirable that the adversary cannot impersonate an uncorrupted client $C_i$ to $S$. We change the simulation rules of the server. More precisely, when $S$ receives a $\text{Send}(S, (C_i, X^*))$ query, it responds according to the description of the protocol and sends $(S, Y^*, PC_i, \sigma_S)$ to the adversary. Moreover, when $S$ receives a $\text{Send}(S, (C_i, \sigma_{C_i}))$ query, it simply rejects and terminates the session without verifying the validity of $\sigma_{C_i}$. It should be noted that $sk_{C_i}$ is unknown to the adversary. The experiments $\text{Exp}_8$ and $\text{Exp}_9$ are undistinguishable unless the adversary forges a valid signature $\sigma_{C_i}$. With a similar analysis with $\text{Exp}_5$, we have

$$\Delta_8 \leq \text{Adv}_{\text{sig}}(O(t)). \tag{10}$$

In this final experiment, only the passive sessions accept and generate session keys. These session keys are all randomly chosen. All the active sessions (except for the session in which the adversary simply relays the messages) are terminated without accepting. Consequently, the adversary has no advantage in distinguishing the session key and break the mutual authentication. Thus, we have

$$Pr[S_{ke}^{SK}] = 1/2$$

$$Pr[A_{\text{CAuth}}^{\text{Auth}}] = 0$$

$$Pr[A_{\text{SAuth}}^{\text{Auth}}] = 0 \tag{11}$$

Now, we evaluate the probability of the the bad event $\text{ActiveAskH}_9$. Note that if the password is uncorrupted, it is never used in the simulation. The event $\text{ActiveAskHC}_9$ corresponds to an attack in which the adversary tries to impersonate the client $C_i$ to the server $S$ without the password. Since the authenticator $\beta$ sent by the adversary has been computed with at most one $pw_{C_i}$ value. Thus, we have

$$Pr[\text{ActiveAskH}_9] \leq \frac{q_{\text{send}}}{|D|}. \tag{12}$$

Combining all the above equations together, we get the announced result.