Practical Threshold Multi-Factor Authentication

Li, Wenting; Cheng, Haibo; Wang, Ping; Liang, Kaitai

DOI
10.1109/TIFS.2021.3081263

Publication date
2021

Document Version
Accepted author manuscript

Published in
IEEE Transactions on Information Forensics and Security

Practical Threshold Multi-Factor Authentication

Wenting Li, Haibo Cheng, Ping Wang*, *Senior Member, IEEE* and Kaitai Liang, *Member, IEEE*

Abstract—Multi-factor authentication (MFA) has been widely used to safeguard high-value assets. Unlike single-factor authentication (e.g., password-only login), t-factor authentication (tFA) requires a user *always* to carry and present t specified factors so as to strengthen the security of login. Nevertheless, this may restrict user experience in limiting the flexibility of factor usage, e.g., the user may prefer to choose any factors *at hand* for login authentication. To bring back usability and flexibility without loss of security, we introduce a new notion of authentication, called (t, n) *threshold MFA*, that allows a user to actively choose t factors out of n based on preference. We further define the "most-rigorous" multi-factor security model for the new notion, allowing attackers to control public channels, launch active/passive attacks, and compromise/corrupt any subset of parties as well as factors. We state that the model can capture the most practical security needs in the literature. We design a threshold MFA key exchange (T-MFAKE) protocol built on the top of a threshold oblivious pseudorandom function and an authenticated key exchange protocol. Our protocol achieves the "highest-attainable" security against all attacking attempts in the context of parties/factors being compromised/corrupted. As for efficiency, our design only requires $4+t$ exponentiations, 2 multi-exponentiations and **2** communication rounds. Compared with existing tFA schemes, even the degenerated (t, t) version of our protocol achieves the strongest security (stronger than most schemes) and higher efficiency on computational and communication. We instantiate our design on real-world platform to highlight its practicability and efficiency.

Index Terms—Threshold, Multi-Factor Authentication, Key Exchange, Password.

1 Introduction

Multi-factor authentication (MFA) has been deployed in real-world applications to safeguard high-value assets, e.g., online banking. A user is required to make use of t factors, at the same time, to execute secure authentication. This t-factor authentication (tFA) naturally brings more challenges, than single-factor authentication, to attackers since there are t factors that need to be compromised for an impersonation. For instance, EMV [1], [2], a widely adopted payment method, allows one to present his smart card and the password/PIN to pay a bill on a POS terminal. Attackers cannot steal the money by only being given the smart card or the password.

Motivation. The current tFA schemes require users to present t *fixed* factors for authentication. This "static"-factor authentication mode could not provide flexibility for authentication because users may not be able to always present the t factors anywhere and anytime, for example, one of the factors may be left at home and even be lost. Furthermore, this mode also limits the preference of factor usage - *which factors should users choose to take*. Users, in practice, prefer to leverage those factors which are at hand. To this end, the usage of factors should be more dynamic. Besides, the number t becomes a bottleneck between the security and usability in tFAs. It is clear that a *higher* security level requires a *larger* t (e.g., $t = 5$) but this brings *lower* usability for users.

- * Corresponding author
- W. Li is with School of Software & Microelectronics, Peking University, China. E-mail: wentingli@pku.edu.cn.
- H. Cheng is with School of Electronics Engineering and Computer Science, Peking University, China. E-mail: hbcheng@pku.edu.cn.
- P. Wang is with the National Engineering Research Center for Software Engineering, the School of Software and Microelectronics, Peking University, and with Key Laboratory of High Confidence Software Technologies (PKU), Ministry of Education, China. E-mail: pwang@pku.edu.cn.
- K. Liang is with Department of Intelligent Systems, Delft University of Technology, The Netherlands. E-mail: kaitai.liang@tudelft.nl.

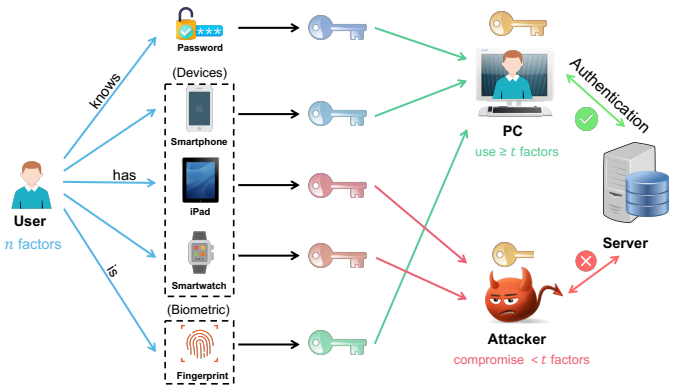


Fig. 1: Threshold multi-factor authentication

We propose the notion of (t, n) *threshold MFA (T-MFA)* to address the issues. As shown in Fig. 1, T-MFA allows a user to register n factors and adaptively choose arbitrary t out of n for authentication, in which t and n can be flexibly set by the user based on its preference (note that (t, n) T-MFA is naturally degenerated to tFA, when $n = t$). Compared with tFA, (t, n) T-MFA significantly improves usability with the same security level (attackers need to compromise t factors to impersonate users). More importantly, T-MFA allows a user to autonomously increase t to achieve higher security according to the number of always-at-hand factors. Since the usage of multiple personal electronic devices at home and workplace has become common [3], the user may hold several devices, e.g., a smartphone, a smartwatch, and a tablet, as multiple factors along with its password (or other types of factor) and later, it just uses a subgroup of factors ($\geq t$) in verification. T-MFA also allows the user to leverage fixed-location (non-portable) devices as factors, e.g., a smart speaker at home, or an intranet server in the workplace. The user may now set a larger t without compromising usability, e.g., $t = 4$: using the password, smartphone and smartwatch as portable factors, and either smart speaker or

server as the unportable one. Besides, if a device is lost, the authentication can still be successfully executed with another one. Naturally, the user can also use more devices, e.g., an old smartphone as a factor, for backup.

Design Challenges. The concept of T-MFA seems natural, but, to our best knowledge, none of the existing studies has paid attention to it. We state that this is because of the significant challenges in designing a T-MFA scheme and further analyzing its security. Note that we here mainly focus on remote authentication rather than local authentication, since the design for the former is more challenging and usually can apply to the latter.

One may think that a trivial way to implement T-MFA can be captured as follows. A user may send any t (may be in challenge-response mode) out of n registered factors to a server, so that the server can verify them one by one. This design is quite similar to the multiple login authentication. For instance, the user can log in to WeChat via either option (1): phone number + password, or option (2): phone number + SMS. But this requires the server to store factors in a file for (possessing knowledge of) verification, *leading to potential vulnerabilities against server compromise*. Via compromising the server, the attacker can access the file storing those factors, which yields the possibility of factor recovery. A classic example could be that the attacker attempts to recover the password (which is one of the factors) via offline password guessing. Once the factors are retrieved, the attacker can impersonate the valid user. It is also probable that all the accounts that used the same revealed factors will be controlled by the attacker. Beyond impersonation, the server compromise may bring even worse consequences - harming user long-term privacy - if the storage file includes personal biometrics (e.g., gait [4], electroencephalogram [5]).

Another potential approach is to leverage threshold cryptography. For multiple devices, a trivial T-MFA design can request each device to share the user's secret for authentication via a threshold secret share scheme. This idea is used in some server-side authentication scenarios, e.g., [6], [7], [8], requesting multiple servers to store cryptographic keys. However, this mechanism still cannot be perfectly compatible with some types of factors, e.g., passwords and biometrics, which are not natural cryptographic keys, and require non-server-side storage.

One may also try to extend the current MFA schemes to yield a T-MFA. However, even for MFA, it is difficult to capture the "highest attainable security" with a precise and well-defined security model and further satisfy the security by a well-designed protocol. It is not trivial to handle all different parties and their factors along with all the combinations over compromise. A single MFA scheme will perform different security levels according to the various compromise cases. *This brings a great challenge for researchers to precisely capture the security of MFA, which also applies to T-MFA.* Due to the lack of a precise security model, most MFA schemes fail to achieve "sound and practical security". The industrial MFA schemes (e.g., [9], [10]) cannot resist server compromise, and most of the academic ones (e.g., [11], [12]) fail to provide forward security, key-compromise impersonation (KCI) security, or suffer from other security issues. To the best of our knowledge, only Jarecki et al. [13] 2FA protocol achieves the highest attainable security. But their protocol requires expensive cost in communication (10 rounds) so it is not practical for real-world applications. Note the detailed analysis for the existing MFA schemes will be given in Section 2. Therefore, *no current MFA design with both the highest attainable security and efficiency can be used to construct T-MFA*. And again, the existing MFA schemes suffer from the "pre-set" limit: the number and the type of factors are fixed in the very beginning, e.g., the password-and-device based 2FAs [9], [10], [13], [14], and biometrics-based 3FAs [11], [12]. *They cannot be trivially extended to offer the dynamic and flexible usage of factors to T-MFA.*

Our contribution. We here briefly describe the contribution.

Real-world Security. We introduce a game-based model to precisely capture the security for (t, n) T-MFA key exchange (T-MFAKE). Our model allows attackers to control communication on public channels and to compromise any parties and factors, and considers the use of different types of factors, including passwords, devices, and biometrics. For each combination of party or factor compromise, we give special security bound (i.e., advantage) to capture the highest attainable security by only allowing inevitable attacks. Specifically, our novel security notion considers the inevitable attacks against password via: 1) if $t-1$ factors (excluding password) are compromised, online password guessing attack is inevitable and allowed; 2) if $t-1$ factors (excluding password) are compromised and server is compromised or corrupted, offline password guessing attack is also inevitable and allowed; 3) if t factors are compromised, user impersonation is trivially inevitable; 4) if server is compromised or corrupted, impersonating server is trivially inevitable; 5) otherwise, no other inevitable attacks are allowed. Note we consider the similar inevitable guessing attacks for a biometric factor. If a protocol is secure in our model, it can achieve practical real-world security (i.e. resisting all the attacks except the "inevitable ones").

Practical Design. We propose a fast and secure (t, n) T-MFAKE protocol. We allow a user to leverage a password, multiple devices, and biometrics (optional) as authentication factors. Because of the advantages of passwords on usability and deployability [15], we require the user to always type his password for authentication (cannot use another factor instead it) in order to provide the last defense for the worst case where the other factors are all compromised. Our construction is built on the top of a threshold oblivious pseudorandom function (TOPRF) and an authenticated key exchange (AKE) protocol. The core idea is to enhance the password to a cryptographic key with any other $t-1$ factors (via TOPRF) and further use the key for authentication (via AKE). Each factor except the password corresponds to a secret key in TOPRF: for a device factor, the key is stored in the device locally; for a biometric factor, the key is converted by fuzzy extractor [16], which prevents the biometrics from leaking even if the storage file is compromised. By TOPRF, our protocol provides *factor invisibility*, i.e., the server cannot see which factors are registered and used for authentication. This means that even if the server is compromised, the factors can stay safe. Besides, using the refreshment mechanism of threshold schemes, our protocol can remotely revoke the lost (or misfunctioned) devices. As for efficiency, our protocol is extremely light-weight w.r.t. computation and communication cost, *requiring* $4+t$ *exponentiations,* 2 *multi-*

exponentiations and only 2 *rounds communication* (by parallel running TOPRF and AKE).

Security Analysis. We further analyze the security of our T-MFAKE protocol. Our protocol holds against all practical and real-world attacks (except the inevitable ones), and thus *achieves the highest attainable security w.r.t. arbitrary combination of party or factor compromise.* The degenerated $(2, 2)$ version of our protocol holds the same security as the state-of-the-art 2FAKE, OpTFA [13], while (t, t) version achieves higher security as compared to other existing tFA schemes.

Efficient Implementation. We implement the $(2, 2)$ version of our protocol on a real-world system with a smartphone, a PC, and a remote server, and further evaluate its runtime performance. The experimental results show that the new version is *138.25% and 148.49% faster than* OpTFA on communication and computation, respectively (note that OpTFA costs 12 exponentiations, 2 multi-exponentiations, and **10**-round communication).

To summarize, our main contributions are as follows:

1) The notion of (t, n) threshold multi-factor authentication (T-MFA), for the first time in the literature, is proposed to allow users to freely and actively choose t factors out of n for authentication.
2) A security model is defined to capture the real-world highest attainable security for T-MFA key exchange (T-MFAKE) protocol.
3) An efficient T-MFAKE protocol is proposed, only requiring $4 + t$ exponentiations, 2 multi-exponentiations and only 2-round communication.
4) A formal security analysis for our protocol is given, presenting that the protocol achieves the highest attainable security.
5) An efficient implementation and the performance evaluation are demonstrated to highlight the efficiency of our design in the real-world platform.

2 Related Works

We here briefly review some typical MFA protocols.

2.1 MFA Schemes in Industry

In industry, many 2FA schemes are widely implemented in the web-based authentication, including Google Authenticator [9], FIDO U2F [17], and Duo 2FA [10]. All these schemes authenticate the factors separately. For example, in password-and-device authentication mode, the user is requested to send a password to the server via a server-authenticated secure channel and meanwhile, the server also needs a PIN generated from the smartphone. But the verification of the password and PIN is separate, meaning that these two factors are not "tightly" bound together as one verification. This may bring several limits:

1) The server can access the password in plaintext, which may be leaked accidentally or on purpose, for example, Github recorded plaintext passwords in secure internal logs because of some bug [18]. Similar issues have also happened to Google, Twitter and Facebook [19].
2) The server stores the hash values of passwords (or other verification values), increasing the leak risk. In practice, billions of passwords have been leaked from hundreds of websites [20]. With hash values, an attacker can efficiently and offline recover the plaintext by exploiting password guessing algorithms (e.g., [21], [22]). Since users usually reuse passwords on different accounts [21], the attacker probably compromises multiple accounts with the same password.
3) The authentication requires a Public Key Infrastructure (PKI) to establish the server-authenticated secure channel. This may not scale well and will leak password or other factors if PKI is infiltrated.

2.2 Academic Studies on MFA

Many studies, e.g., [11], [12], [13], [14], focus on the security analysis along with new designs for MFA. We first review the methods of security analysis they use and then make discussions over their constructions.

Security analysis methods. Some of the analyses are based on heuristic attacks, BAN logic or automatic tools [23], [24], [25], [26], which cannot provide a sound and solid analysis. Others (e.g., [11], [12], [27]) are in game-based security model (usually BPR or ROR model [28], [29]) to provide a provable security analysis. However, most of the game-based analyses (except for [13], [14]) fail to model the highest attainable security of MFA. This is so because they cannot precisely define the security requirements for various combinations of party or factor compromise. Taking an ideally secure 2FA (password-and-device mode) as an example. 1) if the attacker does not compromise the device (case I), she cannot impersonate the user except a *negligible* probability; 2) if the attacker compromises the device (case II), she inevitably can carry out online password guessing attacks to impersonate the user with a *non-negligible* probability. The security requirements for the above cases should be specified respectively. But the current security models just define an upper-bound attacking advantage to define the security requirements w.r.t. all the compromise cases. This bound only captures the security in the worst case (referring to the case where all factors except for password are compromised), and thus cannot precisely model the security for other cases (e.g., the device is not compromised). Take a further example, if a 2FA protocol suffers from online password guessing attacks in case I, it satisfies the security requirement of the current models but is clearly insecure in practice. Thus we state *the limitation of the security models incurs that a concrete protocol - proved secure by current research works - cannot satisfy the highest security requirements in all practical cases.* In other words, *what has been proved secure is not always secure in practice.*

To tackle the issue, Shirvanian et al. [14] first defined different requirements for 2FA w.r.t. a few cases of compromise combination, and provided specified security bound for each of the cases. But their model does not consider the case of server corruption. Later, Jarecki et al. [13] proposed an improved model by taking more other cases into account, capturing the highest attainable security for 2FA. Our security notion is inspired by theirs but also provides the highest attainable security for MFA and even T-MFA.

More academic MFA schemes. Due to the limitation of the security models, current MFA schemes may suffer from unexpected real-world attacks which are not considered

TABLE 1: Summary and Comparison of existing multi-factor authentication schemes

Schemes	Techniques	Advantages	Limitations
Industrial schemes [9], [10], [17]	TLS + Separate factor authentication	Deployability.	Suffer from PKI failures and server compromise.
Shirvanian et al. [14]	Salt reconstruction	Resist server compromise.	Suffer from server corruption.
Jarecki et al. [13]	KE + SAS-MA + PTR + aPAKE	Highest attainable security.	Hight computational and communication costs.
Zhang et al. [11]	PKE + MAC + Fuzzy extractor	Efficiency.	Suffer from KCI attacks.
Wazid et al.'s scheme [12]	Hash + XOR + Encryption	Efficiency.	Suffer from KCI, ESL, offline password guessing attacks.
Ours	TOPRF + SaPAKE (+ fuzzy extractor)	Security (highest attainable), efficiency, and usability.	No above weaknesses.

and covered in the models, meaning they are insecure in practical use. Here, we briefly present some examples.

3FA schemes. Zhang et al. [11] proposed a 3FA scheme combining a password, a device, and biometrics. Although proved secure in the BRP model (with one security bound), the scheme still suffers from key-compromise impersonation (KCI) attacks. Expressly, if all factors (α, β, γ) of a user are compromised, attackers can impersonate the server to interact with the user (by leveraging $Z = H^{(\alpha+\beta+\gamma)}$). Similarly, we find Wazid et al.'s scheme [12] which is proved in the ROR model but still suffers from KCI attacks and ephemeral secret leakage (ESL) attacks. Besides, this scheme cannot hold against offline password guessing, if device and biometrics are compromised.

2FA schemes. Shirvanian et al. [14] proposed a 2FA scheme twisting password and device, based on industrial 2FA design. The scheme is strengthened to resist server compromise (i.e., the leakage of server's data). However, it heavily relies on PKI and cannot prevent server from accessing plaintext passwords, which leads to the aforementioned vulnerabilities identified from the industrial schemes. Jarecki et al. [13] later proposed an improved 2FA, *OpTFA*, to address all the aforementioned vulnerabilities. However, OpTFA is too complex and inefficient. It requires 12 exponentiations, 2 multi-exponentiations on computation cost and **10** rounds on communication. Besides, it is designed for 2FA context only and thus is difficult to be extended to an MFA variant.

3 Preliminaries

In this section, we review the building blocks we are going to use in our design, including a threshold oblivious pseudorandom function (TOPRF) and an authenticated key exchange (AKE) protocol, a password-based authenticated key exchange (PAKE) protocol and the fuzzy extractor.

3.1 Threshold Oblivious Pseudorandom Function

Threshold oblivious pseudorandom function (TOPRF) is introduced by Jarecki et al. [8]. It is a threshold variant of the client-server oblivious pseudorandom function (OPRF). The client-server OPRF consists of a PRF F and a client-server protocol. In the protocol, the server holds a secret key s, and the client inputs x to get $F_s(x)$ without knowing s. Meanwhile, the server knows nothing about the input x. This is an appropriate method to enhance the low-entropy password pw to a cryptographic key $F_s(pw)$. TOPRF leverages multiple servers to collectively control the key to resist server compromise. For a (t, n) TOPRF, the client needs to run the TOPRF protocol with t servers. If no more than t servers are compromised or corrupted, the attacker cannot offline calculate F and has to interact with some servers. The formal security of TOPRF is defined in Universally Composable (UC) Framework [30]. Specifically, the executions of a real TOPRF protocol Π and the ideal TOPRF function $\mathcal{F}_{\text{TOPRF}}$ are indistinguishable.

We will use a TOPRF scheme, *2HashTDH* [8], in our protocol. 2HashTDH is a threshold variant of *2HashDH* [31], an OPRF scheme. We detail 2HashTDH in Fig. 2. In 2HashTDH, the secret key s is shared by Shamir's secret sharing scheme. During the execution of 2HashTDH, s is used to calculate the PRF but not reconstructed on any party, which can avoid the leakage of s (unless t servers are compromised). In addition, the randomness of r guarantees the servers (and attackers) cannot get any information about x. Formally, 2HashTDH is secure under the One-More Diffie-Hellman assumption and in the random-oracle model.

Shamir's secret sharing used in TOPRF is a typical threshold method. In (t, n) Shamir's secret sharing, a (randomly-generated) secret key s is divided into n shares $\{s\}_{i=1}^{n}$. Arbitrary t shares $\{s_{D_i}\}_{i\in I}$ (where I is the index set of the t shares) can reconstruct s. This construction is leveraged by a polynomial of degree t,

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_t x^t,$$

where $a_0 = s$, $a_1, a_2, \ldots, a_t$ are randomly generated. The i-th share of s is $f(i)$. With t shares $\{s_i\}_{i\in I}$, s is reconstructed as

$$s = \sum_{i\in I} s_i \lambda_i,$$

where λ_i is the Lagrange interpolation coefficient for i in I,

$$\lambda_i = \prod_{j\in I, j\neq i} \frac{-j}{i-j}.$$

Note that we made a slight modification on the original 2HashTDH [8]. In the original version, S_i calculates λ_i; in our version (Fig. 2), C does the calculation. In our way, C does not need to send I to S_i, which saves communication costs (note the length of I is proportional to t).

3.2 Authenticated Key Exchange

Authenticated key exchange (AKE) has been studied for decades and there are many protocols have been proposed, e.g., [32], [33]. In contrast to 2FAKE and MFAKE, AKE requires each party to hold a long-lived cryptographic key. Two parties can establish a temporary session key with the long-lived keys and encrypting further communications

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3081263, IEEE Transactions on Information Forensics and Security

Parameter

- G is a cyclic group with a generator g and prime order m.
- H, H' are two hash functions with ranges $\{0,1\}^l$ and G, respectively. The PRF $F_s(x)$ is defined as $H(x, H'(x)^s)$.

Initialization

- n servers initialize a (t, n) Shamir's secret sharing to generate each secret share $\{s_i\}_{i=1}^n$ of s.

Calculation

- With an input x, U picks $r \leftarrow\$ \mathbb{Z}_m$, calculates $\alpha \leftarrow H'(x)^r$, and sends α to t servers $\{S_i\}_{i\in I}$, where I is the index set.
- With a message α, S_i calculates $\beta_i \leftarrow \alpha^{s_i}$, and sends β_i to U.
- With messages $\{\beta_i\}_{i\in I}$, U calculates $H(x, \prod_{i\in I} \beta_i^{\lambda_i/r})$ and outputs it. Here, λ_i is the Lagrange interpolation coefficient for i in I.

Fig. 2: 2HashTDH

using the session key without the long-lived keys. The security definition [32] for AKE protocols is similar to that for MFAKE (see Section 4), which is also defined by distinguishing the real session key and a random one. We use $\mathsf{Adv}^{\mathrm{ake}}_{\Pi}(\mathcal{A})$ to denote the advantage of the attacker $\mathcal{A}$ against an AKE protocol Π. We do not detail the security game for AKE protocols, and readers can refer to [32], [33].

To construct our protocol, we require the AKE protocol to be secure as defined in [32] and additionally resist KCI attacks. This means if the long-lived key of a party P is compromised, the attacker cannot impersonate another party Q to interact with P by P's key. To capture the security, the security game defined in [32] should be modified by including more fresh instances whose internal states and its partner's states are not compromised.

HMQV [33] is a well-known protocol with KCI resistance. We will use it in our protocol, due to its security and efficiency. Fig. 3 details HMQV (with implicit authentication). It only needs one exponentiation and one multi-exponentiation for each party and two communication rounds. Its message flows are similar to those of Diffie-Hellman key exchange, but it additionally uses public key cryptography (k_A, K_A for participant A) for authentication. Although HMQV does not provide explicit authentication, it is trivial to achieve explicit authentication by adding one round as in [28]. Krawczyk proves the AKE security of HMQV as well as KCI security/resistance. Formally, in the aforementioned (modified) game with KCI security, the advantage $\mathsf{Adv}^{\mathrm{ake}}_{\mathrm{HMQV}}$ is negligible under Computational Diffie-Hellman assumption in the random-oracle model.

3.3 Password-based Authenticated Key Exchange

Password-based Authenticated Key Exchange (PAKE) also has been studied for decades. Some important studies are [34], [35], [36]. In symmetric PAKE, two parties share a long-lived password with low entropy for authentication and key exchange. In asymmetric PAKE, one party (the user) holds

Parameter

- G is a cyclic group with a generator g and prime order m.
- H, H' are two hash functions with ranges $\mathbb{Z}_m$ and $\{0,1\}^l$. l is the length of the session key. Note that these parameters may not be the same as those in Fig. 2.

Initialization

- Each participant A generates the private and public keys k_A, K_A by $k_A \leftarrow\$ \mathbb{Z}_m, K_A \leftarrow g^{k_A}$. We assume that each participant securely gets the others' public key (in our MFAKE, the public key is sent on a secure channel in initialization phase).

Authentication

- A picks $x \leftarrow\$ \mathbb{Z}_m$, calculates $X \leftarrow g^x$, sends (A, X) to B.
- With a message (A, X), B picks $y \leftarrow\$ \mathbb{Z}_m$, calculates $Y \leftarrow g^y$ and sends Y to A and outputs it. Then B calculates $SK \leftarrow H((XK_A^{H'(X,K_B)})^{y+H'(Y,K_A)k_B})$.
- With message Y, A calculates $SK \leftarrow H((YK_B^{H'(Y,K_A)})^{x+H'(X,K_B)k_A})$ and outputs it.

Fig. 3: HMQV with implicit authentication. Adding one round as in [28] can achieve explicit authentication (i.e., adding the message $H''(SK, 1)$ and $H''(SK, 2)$ to the last two rounds respectively, where H'' is a hash function with the range $\{0,1\}^l$).

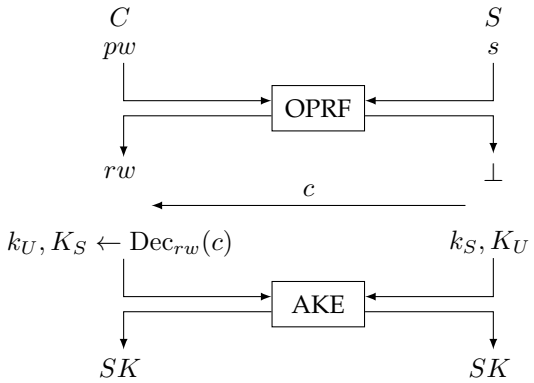


Fig. 4: Schematic diagram of OPAQUE

a password and the other party (the server) holds a non-password-equivalent verifier of the password.

Informally, a 2FAKE protocol can be seen as a PAKE protocol if the device is corrupted. So the former may leverage the latter as a component, e.g., OpTFA [13], to protect the password for the case of device corruption. However, this usually leads to complexity and inefficiency. Instead, we are inspired by the design of a PAKE protocol, OPAQUE [35], and further use it for our T-MFAKE.

As shown in Fig. 4, OPAQUE leverages an OPRF, an AKE and an encryption scheme. Its core roadmap is as follows:

1) The client enhances the password pw to a cryptographic key rw (also called random password) by running OPRF with the server.
2) The secret keys (k_U, K_U) of the user in AKE is en-

Parameter

- All parameters in 2HashDH (i.e., $(1,1)$ 2HashTDH) and HMQV. To distinguish these parameters, we add the subscript 1 and 2 to parameters in 2HashTDH and HMQV, respectively, if necessary.
- An encryption scheme $(\mathrm{Enc}, \mathrm{Dec})$.

Initialization

- C generates the secret key s of S in the 2HashDH and sends s to D; calculates $rw \leftarrow H_1(pw, H_1'(pw)^s)$ (note running 2HashDH with S and input pw will get rw). s can be generated by S. If so, rw is got by C running 2HashDH.
- C generates the private and public keys (k_U, K_U) by $k_U \leftarrow\$ \mathbb{Z}_{m_2}$ and $K_U \leftarrow g_2^{k_U}$ for HMQV; gets S's public key K_S and encrypts (k_U, K_S) to c with the key rw ($c \leftarrow \mathrm{Enc}_{rw}(k_U, K_s)$); sends K_U to S and sends c to D.
- S generates the private and public keys (k_S, K_S) by $k_S \leftarrow\$ \mathbb{Z}_{m_2}$ and $K_S \leftarrow g_2^{k_S}$ for HMQV; sends K_S to C; stores K_U and the share s_S for 2HashDH.

Authentication

- C picks $r \leftarrow\$ \mathbb{Z}_{m_1}$ and calculates $\alpha \leftarrow H_1'(pw)^r$; picks $x \leftarrow\$ \mathbb{Z}_{m_2}$ and calculates $X \leftarrow g_2^x$; sends (U, X, α) to S.
- Getting (U, X, α) from C, S picks $y \leftarrow\$ \mathbb{Z}_{m_2}$ and calculates $Y \leftarrow g_2^y, \beta \leftarrow \alpha^{s_S}$; sends (Y, β) to C; calculates $SK \leftarrow H_2((XK_U^{H_2'(X,K_S)})^{y+H_2'(Y,K_U)k_S})$ and outputs it.
- Getting (Y, β) from S, C calculates $rw \leftarrow H_1(pw, \beta^{1/r})$, $k_U, K_S \leftarrow \mathrm{Dec}_{rw}(c)$; calculates $SK \leftarrow H_2((YK_S^{H_2'(Y,K_U)})^{x+H_2'(X,K_S)k_U})$ and outputs it.

Fig. 5: OPAQUE with implicit authentication.

crypted by rw and the ciphertext is stored on the server.

3) During the authentication phase, the client reconstructs rw by running OPRF with the server, gets the keys in AKE by decrying c, and then runs AKE to establish the session key. Note that c is sent to the client by the server along with the messages of OPRF and AKE.

The details of OPAQUE are given in Fig. 5.

Jarecki et al. [35] prove the security of OPAQUE in the Universally Composable (UC) Framework. This means that the execution of OPAQUE is indistinguishable from that of the ideal functionality $\mathcal{F}_{\text{SaPAKE}}$, i.e., the distinguishing advantage $\mathsf{Adv}^{\text{dis}}_{\text{HMQV},\mathcal{F}_{\text{SaPAKE}}}$ is negligible. $\mathcal{F}_{\text{SaPAKE}}$ only allows two inevitable attacks: 1) the online password guessing attack; 2) the offline password guessing attack in the case of server compromise. Therefore in the game G^{PAKE} for PAKE, we have:

1) If S is corrupted,
$$\mathsf{Adv}^{\text{pake}}_{\text{OPAQUE}} \leq \frac{1}{n}q + \mathsf{Adv}^{\text{dis}}_{\text{OPAQUE},\mathcal{F}_{\text{SaPAKE}}},$$
where q is the number of online password guessing attacks.
2) Otherwise,
$$\mathsf{Adv}^{\text{pake}}_{\text{OPAQUE}} \leq \frac{1}{n}q' + \mathsf{Adv}^{\text{dis}}_{\text{OPAQUE},\mathcal{F}_{\text{SaPAKE}}},$$
where q' is the number of offline password guessing attacks.

The security of OPAQUE requires the security of OPRF and AKE (with KCI resistance) as well as a *random-key robust and equivocable authenticated* encryption scheme. We briefly explain the properties required for the encryption:

1) *Authentication*. Authenticated encryption provides security against chosen ciphertext attacks as well as message integrity and confidentiality. It can be constructed on the top of an encryption scheme and a message authentication code (MAC).
2) *Random-key robustness*. This property means that it is difficult to construct a ciphertext c for two randomly-generated keys k_1, k_2 such that decrypting c using both k_1, k_2 will not fail. Formally, for an arbitrary PPT attacker $\mathcal{A}$,
$$\Pr_{k_1,k_2 \leftarrow\$ \{0,1\}^l}[c \leftarrow \mathcal{A}(k_1,k_2) \text{ s.t. } \mathrm{Dec}_{k_i}(c) \neq \perp, i = 1,2]$$
is negligible.
3) *Equivocability*. This property means that the encryption of a message can be simulated by 1) first creating the ciphertext without knowing the plaintext and 2) then creating the key for the given plaintext. Formally, an arbitrary attacker $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that the following two games are indistinguishable:
   a) The real game: $\mathcal{A}$ gives a message m, generates (k, m) by $k \leftarrow\$ \{0,1\}^l$ and $c \leftarrow \mathrm{Enc}_k(m)$.
   b) The simulated game: $\mathcal{A}$ gives a message m, generates (k, m) by $c \leftarrow \mathcal{S}(|m|)$ and $k \leftarrow \mathcal{S}(m)$.

3.4 Fuzzy Extractor

To generate cryptographic key from biometric characteristic with information protection, several methods have been proposed, e.g., fuzzy vault [37], fuzzy extractor [16]. We choose the latter for our scheme since it provides stronger security. Fuzzy extractor generates a uniformly random key R and transforms the key to a helper string P based on a biometric input w as
$$(R, P) \leftarrow \mathrm{Gen}(w)$$
at initialization. A biometric input w' which is close to w can extract the key R from P. Formally, if $\mathrm{dis}(w', w) \leq t$ (t is a parameter), then
$$\mathrm{Rep}(w', P) = R.$$
Fuzzy extractor guarantees that the helper string P does not leak information of the random key R as well as the biometric input w, maintaining data security and user privacy. In our scenario, the initialization and extraction can be done on the smartphone equipped with the biometric recognition sensor and storing the helper string P. In this paper, we do not review specific constructions of the fuzzy extractor, and the reader can refer to [16] for more details.

4 Security Model for Threshold MFAKE

Inspired by the CK-adversary model for AKE [38], [39], [40] and Jarecki et al.'s security model [13] for 2FA with a password and an auxiliary device, we propose a model for threshold MFAKE (T-MFAKE) supporting various types and various numbers of factors.

Protocol participants. There are several participants in T-MFAKE: a user U, a client C, one or multiple device(s) $\{D_i\}_{i\in I}$ (where I is the index set), and a server S. The user U leverages a password pw, and the device(s) $\{D_i\}_{i\in I}$ (maybe with biometrics) as factors for authentication.

Protocol execution and communication model. In the registration phase of a (t, n) T-MFAKE protocol Π, the user U registers the combination of the n authentication factors on the server S. This registration is assumed to be securely done. Take opening a bank account as an example, the registration can be done physically in the bank, which is regarded as secure execution. Note the user U does not register the client C, which means U does not leverage a fixed client and further may use different clients (e.g., public computers in libraries) in the authentication phase. This brings advantages to usability.

In the authentication phase of Π, the user U leverages t factors (usually containing the password and device(s) $\{D_i\}_{i\in I'}$, where $|I'| = t-1$) to run Π with the server S. The communication between C and S is via a public channel. After successful authentication, both the server S and the client C accept each other, and meanwhile, a session key SK is securely established between them (note SK is unknown to the device(s) and the attacker). This session key usually is further used to build a secure channel between C and S. For this authentication, the server S and the device(s) D_i ($i \in I$) need to generate and store long-lived secrets on themselves during the registration phase. But the client C is not allowed to store any long-lived secrets, since the user is allowed to leverage arbitrary clients for login (which may be different from the one used in the registration).

Besides, we assume there is an authenticated and secure channel between the client C and the device D_i. For a smartphone as the device, the channel can be established by Bluetooth or other means (e.g., QR + Wi-Fi) proposed in [14]. Unlike our model, Jarecki et al. assume the communication between C and D is on a public channel but require a t-bit Short Authenticated String (SAS) channel from C to D, where the message transmitted cannot be tampered by the attacker. This t-bit SAS channel assumption is suitable for some special devices embedded with a small LCD screen (e.g., RSA SecurID). But for smart devices (e.g., smartphones and smartwatches), we can directly assume an authenticated and secure channel between C and D_i.

Our model enables the parties to parallel run different instances (also called sessions) of the protocol Π. This is important to capture the security of Π in the context of parallel running. We use P^i to denote the i-th instance of a party P.

Partnering instances. We use the session id (SID) to define the partnering of the instances. More specifically, each instance of C or S outputs a SID sid, a partner id (PID) pid with the session key SK when it accepts.

Definition 1 (Partnering)**.** C^i and S^j are partners, if both of them accept with SID sid, sid' and PID pid, pid', respectively and the following requirements hold:

1) C^i and S^j output the same SID $sid = sid'$. This means their interaction transcripts are matching and the attacker does not launch active attacks.
2) C^i outputs the PID $pid = S$ and S^j outputs the SID $pid' = C$. This means they accept each other after a successful authentication.

Attacker ability. Since the channel between the client and the server is public, the attacker is allowed to fully control it. This means that *the attacker can overhear, intercept, and synthesize any message on this channel (as being characterized in Dolev–Yao model)*.

Besides, *the attacker is also allowed to corrupt any participants* (except the user) of the protocol and fully control the participants. We here consider the *strong corruption*, i.e., when corrupting a participant, the attacker gets its long-lived secrets and internal states (e.g., the random numbers if they are not be erased at the moment). Some studies only consider the *weak corruption*, where the attacker can only get the long-lived secrets without the internal states. We, in this paper, also cover this corruption. To distinguish these two types of corruptions, we use "$\mathbf{Corrupt}$" to denote the strong corruption operator and "$\mathbf{Compromise}$" for the weak one.

For the authentication factors, we also allow the attacker to compromise them. This modeling is to capture the cases where the attacker steals the factors, e.g., getting the password via shoulder surfing attacks.

Security definition. Our security model is game-based, defining the security by a game within the attacker. The attacker's ability is formally modeled by means of queries and responses. The attacker tries to win the game by these queries, where the winning is defined to break the protocol. *The protocol is (defined to be) secure, if arbitrary attackers cannot win the game with an advantage (or a probability) larger than a given bound.*

For a T-MFAKE protocol Π, the attacker's goal is to compromise the established session keys, more specifically, obtaining any partial information about the session keys. Therefore, our game requires the attacker to distinguish the (real) session key and a random number (of the same length). If failing to tell the difference, then she knows nothing about the real session key, and otherwise, we say that she breaks Π.

In the following, we formally describe the game with the attacker's queries for the T-MFAKE protocol Π. In the game, the registration process of Π is executed first, then the attacker can make the queries:

1) $\mathbf{Send}(P, i, Q, M)$: Execute Π as the instance P^i of the party P getting the message M from Q, and respond the response message of P^i to the attacker. Note that M can be a special message $\mathbf{Init}$ with $Q = \perp$. If $M = \mathbf{Init}$, initialize P^i and respond the first message(s) of P^i to the attacker.
2) $\mathbf{Reveal}(P, i)$: If P^i has accepted, then respond its session key. Otherwise, respond $\perp$.
3) $\mathbf{ESReveal}(P, i)$: Respond the ephemeral secret of P^i.
4) $\mathbf{Corrupt}(P)$: Allow the attacker to fully control P. Note that as mentioned before, the internal states and the long-lived secrets are given to the attacker.
5) $\mathbf{Compromise}(P, U)$ (where $P = S$, D_i, `PW` or `Bio`): Allow the attacker to get the long-lived secrets on P about U or directly steal U's factors.
6) $\mathbf{Test}(P, i)$: If P^i has accepted, flip a coin b.

a) If $b = 1$, respond the (real) session key of P^i;
b) If $b = 0$, randomly generate a number with the same length of the session key and respond the number.

This query does not capture the attacker's ability. Instead, it is used to challenge the attacker to know the partial information about the real session key. *Note that this* $\mathbf{Test}(\cdot,\cdot)$ *query can only be made once.*

At the end of the game, the attacker $\mathcal{A}$ needs to output a guess b' for the coin b. If and only if $b' = b$, the attacker wins the game. The advantage of $\mathcal{A}$ is defined as

$$\mathsf{Adv}_{\Pi}^{\text{t-mfake}}(\mathcal{A}) = |2\Pr[b' = b] - 1|,$$

and max advantage is denoted as

$$\mathsf{Adv}_{\Pi}^{\text{t-mfake}} = \max_{\mathcal{A}} \mathsf{Adv}_{\Pi}^{\text{t-mfake}}(\mathcal{A}).$$

Note that the attacker can trivially get the session keys of some instances, e.g., by making **Reveal** or **Corrupt** queries to the instances. Therefore, we only allow the attacker to **Test** a *fresh* instance, which is expected to be secure in our model. The freshness is formally defined as follows.

Definition 2 (Freshness)**.** An instance P^i is *fresh*, if $\mathbf{Reveal}(P,i)$ and $\mathbf{Reveal}(Q,j)$ were not made, where Q^j is the partner instance of P^i (if it exists), and one of the following conditions holds:

1) None of the queries $\mathbf{Corrupt}(C)$, $\mathbf{Corrupt}(S)$, $\mathbf{Compromise}(S,U)$ was made, and meanwhile at least one factor of the user is honest (not corrupted and not compromised).
2) The internal states of P^i and Q^j (if exists) are not compromised, and no rogue $\mathbf{Send}(P,i,Q,\cdot)$ queries were made.

The attacker can passively deliver the messages among the instances by the **Send** queries. She also can actively intercept and synthesize the messages. We say the **Send** queries in the second case are *rogue*.

Definition 3 (T-MFAKE)**.** A (t,n) *T-MFAKE* Π is secure, if for a uniform password distribution on a dictionary of size n, an arbitrary probabilistic polynomial time (PPT) attacker $\mathcal{A}$, and the security parameter κ, the advantage of $\mathcal{A}$ is bounded as follows:

1) If $t-1$ factors without the password are compromised (or corrupted):
   a) If S is not compromised,
   $$\mathsf{Adv}_{\Pi}^{\text{t-mfake}}(\mathcal{A}) \le \frac{1}{n}(q_C + q_S) + \mathsf{negl}(\kappa),$$
   where, q_C (resp. q_S) denotes the number of rogue $\mathbf{Send}(C,\cdot,S,\cdot)$ (resp. $\mathbf{Send}(D,\cdot,C,\cdot)$) queries that the attacker made, $\mathsf{negl}(\kappa)$ denotes a negligible amount in κ.
   b) If S is compromised or corrupted,
   $$\mathsf{Adv}_{\Pi}^{\text{t-mfake}}(\mathcal{A}) \le \frac{1}{n}q'_S + \mathsf{negl}(\kappa).$$
   Here q'_S denotes the number of (offline) operators that the attacker made on S's long-lived secrets.
2) Otherwise,
   $$\mathsf{Adv}_{\Pi}^{\text{t-mfake}}(\mathcal{A}) \le \mathsf{negl}(\kappa).$$

In the definition, we assume the password follows the uniform distribution for simplification. This can be naturally extended to an arbitrary password distribution. For a distribution with the cumulative probability function f, the terms $\frac{1}{n}(q_C + q_S)$ and $\frac{1}{n}q'_S$ in the bounds should be $f(q_C + q_S)$ and $f(q'_S)$, respectively.

Explanations of the bounds. In Case 2, a secure T-MFAKE protocol achieves the same security as a secure AKE protocol. Therefore, the bound is negligible as AKE security. In Case 1, the T-MFAKE protocol achieves the same security as a secure PAKE protocol. Specially, in Case 1a, the attacker can carry out an unavoidable online password guessing attack. If the attacker successfully guesses the password, then she gets the session key (to win the game). For each guess, the attacker needs to make a rough **Send** queries and has $\frac{1}{n}$ probability to guess the right password, which defines the security bound. In Case 1b, the attacker can launch an unavoidable offline password guessing attack. Similarly, for each offline guess, the attacker should do the corresponding offline operators, which gives the bound.

2FA security. To facilitate understanding the definition of T-MFAKE security, we here present the 2FAKE security degenerated from Definition 3. Note that in 2FAKE, there only exists one device denoted as D.

Definition 4 (2FA security)**.** A 2FAKE protocol Π is secure, if for a uniform password distribution on a dictionary of size n, an arbitrary probabilistic polynomial time (PPT) attacker $\mathcal{A}$, and the security parameter κ, the advantage of $\mathcal{A}$ is bounded as follows:

1) If D is not corrupted,
   $$\mathsf{Adv}_{\Pi}^{\text{2fa}}(\mathcal{A}) \le \mathsf{negl}(\kappa).$$
2) If only D is corrupted (or compromised),
   $$\mathsf{Adv}_{\Pi}^{\text{2fa}}(\mathcal{A}) \le \frac{1}{n}(q_C + q_S) + \mathsf{negl}(\kappa).$$
3) If only D and S are corrupted (or compromised),
   $$\mathsf{Adv}_{\Pi}^{\text{2fa}}(\mathcal{A}) \le \frac{1}{n}q'_S + \mathsf{negl}(\kappa).$$

Here, q_C, q_S, q'_S are the same as in Definition 3.

This definition for 2FA security is similar to that given in [13]. If the attacker does not compromise D, then she cannot impersonate the user or the server except a negligible probability; otherwise, she inevitably can carry out online password guessing attacks (only one time per session) to perform impersonation with a non-negligible probability. Further, if D and S are compromised at the same time, the attacker inevitably can launch offline password guessing (as many times as she prefers to) to obtain the password plaintext with a non-negligible probability, so that she can impersonate the user with the password and D's long-term secret. Note she is also able to impersonate the server via the S's long-term secret.

Support for fuzzy factors. In the above T-MFA security, we require each factor except the password can provide to a high-entropy cryptographic key, e.g., a smartphone (storing a cryptographic key). However, this does not work for fuzzy factors. The factors, in practice, may not have sufficient entropy and can be easily cracked like a simple password. More importantly, their readings may have some noise, which leads to *false acceptances and false rejections* (also called *false positives and false negatives*) with a small probability.

Therefore, the security definition (more specifically, the security bound) must be revised to adapt fuzzy factors. Here we only consider one fuzzy factor. If there are multiple fuzzy factors, we will combine them into one factor with a smaller probability of false acceptances and false rejections (e.g., by multimodal machine learning for biometrics [41], [42]), instead of using them separately.

Definition 5 (T-MFAKE with a fuzzy factor)**.** This definition is the same as Definition 3, except the modified security bounds. For a uniform password distribution on a dictionary of size n, a fuzzy factor with the min-entropy $H_{\min}$ and the probability p_{false} of false acceptances, an arbitrary PPT attacker $\mathcal{A}$, and the security parameter κ, the advantage of $\mathcal{A}$ is bounded as follows:

1) If S is not compromised,
$$\mathsf{Adv}_{\Pi}^{\text{t-mfake}}(\mathcal{A}) \leq p_{\text{c}}(q_C + q_S) + \mathsf{negl}(\kappa).$$
2) If S is compromised or corrupted,
$$\mathsf{Adv}_{\Pi}^{\text{t-mfake}}(\mathcal{A}) \leq p_{\text{c}} q'_S + \mathsf{negl}(\kappa).$$

Here, q_C, q_S, q'_S are the same as in Definition 3, and p_{c} is defined as follows:

1) If $t-2$ factors without the password and the fuzzy factor are compromised (or corrupted):
$$p_{\text{c}} = \frac{1}{n}(\frac{1}{2^{H_{\min}}} + p_{\text{false}}).$$
2) If $t-1$ factors without the password or the fuzzy factor or both are compromised (or corrupted):
   a) If the password is compromised but not the fuzzy factor:
   $$p_{\text{c}} = \frac{1}{2^{H_{\min}}} + p_{\text{false}}.$$
   b) If the fuzzy factor is compromised but not the password:
   $$p_{\text{c}} = \frac{1}{n}.$$
   c) If none of the fuzzy factor and the password are compromised:
   $$p_{\text{c}} = \max\{\frac{1}{n}, \frac{1}{2^{H_{\min}}} + p_{\text{false}}\}.$$
3) Otherwise,
$$p_{\text{c}} = 0.$$

Note that the fuzzy factor can be guessed as the password. We use the min-entropy $H_{\min}$ to bound the probability of one guess for the fuzzy factor. Recall that the min-entropy $H_{\min}(X)$ of a random variable X is defined as
$$H_{\min}(X) = -\log_2 \max_{x \in \text{Range}(X)} \Pr[x].$$
Therefore, $\Pr[x] \leq \frac{1}{2^{H_{\min}(X)}}$, for $x \in \text{Range}(X)$. For the fuzzy factor, the probability that a guess is correct is not more than $\frac{1}{2^{H_{\min}}}$. Besides, there is a small probability p_{false} of false acceptances. Therefore, the cracked probability of each guess is not more than $\frac{1}{2^{H_{\min}}} + p_{\text{false}}$.

If compromising sufficient factors, the attacker can make online guessing for the password or the fuzzy factor or both to compromise the session keys. If the server is also corrupted, offline guessing becomes possible. We use p_{c} to denote the cracked probability of one guess (for the password, the fuzzy factor or both if necessary). For guessing the password and the fuzzy factor,
$$p_{\text{c}} = \frac{1}{n}(\frac{1}{2^{H_{\min}}} + p_{\text{false}}).$$
For guessing the password,
$$p_{\text{c}} = \frac{1}{n}.$$
For guessing the fuzzy factor,
$$p_{\text{c}} = \frac{1}{2^{H_{\min}}} + p_{\text{false}}.$$
For guessing the password or fuzzy factor (since the attacker can choose one of them for guessing),
$$p_{\text{c}} = \max\{\frac{1}{n}, \frac{1}{2^{H_{\min}}} + p_{\text{false}}\}.$$
If there are not enough factors that are compromised, the guessing attack cannot be carried out. In this case, we let
$$p_{\text{c}} = 0.$$
With the above definition of p_{c}, we complete all the security bounds for T-MFAKE with a fuzzy factor.

5 Our Threshold MFAKE Protocol

In this section, we first propose an efficient T-MFAKE protocol and formally prove its security in our model.

5.1 A Variant of TOPRF

The main component of our T-MFAKE protocol is the TOPRF. We do not perform a direct use of the TOPRF but require its specific variant. Specifically, in the variant with n parties, running PRF needs a fixed party and arbitrary $t-1$ parties from the rest (not arbitrary t parties).

This variant can be achieved by leveraging *access structure* schemes [43]. Access structure is generalized from the notion of threshold secret sharing. In the access structure, the party combinations for secret reconstruction can be freely specified. The only requirement is monotonicity, i.e., if a combination A can construct the secret, then any combination B including A can construct the secret. With an access structure scheme (e.g., [43]), setting our required party combinations can naturally construct a variant of TOPRF we need. However, this construction usually is inefficient. Instead, we use another method to achieve the required TOPRF variant. Informally, the fixed party holds half of the secret and each of the rest holds $\frac{1}{t-1}$ of the other half secret. From this, we can use a $(2,2)$ secret sharing and a $(t-1, n-1)$ secret sharing to construct the variant. Specifically, the secret key s of OPRF is divided to s_1, s_2 by a $(2,2)$ secret sharing, and further s_2 is divided to $\{s_{2i}\}_{i=1}^{n-1}$ by a $(t-1, n-1)$ secret sharing; the fixed party holds s_1, and each of the rest holds s_{2i}. To construct s for running PRF, s_1 is needed along with $t-1$ shares in $\{s_{2i}\}_{i=1}^{n-1}$. Therefore, the constructed variant of TOPRF satisfies our requirement.

5.2 Our T-MFAKE

As shown in Fig. 6, our T-MFAKE protocol is built on the top of TOPRF and AKE. It works as follows.

1) The user/client first leverages the above variant of TOPRF to enhance the password pw to a cryptographic key rw (also called random password), with the server (as the fixed party) and arbitrary $t-1$ devices. Note that the commutation between the client and the device is on a mutual-authenticated and secure channel.

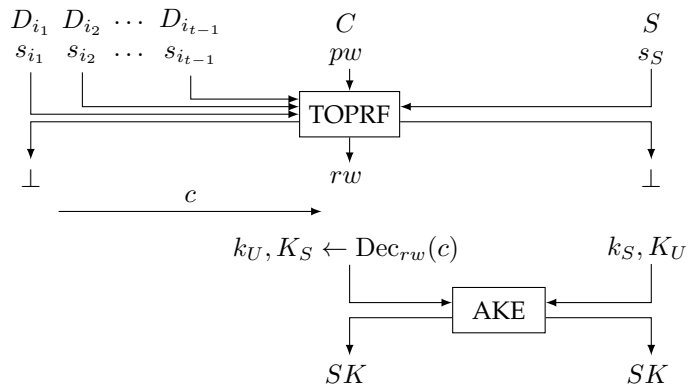


Fig. 6: Schematic diagram of our threshold MFAKE protocol

2) The user uses rw to decrypt the ciphertext c to get the keys k_U, K_S for AKE, where the ciphertext is stored on the devices and sent to the client.
3) The user runs AKE with the keys k_U, K_S with the server and outputs the session key of AKE, where k_U is the private key of the user and K_S is the public key of the server.

We use the 2HashTDH and HMQV to instantiate the TOPRF and AKE components. The details of our protocol are given in Fig. 7. With the instantiation, our T-MFAKE protocol only needs *two commutation rounds* by parallel running the TOPRF and AKE protocols. To make this point clear, we give the communication flows in Fig. 8.

Discussions on our design. The main challenge to achieve T-MFAKE security is to prohibit the password guessing attack as much as possible. A natural and potential way to achieve security is to leverage the technique of well-studied PAKEs (e.g., [34], [35], [36], [44]). In many PAKE protocols, we find the construction of OPAQUE [35], i.e., OPRF+AKE, is easily extended to T-MFAKE. In OPAQUE, the password is enhanced to a cryptographic key rw (in the OPRF part) and used for further authentication (in the AKE part). In our T-MFAKE protocol, we can further extend this enhancement by leveraging multiple devices to provide stronger security.

Informally, if less than $t-1$ devices are corrupted or compromised, the cryptographic key rw cannot be reconstructed even with the password, and therefore the attacker cannot do further authentication. In this case, our T-MFAKE protocol is secure as the AKE protocol. If $t-1$ devices are corrupted or compromised, our T-MFAKE protocol downgrades to OPAQUE (not precisely but closely). Therefore, in this case, our T-MFAKE protocol is secure as OPAQUE. More specifically, our T-MFAKE protocol only suffers from: 1) inevitable online password guessing attacks in the case where $t-1$ devices are corrupted or compromised; 2) offline password guessing attacks in the case where $t-1$ devices are corrupted or compromised and meanwhile the server is corrupted or compromised.

Note that if we use original TORPF instead of our variant, then the attacker compromising t devices can offline run the PRF without the help of the server, and further can carry out offline password guessing. In contrast, using our TOPRF variant prohibits offline password guessing except the server is also corrupted or compromised.

Parameter

- All parameters in 2HashTDH (our variant) and HMQV. To distinguish these parameters, we add the subscript 1 and 2 to parameters in 2HashTDH and HMQV, respectively, if necessary.
- An encryption scheme $(\mathrm{Enc}, \mathrm{Dec})$.

Initialization

- C generates the secret key s for 2HashTDH, generates the two shares s_D, s_S of s ($s = s_D + s_S$), send s_S to S; then generates shares s_{D_i} of s_D by the $(t-1, n-1)$ Shamir's secret sharing, and sends s_{D_i} to D_i, respectively; calculates $rw \leftarrow H_1(pw, H_1'(pw)^s)$.
- C generates the private and public keys (k_U, K_U) by $k_U \leftarrow\$ \mathbb{Z}_{m_2}$ and $K_U \leftarrow g_2^{k_U}$ for HMQV; gets S's public key K_S and encrypts (k_U, K_S) to c with the key rw ($c \leftarrow \mathrm{Enc}_{rw}(k_U, K_s)$); sends K_U to S and sends c to D_i ($1 \le i \le n-1$).
- D_i stores c and the share s_{D_i} for 2HashTDH.
- S generates the private and public keys (k_S, K_S) by $k_S \leftarrow\$ \mathbb{Z}_{m_2}$ and $K_S \leftarrow g_2^{k_S}$ for HMQV; sends K_S to C; stores K_U and the share s_S for 2HashTDH.

Authentication

- C picks $r \leftarrow\$ \mathbb{Z}_{m_1}$ and calculates $\alpha \leftarrow H_1'(pw)^r$; picks $t-1$ devices with index set I; picks $x \leftarrow\$ \mathbb{Z}_{m_2}$ and calculates $X \leftarrow g_2^x$; sends (U, X, α) to S and sends (U, S, α) to D_i ($i \in I$).
- Getting (U, S, α) from C, D_i calculates $\beta_{D_i} \leftarrow \alpha^{s_{D_i}}$ and sends (β_{D_i}, c) to C.
- Getting (U, X, α) from C, S picks $y \leftarrow\$ \mathbb{Z}_{m_2}$ and calculates $Y \leftarrow g_2^y$, $\beta_S \leftarrow \alpha^{s_S}$, $SK \leftarrow H_2((XK_U^{H_2'(X,K_S)})^{y+H_2'(Y,K_U)k_S})$; sends (Y, β_S) to C; outputs SK.
- Getting (Y, β_S) from S and (β_{D_i}, c) from D_i ($i \in I$), C calculates $rw \leftarrow H_1(pw, \beta_S^{1/r} \prod_{i \in I} \beta_{D_i}^{\lambda_i/r})$, $k_U, K_S \leftarrow \mathrm{Dec}_{rw}(c)$, $SK \leftarrow H_2((YK_S^{H_2'(Y,K_U)})^{x+H_2'(X,K_S)k_U})$; outputs SK. Here, λ_i is the Lagrange interpolation coefficient for i in I.

Fig. 7: Our threshold MFAKE protocol.

Support for fuzzy factors. Our T-MFAKE in Fig. 7 requires the password and $n-1$ devices as the n authentication factors. It is not suitable for users who only have smartphones without other devices. A practical way to provide strong security is to leverage biometric characteristics as authentication factors.

To support fuzzy factors (including biometric factors), we leverage fuzzy extractor to convert the fuzzy factor to a cryptographic key R, and use R as one share in TOPRF. Therefore, the fuzzy factor can replace one device in the authentication phase.

Note that the reconstruction of R needs the helper string P and fuzzy input. Therefore, P should be stored on the devices. If one device is corrupted or compromised, the attacker can get the helper string, and try to reconstruct the key by guessing fuzzy input. Excepts that, our T-MFAKE with a fuzzy factor achieves the same security as the original version.

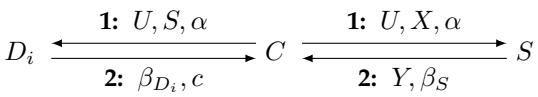


Fig. 8: The message flows of our 2FAKE within *only two* rounds

5.3 The Security of Our T-MFAKE Protocol

Since the security requirements of T-MFAKE are relatively complex, we will analyze the security step by step. Recall that 2FAKE and MFAKE are the special cases of T-MFAKE. Based on this, we first prove the security of our 2FAKE ($(2,2)$ T-MFAKE) protocol, then extend it for MFAKE and finally T-MFAKE.

Theorem 1. *Our 2FAKE ($(2,2)$ T-MFAKE) protocol (with a password pw and a device D as factors) is secure as in Definition 3. Specifically:*

1) If D is not corrupted,

$$\mathsf{Adv}^{\text{t-mfake}}_{Our2FAKE}(\mathcal{A}) \leq \mathsf{Adv}^{\text{ake}}_{HMQV} + q_C(\frac{1}{2^\kappa} + \mathsf{Adv}^{\text{auth}}_{\text{Enc}}).$$

2) If D is corrupted,

$$\mathsf{Adv}^{\text{t-mfake}}_{Our2FAKE}(\mathcal{A}) \leq \mathsf{Adv}^{\text{pake}}_{OPAQUE} :$$

a) If S is not corrupted,

$$\mathsf{Adv}^{\text{pake}}_{OPAQUE} \leq \frac{1}{n}(q_C + q_S) + \mathsf{Adv}^{\text{dis}}_{OPAQUE,\mathcal{F}_{SaPAKE}}.$$

b) Otherwise,

$$\mathsf{Adv}^{\text{pake}}_{OPAQUE} \leq \frac{1}{n}\min\{q'_S, q'_D\} + \mathsf{Adv}^{\text{dis}}_{OPAQUE,\mathcal{F}_{SaPAKE}}.$$

Proof. In the analysis, our 2FAKE protocol achieves different security bounds. We present the bound for each case, as we need to construct different games and reductions.

Case I (D is not corrupted). As discussed above, if D is not corrupted, the attacker does not know extra information about the key in the AKE protocol. Following this intuition, we reduce the security of our 2FAKE to the security of the AKE protocol. The main process of proof is to 1) slightly modify the real attack game G_0 (for 2FAKE) to G_1, and 2) reduce the game G^{AKE} for AKE to G_1.

Let G_1 simulate the messages of 2HashTDH without executing it. Specifically, G_1 is the same as G_0 except the following:

1) When C^i is initializing (getting $\mathbf{Send}(C, i, \perp, \mathbf{Init})$), pick $\alpha \leftarrow\$\, G$ and send it to the attacker as aimed at S. Meanwhile run HMQV.
2) When getting $\mathbf{Send}(C, i, S, \beta)$, check if $\beta = \alpha^{s_S}$. If not, abort C^i, otherwise, run HMQV.

G_0 and G_1 are indistinguishable except $\beta \neq \alpha^{s_S}$ but C^i does not abort. This only happens in G_0 when (the incorrect) β yields to the correct rw (a hash collision) or a successful decryption of c (breaking the authentication). Therefore,

$$|\Pr[\mathcal{A} \text{ wins in } G_0] - \Pr[\mathcal{A} \text{ wins in } G_1]| \leq q_C(\frac{1}{2^\kappa} + \mathsf{Adv}^{\text{auth}}_{\text{Enc}}).$$

Then we extend the game G^{AKE} for AKE to G_1 by simulating 2HashTDH (as above) and do the reduction to the AKE security. Leveraging the attacker $\mathcal{A}$ for G_1, we can naturally construct an attacker $\mathcal{A}'$ for G^{AKE}. $\mathcal{A}$ is the same as $\mathcal{A}'$ except the 2HashTDH part:

1) If $\mathcal{A}$ makes $\mathbf{Send}(C, i, S, (U, X, \alpha))$ query, then $\mathcal{A}'$ makes $\mathbf{Send}(C, i, S, (U, X))$ query.
2) If $\mathcal{A}$ makes $\mathbf{Send}(S, j, C, (Y, \beta_S))$ query, then $\mathcal{A}'$ makes $\mathbf{Send}(S, j, C, (U, Y))$ query.

If $\mathcal{A}$ wins in G_1, then $\mathcal{A}'$ will win in G^{AKE}. Therefore,

$$\Pr[\mathcal{A} \text{ wins in } G_1] \leq \Pr[\mathcal{A}' \text{ wins in } G^{\text{AKE}}] \leq \mathsf{Adv}^{\text{ake}}_{\text{HMQV}}.$$

Finally,

$$\mathsf{Adv}^{\text{t-mfake}}_{\text{Our2FAKE}}(\mathcal{A}) \leq \mathsf{Adv}^{\text{ake}}_{\text{HMQV}} + q_C(\frac{1}{2^\kappa} + \mathsf{Adv}^{\text{auth}}_{\text{Enc}}).$$

Case II (D is corrupted). If D is corrupted, our 2FAKE protocol is downgraded to OPAQUE. Based on this idea, we reduce the security of our 2FAKE to the security of the PAKE protocol. The main process of proof is to 1) modify the real attack game G_0 (for 2FAKE) to G_1, and 2) slightly reduce the game G^{PAKE} for PAKE to G_1. The process is similar to that of the first case, but the constructed game G_1 and the reduction are totally different.

We modify G_0 to G_1 by slightly changing the query to H. G_0 and G_1 are the same except:

1) If (x, y) is queried for H (by $\mathcal{A}$ or participants), return $H(x, y/H'(x)^{s_D})$.

Since both H and H' are random oracles, G_0 and G_1 are totally indistinguishable. Therefore, $\Pr[\mathcal{A} \text{ wins in } G_0] = \Pr[\mathcal{A} \text{ wins in } G_1]$.

Then we extend the game G^{PAKE} for PAKE to G_1 and do the reduction to the PAKE security. Specifically, G^{PAKE} is extended to G_1 by simulating the interaction of C and D as follows:

1) If (U, X, α) aimed at S is responded to the attacker in G^{PAKE}, then respond the same message as well as (U, S, α) aimed at D in G_1.

Leveraging the attacker $\mathcal{A}$ for G_1, we construct an attacker $\mathcal{A}'$ for G^{PAKE} which is the same as $\mathcal{A}$ except the following case:

1) If $\mathcal{A}$ makes $\mathbf{Send}(S, j, C, (Y, \beta_S))$ query and sends (β_D, c) to C as the corrupted D, then $\mathcal{A}'$ makes $\mathbf{Send}(S, j, C, (Y, \beta_S\beta_D/\alpha^{s_D}, c))$ query.

Note that C^i does the same calculation in G_0 with $\mathcal{A}$ as in G^{PAKE} with $\mathcal{A}'$. More specifically,

$$rw \leftarrow H(pw, (\beta_D\beta_S)^{1/r}/H'(pw)^{s_D})$$

in G_0 is equal to

$$rw \leftarrow H(pw, (\beta_D\beta_S/\alpha^{s_D})^{1/r})$$

in G^{PAKE}, since $\alpha = H'(pw)^r$ and $(\beta_D\beta_S)^{1/r}/H'(pw)^{s_D} = (\beta_D\beta_S/\alpha^{s_D})^{1/r}$. Therefore, if $\mathcal{A}$ wins in G_1, then $\mathcal{A}'$ wins in G^{PAKE}. Consequently,

$$\Pr[\mathcal{A} \text{ wins in } G_1] \leq \Pr[\mathcal{A}' \text{ wins in } G^{\text{PAKE}}] \leq \mathsf{Adv}^{\text{pake}}_{\text{OPAQUE}}.$$

Further,

$$\mathsf{Adv}^{\text{t-mfake}}_{\text{Our2FAKE}}(\mathcal{A}) \leq \mathsf{Adv}^{\text{pake}}_{\text{OPAQUE}}.$$

Due to the PAKE security of OPAQUE, $\mathsf{Adv}^{\text{pake}}_{\text{OPAQUE}}$ is bounded as explained in Section 3.3 (in the cases where S is corrupted or not). This gives the corresponding bounds for our 2FAKE protocol in these two cases. Note that for an online password guess, the attacker needs to make a rough $\mathbf{Send}(C, \cdot, S, \cdot)$ or $\mathbf{Send}(D, \cdot, C, \cdot)$ query, therefore, $q \leq q_C + q_S$; for an offline password guess, the attacker needs to make an offline operator on S's long-lived key as well as an operator on D's key, therefore, $q' \leq \min\{q'_C, q'_S\}$. □

Theorem 2. *Our MFAKE ((n, n) T-MFAKE) protocol (without fuzzy factors) is secure as in Definition 3.*

Our MFAKE protocol is the same as our 2FAKE variant, except a single device is expanded to multiple. If at least one device in MFAKE is honest, then the attacker cannot carry out effective attacks. Informally, the corruption of the device in 2FAKE corresponds to the corruption of all devices here. The proof for Theorem 2 can easily make use of the analysis result from Theorem 1, we thus omit it.

Theorem 3. *Our MFAKE ((n, n) T-MFAKE) protocol (with a fuzzy factor) is secure as in Definition 5.*

Proof. Due to the security of the fuzzy extractor, the cryptographic key can only be reproduced from the fuzzy input. Therefore, without the (right or close) fuzzy input, the attacker cannot get the random password and fail to distinguish the real session key with a random number. Besides, the attacker only has as most $\frac{1}{2^{H_{\min}}} + p_{\text{false}}$ probability to reproduce the key ($\frac{1}{2^{H_{\min}}}$ for guessing the correct input, p_{false} for the false acceptance). Consequently, we can achieve the corresponding bound in Definition 3.

In the following, we give a formal analysis. In the cases where one device factor is not compromised or corrupted, the proof is trivial. In the cases where the biometric factor is compromised or corrupted, the proof is similar to that for Theorem 2. We only consider other cases.

Let G_0 be the real attack game. G_1 is the same as G_0 except that the helper string P' generated by a randomly-generated key R' (not the right one R used in TOPRF) is given to the attacker instead of the real helper string P. Therefore, G_0 and G_1 are indistinguishable, except the attacker reproduces the right key R, reconstructs rw and further uses rw to decrypt c. If the password is compromised, then the probability of reconstruction is not more than

$$p_{\text{c}} = \frac{1}{2^{H_{\min}}} + p_{\text{false}}$$

for each guess. Otherwise, the attacker has to guess the password as well as the fuzzy input, therefore the probability of reconstruction is not more than

$$p_{\text{c}} = \frac{1}{n}(\frac{1}{2^{H_{\min}}} + p_{\text{false}})$$

for each guess.

In the case where S is not corrupted, each guess requires at least one (usually several) $\mathbf{Send}(C, \cdot, S, \cdot)$ or $\mathbf{Send}(C, \cdot, S, \cdot)$ query. Therefore, we have

$$|\Pr[\mathcal{A} \text{ wins in } G_0] - \Pr[\mathcal{A} \text{ wins in } G_1] \leq p_{\text{c}}(q_C + q_S).$$

Further, since $\Pr[\mathcal{A}$ wins in $G_1]$ is negligible (without knowing R),

$$\mathsf{Adv}^{\text{t-mfake}}_{\text{OurMFAKE2}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins in } G_0] \leq p_{\text{c}}(q_C + q_S) + \mathsf{negl}(\kappa).$$

In the case where S is corrupted, each guess requires at least one (usually several) offline operation on S's and on D's long-lived secrets. Similar to the above reasoning, we have

$$\mathsf{Adv}^{\text{t-mfake}}_{\text{OurMFAKE2}}(\mathcal{A}) \leq p_{\text{c}} q'_S + \mathsf{negl}(\kappa).$$

Therefore, our MFAKE is secure as in Definition 5. □

Theorem 4. *Our T-MFAKE without (or with) a fuzzy factor meets the requirements in Definition 3 (or 5).*

The analysis for our (t, n) T-MFAKE is also similar to that for our (t, t) T-MFAKE (with or without a fuzzy factor). We omit it here.

5.4 Extensions for Refreshment and Anonymity

Refreshment. Besides the advantages of usability, our T-MFAKE also brings benefits to security by the *refreshment*. Refreshment is a mechanism that periodically refreshes the shares in the threshold context to avoid massive (t) shares being compromised over time. If the attacker does not compromise t shares in the current period, then it needs to re-compromise them in the next period.

In our T-MFAKE, we refresh the secret keys in TOPRF without the password and updates the ciphertext c without changing the private key k_U. Specifically, the refreshment is designed as follows:

1) The client C generates a new secret key s' in TOPRF and each shares s'_i of s', sends the shares to the corresponding parties; calculates $rw' \leftarrow H_1(pw, H'_1(pw)^{s'})$, and encrypts k_U, K_S with rw' to c'; sends c' to the device(s).
2) The server S updates its share with the new one s'_n.
3) The device D_i updates its share and the ciphertext with the new ones s'_i and c'.

As well as to resist (factor) compromise by periodic executions, this refreshment can be used to *revoke* lost devices by active executions. The user only needs to refresh the keys without the lost devices like the above process.

Note that we do not require the user to change the password, which will bring an extra burden on memory and cannot achieve expected security (for instance, a new password is just slightly modified from the old one [21]).

Anonymity. In some scenarios (e.g., secret ballot), parties may choose to hide their usernames to protect privacy. We should provide user anonymity so that:

1) the attacker cannot identify the user ID in a session.
2) the attacker cannot tell if two sessions correspond to the same user.

Our 2FAKE and MFAKE do not provide anonymity like OpTFA [13], since our protocols directly send the username on the public channel. A trivial way to achieve user anonymity is to establish a server-authenticated secure channel first and then run our protocol on the channel. This requires a public key of the server to establish this channel, which can be stored on the device(s). However, this method requires two extra communication rounds.

For our design, there is a faster and simpler way to achieve anonymity - encrypting the username by a probability public encryption scheme (e.g., ElGamal). With the encryption, the username is only known by the server and two ciphertexts of the same username are different. Therefore, the attacker cannot extract any information about the username from that. Besides, the attacker cannot extract any information about or linked to the username from the other transmitted messages (as they are all random numbers). This method will only slightly require the encryption and decryption operations of the public encryption scheme without an extra round for communication.

5.5 Discussions on Security

Since our T-MFAKE protocol is secure as in Definition 3 or 5, it can resist ephemeral secret leakage, replay, man-in-the-middle, impersonation and privileged-insider attacks. Here we briefly present some discussions on these attacks and our design.

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/span/span/testsuite/results/tmfake.if
GOAL
  As Specified
BACKEND
  CL-AtSe
STATISTICS
  Analysed   : 64 states
  Reachable  : 64 states
  Translation: 0.01 seconds
  Computation: 0.00 seconds
```

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/tmfake.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.53s
  visitedNodes: 512 nodes
  depth: 9 plies
```

Fig. 9: Analysis results for our T-MFAKE using AVISPA with the CL-AtSe and OFMC back-ends

Ephemeral secret leakage attacks. The main component of our design, HMQV [33], can resist the attacks, and the protocol benefits from this security feature. More specifically, for the session key $SK = H_2((YK_S^{H_2'(Y,K_U)})^{x+H_2'(X,K_S)k_U})$ (calculated by the client) in HMQV, if the attacker gets the ephemeral secret x, she cannot calculate SK without the private key k_U. So the session key SK is safe. Even if capturing both x and SK, the attacker cannot calculate the private key k_U except breaking the hash function and solving the discrete logarithm problem. Thus, the keys of other sessions are kept safe. Further, if the attacker compromises the long-term secret k_U, these sessions without leaking the ephemeral secrets are still safe.

Replay attacks. For each session, the client is able to randomly choose an ephemeral exponent as well as the server, so that the attacks cannot be carried.

Impersonation attacks. As we proved previously, our protocol can resist impersonation attacks in arbitrary cases except the two trivial ones: 1) impersonating the user by compromising the user's $t-1$ factor and the password (where the password may be captured by online/offline guessing attacks); and 2) impersonating the server by compromising the server's long-term secret. In normal (but not exceptional) cases, without at least $t-1$ factor and the password, the attacker cannot impersonate the user to run TOPRF and therefore cannot execute SaPAKE to calculate the session key. Similarly, without compromising the server's long-term secret, the attacker also cannot obtain the session key by executing SaPAKE (as the server does).

Man-in-the-middle attacks. Based on our analysis on the impersonation attacks, the attacker can only launch man-in-the-middle attacks by compromising the credentials of both the user and the server.

Privileged-insider attacks. In the registration phrase, the user generates the shares of TOPRF and the private key on the client, but the server does not know the devices' shares of TOPRF and the user's private key. Thus, our protocol can resist privileged-insider attacks.

5.6 Automated Security Analysis with AVISPA

We leverage an automated validation tool, AVISPA (Automated Validation of Internet Security Protocols and Applications), to evaluate the security of our protocol. AVISPA can automatically check if a given protocol achieves the target security goals via running the protocol. It also can investigate potential security risks and further provide the attacking paths to help improve the protocol. Due to the advantages, it has been widely used in the study on authentication protocols [45], [46], [47].

We first describe our protocol and the security goals in High Level Protocol Specification Language (HLPSL), and then run AVISPA with two back-ends, namely CL-AtSe and OFMC. The analysis results, shown in Fig. 9, confirm that our protocol meets the security requirements.

6 Implementation and Performance

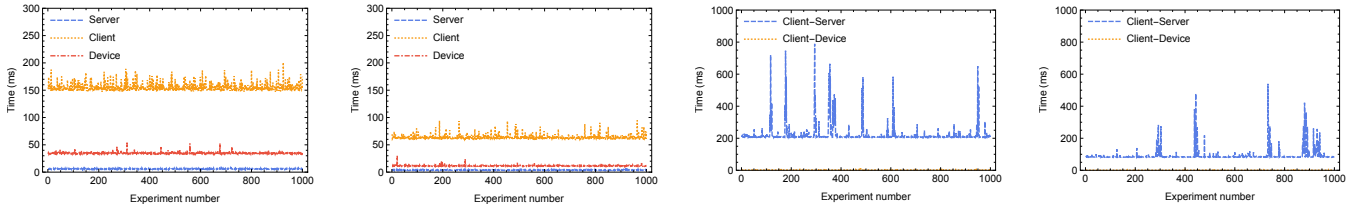
We implement our protocol and evaluate its performance on computation and communication. Here, we only implement a $(2,2)$ variant of our protocol. We note the (t,n) protocol only requires extra computation and communication cost (among extra devices), but does not increase the total running time of the protocol due to the parallel running.

6.1 Implementation Details

Participant. We need the following three parties in the implementation.

1) The server S in a Docker container running on a remote server in Alibaba Cloud. The Docker container is assigned with 2 CPUs (Intel Core i5-7300HQ CPU @ 2.50Hz 2.50Hz) and 2.0GB memory. In the container, Ubuntu 18.04 with Apache 2.0, PHP 7.4 and MySQL 7.4.8 is deployed.
2) The client C is a web browser, Chrome 85.0.4183.102 on a PC with Windows 10. The PC is equipped with an Intel Core i5-7300HQ CPU, 16.0GB memory and Bluetooth 5.0 adapter. To support our protocol on the client, we implement a Chrome Application to perform the calculation and communication.
3) The device D is a smartphone, Huawei P30 (ELE-AL00), with Huawei Kirin 980, 8.0GB memory, Bluetooth 5.0 adapter and Android 10. We also implement an APP on the device for the protocol.

Communication. The communication between the server and the client is on the Internet. The round trip time between them is approximately 80 ms. The client and the device are connected via the Bluetooth channel and meanwhile, their physical distance is less than half a meter when running the experiments. The round trip time of this channel takes around 1 ms.



(a) OpTFA: computation cost (b) Ours: computation cost (c) OpTFA: communication cost (d) Ours: communication cost

Fig. 10: Performance comparison

TABLE 2: Performance comparison

Protocol	Computation cost			Storage cost			Communication cost	
	Server	Client	Device	Server	Client	Device	Cline-Server	Client-Device
OpTFA	4 exp + 1 mexp 5.98 ms	9 exp + 1 mexp 155.16 ms	3 exp 34.65 ms	1280 bit	0 bit	512 bit	5 rounds 222.36 ms	5 rounds 2.36 ms
Ours	2 exp + 1 mexp 3.67 ms	3 exp + 1 mexp 63.58 ms	1 exp 11.55 ms	768 bit	0 bit	768 bit	2 rounds 93.38 ms	2 rounds 0.94 ms

[1] exp: number of exponentiation; mexp: number of multi-exponentiation.
[2] The communication on the two channels is run parallel in our protocol, but not in OpTFA.
[3] The client stores nothing, but the user needs to memorize the password.

Computation. The client, server, and device leverage SJCL, OpenSSL 1.1.1g, and javax.crypto to support the cryptography algorithms, respectively. To instantiate cryptography algorithms, we use SHA-256 for the hash operator, HMAC-SHA256 for HMAC, AES for symmetric encryption, Encrypt-then-MAC with AES and HMAC-SHA256 for authentication encryption, and NIST P-256 for exponentiation.

Storage. The user needs to remember the password, while the client does not store anything. Each device stores its shares of TOPRF and the ciphertext (of the user's private key and the server's public key). The server stores the shares of TOPRF, its private key and the user's public key. Note that we use the compressed form to represent an ECC point without the first 8 bits (0x04) indicating the compressed form [48].

User operations. A user first enters the username and the password on a PC - more specifically, on the web page of the server shown in the browser. Then he clicks the login button on the page, and gets a notice from the smartphone. After approving the login via a click on the APP installed on the smartphone, the user can log in the server.

6.2 Performance Evaluation

To present a fair and comprehensive performance evaluation, we run our protocol 1,000 times along with OpTFA for comparison (although there are some other MFA protocols, none of them achieve the strong security defined in this paper). Note that we here directly simulating the computation cost and ignore the cost a user spends in information entering and button clicking.

Fig. 10 shows the time cost for each experiment while Table 2 gives us a general picture of the average values. The experimental results show that our protocol achieves a significant improvement in efficiency as compared to OpTFA - 138.25% and 148.49% faster on communication and computation, respectively. Besides, our storage cost is quite close to that of OpTFA - 512 bits less cost on the server, and just 128 bits more on the device. From the experimental results, we conclude that our protocol is efficient and practical enough for real-world applications.

7 Conclusion

We propose a new notion of (t, n) threshold multiple-factor authentication (T-MFA), allowing users to autonomously choose t at-hand factors out of n for authentication. It brings advantages to usability and security. We also construct a T-MFA key exchange protocol and prove that it achieves the highest attainable security. Our protocol only requires $4 + t$ exponentiations, 2 multi-exponentiations and **2** communication rounds. Via the implementations and experiments, we show that our design is fast, secure and practical in the real-world applications.

Acknowledgment

This research is supported by National Key R&D Program of China (2020YFB1805400), National Natural Science Foundation of China (62072010), and European Union's Horizon 2020 research and innovation programme under grant agreement No. 952697 (ASSURED).

References

[1] EMVCo, "Contact EMV," https://www.emvco.com/emv-technologies/contact/, 2021.
[2] M. Bond, O. Choudary, S. J. Murdoch, S. Skorobogatov, and R. Anderson, "Chip and skim: cloning emv cards with the preplay attack," in *Proc. IEEE S&P 2014*, pp. 49–64.
[3] E. Remaley, "Ntia data reveal shifts in technology use, persistent digital divide," 2020, https://www.ntia.doc.gov/blog/2020/ntia-data-reveal-shifts-technology-use-persistent-digital-divide.
[4] T. Zhu, L. Fu, Q. Liu, Z. Lin, Y. Chen, and T. Chen, "One cycle attack: Fool sensor-based personal gait authentication with clustering," *IEEE Trans. Inf. Foren. Secur.*, vol. 16, pp. 553–568, 2020.
[5] T. Nakamura, V. Goverdovsky, and D. P. Mandic, "In-ear eeg biometrics for feasible and readily collectable real-world person authentication," *IEEE Trans. Inf. Foren. Secur.*, vol. 13, no. 3, pp. 648–661, 2017.
[6] P. MacKenzie, T. Shrimpton, and M. Jakobsson, "Threshold password-authenticated key exchange," in *Proc. CRYPTO 2002*. Springer, pp. 385–400.
[7] S. Agrawal, P. Miao, P. Mohassel, and P. Mukherjee, "Pasta: password-based threshold authentication," in *Proc. ACM CCS 2018*, pp. 2042–2059.
[8] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu, "Toppss: Cost-minimal password-protected secret sharing based on threshold oprf," in *Proc. ACNS 2017*. Springer, pp. 39–58.

[9] Google, "GitHub - google/google-authenticator: Open source version of Google Authenticator (except the Android app)," https://github.com/google/google-authenticator, 2020.

[10] CISCO, "Duo security two-factor authentication." https://goo.gl/wT3ur9., 2021.

[11] R. Zhang, Y. Xiao, S. Sun, and H. Ma, "Efficient multi-factor authenticated key exchange scheme for mobile communications," *IEEE Trans. Depend. Secur. Comput.*, vol. 16, no. 4, pp. 625–634, 2019.

[12] M. Wazid, A. K. Das, V. Odelu, N. Kumar, and W. Susilo, "Secure remote user authenticated key establishment protocol for smart home environment," *IEEE Trans. Depend. Secur. Comput.*, vol. 17, no. 2, pp. 391–406, 2020.

[13] S. Jarecki, H. Krawczyk, M. Shirvanian, and N. Saxena, "Two-factor authentication with end-to-end password security," in *Proc. PKC 2018*. Springer, pp. 431–461.

[14] M. Shirvanian, S. Jarecki, N. Saxena, and N. Nathan, "Two-factor authentication resilient to server compromise using mix-bandwidth devices." in *Proc. NDSS 2014*, pp. 1–16.

[15] J. Bonneau, C. Herley, P. C. Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proc. IEEE S&P 2012*, pp. 553–567.

[16] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Proc. EUROCRYPT 2004*. Springer, pp. 523–540.

[17] FIDO Alliance, "Universal 2nd factor (U2F) overview," https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.html, 2017.

[18] Z. Whittaker, "Github says bug exposed some plaintext passwords," https://www.zdnet.com/article/github-says-bug-exposed-account-passwords/, 2018.

[19] L. H. Newman, "Google has stored some passwords in plaintext since 2005," https://www.wired.com/story/google-stored-gsuite-passwords-plaintext/, 2019.

[20] "Have i been pwned?" https://haveibeenpwned.com.

[21] B. Pal, T. Daniel, R. Chatterjee, and T. Ristenpart, "Beyond credential stuffing: Password similarity models using neural networks," in *Proc. IEEE S&P 2019*, pp. 814–831.

[22] D. Pasquini, A. Gangwal, G. Ateniese, M. Bernaschi, and M. Conti, "Improving password guessing via representation learning," in *Proc. IEEE S&P 2021*, pp. 265–282.

[23] Y. Xiong, C. Su, W. Huang, F. Miao, W. Wang, and H. Ouyang, "Smartverif: Push the limit of automation capability of verifying security protocols by dynamic strategies," in *Proc. USENIX Security 2020*, pp. 253–270.

[24] M. Barbosa, G. Barthe, K. Bhargavan, B. Blanchet, C. Cremers, K. Liao, and B. Parno, "Sok: computer-aided cryptography," in *Proc. S&P 2021*, pp. 123–141.

[25] M. Wazid, A. K. Das, N. Kumar, A. V. Vasilakos, and J. J. Rodrigues, "Design and analysis of secure lightweight remote user authentication and key agreement scheme in internet of drones deployment," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3572–3584, 2018.

[26] J. Ni, K. Zhang, and A. V. Vasilakos, "Security and privacy for mobile edge caching: Challenges and solutions," *IEEE Wirel. Commun.*, 2020, dOI: 10.1109/MWC.001.2000329.

[27] B. Bera, S. Saha, A. K. Das, and A. V. Vasilakos, "Designing blockchain-based access control protocol in IoT-enabled smart-grid system," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5744–5761, 2021.

[28] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proc. EUROCRYPT 2000*. Springer, pp. 139–155.

[29] M. Abdalla, P.-A. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *Proc. PKC 2005*. Springer, pp. 65–84.

[30] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. IEEE FOCS 2001*, pp. 136–145.

[31] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu, "Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online)," in *Proc. EuroS&P 2016*, pp. 276–291.

[32] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Proc. CRYPTO 1993*. Springer, pp. 232–249.

[33] H. Krawczyk, "Hmqv: A high-performance secure diffie-hellman protocol," in *CRYPTO 2005*, pp. 546–566.

[34] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *Proc. IEEE S&P 1992*, pp. 72–84.

[35] S. Jarecki, H. Krawczyk, and J. Xu, "OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks," in *Proc. EUROCRYPT 2018*. Springer, pp. 456–486.

[36] M. Abdalla, M. Barbosa, T. Bradley, S. Jarecki, J. Katz, and J. Xu, "Universally composable relaxed password authenticated key exchange," in *Proc. CRYPTO 2020*. Springer, pp. 278–307.

[37] A. Juels and M. Sudan, "A fuzzy vault scheme," *Des. Codes, Cryptogr.*, vol. 38, no. 2, pp. 237–257, 2006.

[38] R. Canetti and H. Krawczyk, "Universally composable notions of key exchange and secure channels," in *Proc. EUROCRYPT 2002*, pp. 337–351.

[39] J. Srinivas, A. K. Das, M. Wazid, and N. Kumar, "Anonymous lightweight chaotic map-based authenticated key agreement protocol for industrial internet of things," *IEEE Trans. Depend. Secur. Comput.*, vol. 17, no. 6, pp. 1133–1146, 2018.

[40] S. Jangirala, A. K. Das, M. Wazid, and A. V. Vasilakos, "Designing secure user authentication protocol for big data collection in IoT-based intelligent transportation system," *IEEE Internet Things J.*, 2020, dOI: 10.1109/JIOT.2020.3040938.

[41] R. Snelick, U. Uludag, A. Mink, M. Indovina, and A. Jain, "Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 450–455, 2005.

[42] L. Wu, J. Yang, M. Zhou, Y. Chen, and Q. Wang, "Lvid: A multimodal biometrics authentication system on smartphones," *IEEE Trans. Inf. Foren. Secur.*, vol. 15, pp. 1572–1585, 2019.

[43] M. Ito, A. Saito, and T. Nishizeki, "Secret sharing scheme realizing general access structure," *Electron. Comm. Jpn. Pt. III*, vol. 72, no. 9, pp. 56–64, 1989.

[44] T. Bradley, S. Jarecki, and J. Xu, "Strong asymmetric pake based on trapdoor ckem," in *Proc. CRYPTO 2019*, pp. 798–825.

[45] S. Jangirala, A. K. Das, and A. V. Vasilakos, "Designing secure lightweight blockchain-enabled rfid-based authentication protocol for supply chains in 5g mobile edge computing environment," *IEEE Trans. Industr. Inform.*, vol. 16, no. 11, pp. 7081–7093, 2019.

[46] M. Wazid, A. K. Das, V. Bhat, and A. V. Vasilakos, "Lam-ciot: Lightweight authentication mechanism in cloud-based IoT environment," *J. Netw. Comput. Appl.*, vol. 150, p. 102496, 2020.

[47] S. Challa, A. K. Das, P. Gope, N. Kumar, F. Wu, and A. V. Vasilakos, "Design and analysis of authenticated key agreement scheme in cloud-assisted cyber–physical systems," *Future Gener. Comput. Syst.*, vol. 108, pp. 1267–1286, 2020.

[48] S. Turner, K. Yiu, R. Housley, D. R. Brown, and T. Polk, "RFC 5480 elliptic curve cryptography subject public key information," 2009.