UC-Secure Multi-Factor Authentication with Dynamic Password Recovery and Fine-Grained Access Control

Liufu Zhu and Ding Wang

***Abstract*—Multi-factor authentication (MFA) has been widely applied in various fields, including smart homes, autonomous driving, and mobile communication. Although a number of MFA schemes with different security goals and properties have been proposed, most of them are found to pay little attention to password forgetting and loss issues, which may lead to the permanent loss of the account. Additionally, little effort has been devoted to designing MFA schemes with fine-grained access control to perform authentication flexibly. Therefore, the above issues raise the question of *"how to construct a MFA scheme with dynamic password recovery and fine-grained access control?"*.**

In this paper, we, for the first time, introduce attributes and a dynamic password recovery method to propose a multi-factor authentication scheme with dynamic password recovery and fine-grained control, named MFA-DPRF. In MFA-DPRF, authentication succeeds only when a user provides both a valid password and a set of attributes satisfying the specified access policy, thereby enabling fine-grained access control. Furthermore, a dynamic password recovery method based on secret questions and the secret sharing technique has been designed to address password forgetting and loss issues. Users can not only recover the original password locally, but also update secret values such as security questions to enhance security. The security of MFA-DPRF can be reduced to the computational Diffie-Hellman problem under the Random Oracle Model. We also analyze the security of MFA-DPRF in the universally composable framework to ensure composable security. The informal analysis proves that MFA-DPRF is secure against known attacks. Compared with the state-of-the-art works, performance analysis shows that MFA-DPRF is superior in security and efficiency.

***Index Terms*—Multi-factor authentication; Password recovery; Fine-grained access control; UC security**

I. Introduction

With the rapid advancement of mobile communication and wireless sensor networks, digital technologies such as wireless communication, Internet of Things, and Internet of Vehicles have been extensively deployed across critical areas like e-commerce, industrial automation, and national defense. However, the widespread applications of these technologies have also introduced significant security vulnerabilities, including identity impersonation or phishing attacks, and sensitive data leakage. Therefore, how to ensure network security and avoid illegal access remains a critical challenge. Identity authentication, serving as the first line of defense for network security, plays a vital role in addressing the above issues [1], [2]. Generally, authentication factors can be categorized into three types: user-known information like passwords; user-unique biometrics like fingerprints; and user-owned physical information like smart cards. Multi-factor authentication (MFA) [3]–[7] has been introduced and thoroughly investigated during the past years, which employs two or more factors to prevent threats even if some authentication factors (not all of them) have been compromised.

As shown in Fig. 1, three entities are commonly involved in a MFA system, including the user UR, the authentication gateway AGW, and the server SR. Firstly, UR sends a registration request to AGW secretly, and then AGW generates partial smart card data and then returns it to UR. Secondly, when UR wants to communicate with SR, she transmits a login and authentication request to AGW and attempts to exchange a session key with SR later. Finally, a session key K will be established between UR and SR for futher communication.

Password remains one of the most widely used authentication factors in MFA schemes due to its ease of deployment and update [3], [5], [8]–[12]. Most of the existing schemes assume that passwords will not be forgotten or lost and require users to provide their original passwords during the login and password update phases. Due to the limited memory, people can only remember $5 \sim 7$ passwords correctly [13]. Nevertheless, on the one hand, more and more systems require users to set complex passwords to prevent password reuse and password guessing attacks [14]; on the other hand, the number of passwords created by users is becoming larger. According to research conducted by NordPass, the average number of passwords per individual has reached 168 [15], significantly increasing the likelihood of password forgetting across different systems.

To solve this problem, password recovery methods [16]–[19] have been introduced to help users recover their passwords through fallback authentication, improving the reliability of the authentication system. For MFA schemes, implementing password recovery involves performing fallback authentication with other factors (independent of the password), which should be easier to remember and secure in practical scenarios [20]. In addition, these fallback authentication

Received on September 14th, 2024; revised on April 14th, 2026; accepted on May 17th, 2026. This work is supported in part by Tianjin Natural Science Foundation Project (Grant No. 25JCJQJC00230) and by the Fundamental Research Funds for the Central Universities, Nankai University (Grant No. 63263258). (Corresponding author: Ding Wang.)

Liufu Zhu and Ding Wang are with the College of Cryptology and Cyber Science, Nankai University, Tianjin 300350, China, with the State Key Laboratory of Cryptology, Beijing 100878, China, with Key Laboratory of Data and Intelligent System Security, Nankai University, Ministry of Education, Tianjin 300350, China, with the Tianjin Key Laboratory of Network and Data Security Technology, Nankai University, Tianjin 300350, China; also with the Beijing Key Laboratory of Post-Quantum Cryptography, Beijing 100083, China. (Email: zhuliufu@mail.nankai.edu.cn; wangding@nankai.edu.cn)

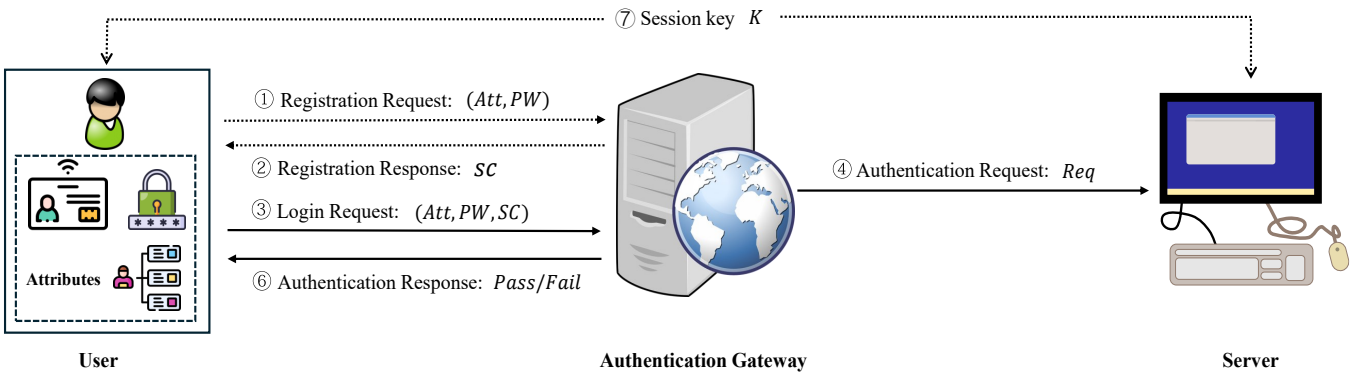

Fig. 1: The system framework of multi-factor authentication. Att, PW, and SC denote the attributes, password, and smart card, respectively. The dashed line represents the secure channel, while the solid line represents the public channel. K is the session key established between the user and server. $Pass$ or $Fail$ denotes the outcome of authentication. In addition, the serial number shows the order of message transmission.

factors should also be dynamically updated to avoid privacy leakage. As far as we know, little attention has been given to designing MFA schemes with dynamic password recovery.

In addition, existing MFA schemes such as [3], [9], [10], [12] fail to take fine-grained access control into consideration. They only verify whether a user is legitimate (e.g., by checking valid authentication factors), without evaluating whether the user is authorized to access specific resources. For example, in a streaming service subscription, traditional MFA only verifies that a user holds the valid account credentials, allowing any family member to access all content—including age-restricted material—without further authority restrictions. Attribute-based authentication [21]–[24] addresses this limitation by using attributes instead of explicit identities, enabling anonymous authentication with fine-grained access control through flexible policies. For instance, an age-based policy can verify whether a user meets a minimum age requirement without revealing the exact birthdate. However, to the best of our knowledge, little attention has been given to combining attributes with other factors to construct MFA schemes, making them unsuitable for scenarios that require both strong multi-factor security and fine-grained access control.

Above all, it remains an open problem of *how to construct MFA schemes with dynamic password recovery and fine-grained access control.*

A. Related Work

Since Lamport [25] proposed the first authentication scheme in 1981, considerable efforts have been made to propose various multi-factor authentication (MFA) schemes with varied security goals and desirable properties over the last 30 years. In 2006, Bhargav et al. [26] proposed a biometric-based MFA scheme that employs the password and biometrics to conduct authentication and applies zero-knowledge proof to hide actual biometrics. However, it cannot provide mutual authentication among the different parties. To address this issue, Huang et al. [27] introduced an improved MFA scheme, which not only provides mutual authentication but also achieves anonymity. However, a formal analysis of anonymity is absent, and it also fails to resist man-in-the-middle attacks.

To provide a formal analysis of anonymity, Wang et al. [3] summarized the limitations and challenges of achieving anonymity and introduced an evaluation criterion to measure the security and efficiency of anonymous schemes. To secure against man-in-the-middle attacks, Roy et al. [28] applied the Chebyshev chaotic mapping technique and proposed a scheme that creates random password credentials and resists man-in-the-middle attacks. However, it has expensive computational costs due to the application of expensive operations.

To solve this problem, Modarres et al. [29] proposed a lightweight MFA scheme that avoids complex operations, making it more suitable for resource-limited devices. However, the publicly transmitted message may leak biometric information. To address this issue, Zhang et al. [30] employed the fuzzy extraction method to introduce a MFA scheme to keep biometrics secret. With the advancement of quantum computing technology, traditional public-key cryptographic algorithms become vulnerable to quantum attacks. To resist this threat, Wang et al. [7], [31] proposed MFA schemes against quantum attacks while ensuring practical deployment in lightweight environments.

However, the above schemes only deploy authentication in single-user scenarios where users choose and store their authentication factors secretly, failing to provide authentication and fine-grained access control in multi-user scenarios. To address these issues, Guo et al. [21] proposed an attribute-based authentication scheme for mobile health networks, where users succeed in authentication as long as their attributes satisfy the access policy, but it lacks mutual authentication where users cannot check the validity of the authentication gateway.

To address this issue, Sucasas et al. [32] proposed a scheme that not only provides mutual authentication but also achieves key exchange. However, it fails to provide perfect privacy preservation for users. To solve this problem, Alpar et al. [33] introduced a novel attribute-based authentication scheme, achieving perfect privacy preservation and mutual authentication via zero-knowledge proof. To further protect privacy, Sucasas et al. [34] introduced an attribute-based anonymous authentication scheme that employs anonymous credentials and pseudonym techniques to avoid privacy leakage.

Although the above schemes achieve fine-grained control when users' attributes satisfy the access policy, they fail to ensure security and reliability in case of attribute leakage. To address this issue, Papadamou et al. [23] introduced a device and attribute-based authentication scheme that performs additional authentication to enhance security. However, it

requires users to be equipped with additional authentication devices, reducing the practicality of the scheme. To solve this problem, Song-Wang [35] presented an attribute-based password authentication scheme, ensuring that only users who share the same password and provide attributes satisfying the access policy can succeed in authentication.

Password-based authentication is widely adopted due to its ease of deployment and update [36]. However, due to the limitations of human memory, users can typically remember only 5 to 7 distinct passwords [13]. If multiple passwords are set simultaneously, it may result in password forgetting or loss issues. As far as we know, most schemes assume that the password will not be forgotten or lost, and they require users to resubmit their original passwords to conduct authentication and perform password updating. Therefore, these schemes fail to be deployed successfully in the password forgetting and loss scenarios. As pointed out by Weinshall et al. [37], users will be unable to access their accounts permanently if they forget or lose their passwords.

To address these issues, Ellison et al. [38] constructed a password recovery scheme based on the encryption algorithm, where the password is encrypted locally and can be decrypted by users who own the secret key. According to their scheme, the secret key can be reconstructed through the correct answers to secret questions via the secret sharing method. While Ellison et al. [38]'s method achieves password recovery to help users find their password, it fails to consider the security guaranteed by underlying secret questions. To tackle this problem, Just et al. [39] evaluated the security of secret questions and found that many user-selected questions have a low entropy, making them susceptible to guessing attacks.

Bonneau et al. [40] and Golla-Durmuth [41] identified vulnerabilities in secret question-based authentication through comprehensive analyses of large-scale, real-world datasets. Their studies provided critical empirical insights and design recommendations for account recovery mechanisms. Nevertheless, requiring users to recall all secret answers may impose an impractical memory burden, undermining the feasibility of such recovery methods. To solve this problem, Li et al. [19] introduced a password recovery method based on email and short messages, but it fails to provide a formal analysis under takeover or phishing attacks.

Although several password recovery methods have been proposed to address password forgetting and loss issues, few of them support dynamic password recovery—a feature that allows users to update recovery credentials and secret values to prevent privacy leakage. Moreover, these methods are vulnerable to secret question guessing attacks, where an adversary can compromise the recovery process by exploiting easily guessable or publicly available personal information. Given the multi-factor security of MFA schemes, the password recovery method must satisfy stronger security requirements, such as resistance to factor compromise. As a result, *the above password recovery methods cannot be directly deployed in MFA schemes.* Therefore, designing a dynamic password recovery method for MFA schemes remains an open challenge, which serves as the main motivation of this work.

B. Motivations

Password is the most widely used authentication factor in constructing MFA schemes (see [29], [42]–[44]) because of easy modification and deployment [36]. However, user-chosen passwords tend to follow Zipf's law [45], [46], which makes highly susceptible to guessing attacks. To reduce this threat, an increasing number of systems enforce stringent password policies that require a combination of uppercase and lowercase letters, digits, and special symbols [14]. Although implementing a complex password policy can enhance security, it may simultaneously increase the burden on users' memory, potentially resulting in password forgetting and loss issues [47]. Additionally, with the increase in the number of various websites, the average number of passwords owned by per person is also increasing, which is 168 in 2024. (see the research conducted by NordPass [15]).

Some password recovery methods have been proposed to address the password forgetting and loss issues. However, these methods cannot be directly deployed in multi-factor authentication schemes. On the one hand, existing password recovery methods (see [16], [17], [19]) require users to interact with the server and store recovery credentials on the server side. These credentials may contain some sensitive information, such as secret questions, biometrics, and email addresses, which may result in privacy leakage. On the other hand, existing password recovery methods (see [48]) fail to update secret information, such as secret questions and secret keys. Dynamically updating the above secret information can reduce the risk of privacy leakage and improve the security and reliability of MFA schemes.

In MFA schemes, recovery relies on fallback authentication when the password is unavailable. As noted by Lassak et al. [20], fallback authentication factors should be easy to remember and secure. Existing password recovery methods primarily rely on secret questions or SMS/email-based verification. However, these methods fail to consider malicious behaviours like secret question guessing attacks and SMS/email spoofing attacks, and thus cannot ensure security. Therefore, achieving secure and dynamic password recovery in multi-factor schemes deserves further research.

Furthermore, most existing MFA schemes (see [3], [5], [43], [49]) are introduced for single-user applications where factors are generated and kept by the user secretly. There are several concerns raised when implementing these schemes. One the one hand, it may leak user's privacy information such as identity and biometrics, thus failing to provide privacy protection. One the other hand, using authentication factors bound to the individual user is unable to provide fine-grained access control through a flexible access policy. For example, users can pass the authentication if she can provide a threshold of valid factors. In contrast, attribute-based cryptography enables fine-grained access control by allowing authentication only when a user's attributes satisfy a predefined access policy.

As far as we know, only Song-Wang's scheme [35] at IEEE TIFS'24 is the closest to this work. They presented an attribute-based password authentication scheme, ensuring that users who pre-share a password and provide valid attributes can exchange a session key with each other. However, Song-

Wang's work is designed for peer-to-peer model and is based on the assumption that the password will not be forgotten. In addition, they fail to define and prove their protocol in the UC framework, thus failing to provide composable security. Above all, it is urgent to solve the problem *how to construct a UC-secure MFA scheme with dynamic password recovery and fine-grained access control*.

C. Contributions

To address these challenges, we propose MFA-DPRF, a UC-secure multi-factor authentication scheme that simultaneously supports dynamic password recovery and fine-grained access control. The proposed scheme employs attributes as one of the authentication factors, ensuring only users who provide valid attributes, the correct password and the valid smart card can pass the authentication. It also provides dynamic password recovery, allowing users to recover their passwords locally and update recovery credentials securely. The specific contributions are as follows:

(1) We propose a UC-secure multi-factor authentication scheme with dynamic password recovery and fine-grained access control, named MFA-DPRF. It guarantees that only users who provide the correct password, the valid smart card, and attributes satisfying the access policy can pass authentication.

(2) We introduce a dynamic password recovery method that allows users to recover their passwords locally without server interaction. The method leverages secret questions as fallback authentication, which are bound to secret shares of a decryption key using a linear secret sharing scheme. The ciphertext of the password is stored in the smart card, so it can recover the password locally. Moreover, MFA-DPRF supports dynamic updates of secret values, minimizing the risk of privacy leakage. To resist password and secret question guessing attacks, it employs the fuzzy verifier technique to mix up the distribution of passwords and answers, and then provides real-time detection of malicious behavior.

(3) We prove the security of MFA-DPRF in the random oracle model, reducing its security to the computational Diffie-Hellman assumption. Furthermore,we further define an ideal functionality F_{MFA} and prove that the proposed scheme securely realizes it in the universally composable framework. Informal security analysis demonstrates that MFA-DPRF resists various known attacks, including password guessing, impersonation, insider, node capture, and replay attacks. Performance evaluation shows that MFA-DPRF achieves robustf security and efficiency compared to existing state-of-the-art MFA schemes.

D. Paper Organization

Section II introduces the preliminary knowledge, including the monotonic access structure, linear secret sharing scheme, the oblivious pseudorandom functionality, the authenticated encryption, the adversary model, and systematic evaluation criteria; Section III presents the construction of the proposed scheme; Section IV, Section V, Section VI, and Section VII provide the formal and informal analysis; Section VIII presents a performance analysis; Section IX concludes this paper.

II. Preliminaries

This section outlines the cryptographic primitives and techniques employed in MFA-DPRF, such as monotonic access structures, LSSS-based secret sharing, honeywords method, and adversary models.

A. Monotonic Access Structures

Let U represent the set of all attributes and $P(U)$ denote the power set of U, i.e., the set of all possible subsets of U. Define an access structure as any non-empty subset A of $P(U) \setminus \{\emptyset\}$. Within this framework, subsets belonging to A are termed authorized sets, whereas those not included in A are referred to as unauthorized sets. A is defined as a monotone access structure if it maintains the property of authorization under subset inclusion. Specifically, if B is an authorized set (i.e., $B \in A$), then any superset C of B (where $C \supseteq B$ and $C \in P(U)$) is also authorized, that is, $C \in A$.

B. LSSS-Linear Secret Sharing Scheme

Define U as a universe of attributes. A linear secret-sharing scheme Π_A over U for access structure A, operating within $\mathbb{Z}_p$, encompasses two polynomial-time algorithms. The method utilizes an $l \times k$ share-generating matrix M and a row labeling function $\rho : [l] \to I_U$ that assigns each row of M to an attribute in A, where I_U represents the index of U.

1 **Distribute** (M, ρ, α)**:** This algorithm, given M, ρ, and a secret $\alpha \in \mathbb{Z}_p$, randomly selects $z_2, z_3, \ldots, z_k \in \mathbb{Z}_p$ and forms the vector $v = (\alpha, z_2, z_3, \ldots, z_k) \in \mathbb{Z}_p^k$. It generates l shares as $\{M_i \cdot v : i \in [l]\}$, where M_i, the i-th row of M, spans $\mathbb{Z}_p^k$. Each share $\lambda_{\rho(i)} = M_i \cdot v$ is associated with attribute $\rho(i)$.

2 **Reconstruct**(M, ρ, Att)**:** Accepting M, ρ, and a set of attributes $Att \in A$, this algorithm identifies $I = \{i \in [l] : \rho(i) \in I_{Attr}\}$, where I_{Att} indexes Att. It outputs $\{attr_i : i \in I_{Att}\}$, ensuring $\sum_{i \in I_{Att}} \lambda_{\rho(i)}$ is a valid share set of α under Π_A.

The access structure is characterized by the target vector $(1, 0, \ldots, 0)$, meaning $Att \in A$ iff $(1, 0, \ldots, 0)$ lies in the linear span of rows of M indexed by Att.

Lemma 1. Given (M, ρ) realizing A over U, where M is an $l \times k$ share-generating matrix, for any $Att \subset U$ such that $Att \in A$, a polynomial-time algorithm exists that outputs a vector $w = (-1, w_2, \ldots, w_k) \in \mathbb{Z}_p^k$ such that $M_i \cdot w = 0$ for each i where $\rho(i) \in I_{Att}$, with I_{Att} as the index set for Att. Att satisfies the access structure A (or equivalently, A accepts Att) if and only if Att is an authorized set within A, explicitly $Att \in A$.

C. Oblivious pseudorandom functionality (OPRF) [50], [51]

A pseudorandom function (PRF) is an efficiently computable keyed function $f_k(\cdot)$ whose values are indistinguishable from random elements in the function range for a randomly chosen key k. The oblivious PRF (or OPRF) is a

Functionality F_{OPRF}

The functionality F_{OPRF} is parameterized by a security parameter κ. It interacts with the user $\mathcal{UR}$, the server $\mathcal{SR}$, and the simulator $\mathcal{SIM}$ via the following queries. For each server $\mathcal{SR}$, a table $T(\mathcal{SR},\cdot)$ is initialized to empty.

- Upon receiving a query $(\text{EVAL}, sid, \mathcal{SR}, x)$ from $\mathcal{UR}$ (or $\mathcal{SIM}$), record $(\text{evalrec}, sid, \mathcal{UR}, \mathcal{SR}, x)$ (or $(\text{evalrec}, sid, \mathcal{SIM}, \mathcal{SR}, x)$), and send $(\text{EVAL}, sid, \mathcal{UR}, \mathcal{SR}, |x|)$ (or $(EVAL, sid, SIM, \mathcal{SR}, x)$) to $\mathcal{SIM}$.
- Upon receiving a query $(\text{SNDRCOMPLETE}, sid, \mathcal{SR})$ from $\mathcal{SIM}$, increment $tx(\mathcal{SR})$ (or set to 1 if previously undefined) and send $(\text{SNDRCOMPLETE}, sid, \mathcal{SR})$ to $\mathcal{SR}$ or to $\mathcal{SIM}$ if $\mathcal{SR}$ is not a server's identity.
- Upon receiving the query $(\text{RCVCOMPLETE}, sid, \mathcal{UR}, \mathcal{SR}, \mathcal{SR}^*)$ from $\mathcal{SIM}$, recover the record $(\text{evalrec}, sid, \mathcal{UR}, \mathcal{SR}, x)$. If there exists no such record or $\mathcal{SR}$ is honest and $\mathcal{SR}^* \neq \mathcal{SR}$ or $tx(\mathcal{SR}^*) = 0$ (or undefined), abort the session. Otherwise, decrement $tx(\mathcal{SR}^*)$ and do the following:
  - ▷ If $T(\mathcal{SR}^*, x)$ is defined, then send $(\text{EVALCOMPLETE}, sid, \mathcal{SR}/\mathcal{SR}^*, T(\mathcal{SR}^*, x)))$ to $\mathcal{UR}$.
  - ▷ Else, choose $sk \in_R \{0,1\}^l$, set $T(\mathcal{SR}^*, x) = sk$ and send $(\text{EVALCOMPLETE}, sid, \mathcal{SR}/\mathcal{SR}^*, T(\mathcal{SR}^*, x))$ to $\mathcal{UR}$.

Fig. 2: The ideal functionality F_{OPRF}. It interacts with $\mathcal{UR}$, $\mathcal{SR}$,and $\mathcal{SIM}$ via the EVAL query, SNDRCOMPLETE query, and RCVCOMPLETE query.

protocol that allows the server SR, on input the key k, to let the user UR compute the value $f_k(x)$ of a PRF $f_k(\cdot)$ on any input x of UR's choice without releasing any other information to SR to make sure that the SR learns nothing from the protocol. An OPRF protocol corresponding to a PRF function $f_k(\cdot)$ is a secure computation protocol for functionality F_{OPRF}: $(k, x) \rightarrow (f_k(x), \perp)$.

Definition 1. A two-party protocol π between a user UR and a server UR is an oblivious pseudorandom functionality (OPRF) if there exists some PRF family $f_k(\cdot)$ that π securely realizes the following functionality:

1) UR inputs x; SR inputs key k.
2) UR outputs $f_k(x)$; SR outputs nothing.

The ideal functionality of the oblivious pseudorandom functionality (OPRF) is shown in Fig. 2.

The ideal functionality F_{OPRF} involves the user $\mathcal{UR}$, the sever $\mathcal{SR}$, and an ideal-world simulator $\mathcal{SIM}$. We denote by λ the security parameter, which is the length of the output of F_{OPRF}. As shown in Fig. 2, the evaluation is triggered by an (EVAL, sid, $\mathcal{SR}$, x) query from $\mathcal{UR}$ requesting the computation of F_{OPRF} with $\mathcal{SR}$ on input x. SNDRCOMPLETE and RCVCOMPLETE denote the completion of $\mathcal{SR}$'s and $\mathcal{UR}$'s computation during the session identified by sid. An (EVAL, sid, $\mathcal{SR}$, x) query from $\mathcal{UR}$ with $\mathcal{SR}$ is completed by a (RCVCOMPLETE, sid, $\mathcal{UR}$, $\mathcal{SR}$, $\mathcal{SR}^*$) where $\mathcal{SR}^*$ is specified by adversary $\mathcal{A}$ and may not be the server $\mathcal{SR}$ in the EVAL query. In the case $\mathcal{SR} = \mathcal{SR}^*$, we have a computation with the intended server $\mathcal{SR}$. While in the other case $\mathcal{SR} \neq \mathcal{SR}^*$, it corresponds to the attacker channeling the request to a different server, possibly a corrupted one. We allow the adversary to specify a value $\mathcal{SR}^*$ which may not even be a server identity (in such case $\mathcal{SR}^*$ will be interpreted as a pointer to a function table). Thus, there is no guarantee of the correctness of the evaluation request and two requests with the same $\mathcal{SR}$ and x can be answered differently.

The functionality F_{OPRF} maintains a table T used to record the results of the OPRF evaluation by different senders on user-requested inputs x. Specifically, $T(\mathcal{SR}, x)$ is defined as $\mathcal{SR}$'s OPRF evaluated on input x. The table's entries are initially undefined and are filled with random values by F_{OPRF} upon RCVCOMPLETE activations. Note that $T(\mathcal{SR}, x)$ is chosen at random even in the case that $\mathcal{SR}$ is corrupted or it corresponds to a function table of the adversary. As a result, any realization of F_{OPRF} needs to ensure that evaluations with corrupted servers result in pseudorandom outputs. Note that the values $T(\mathcal{SR}, x)$ are communicated to the requesting user, but the inputs x remain fully hidden as they are never communicated to any party, including $\mathcal{SIM}$. Finally, note that F_{OPRF} provides the adversary with direct access to all function tables by allowing EVAL queries directly to F_{OPRF}.

A highly efficient and composable Oblivious Pseudorandom Function (OPRF) protocol is proposed by Jarecki et al. [51], which requires only a single exponentiation operation for the server and two exponentiation operations for the user. This OPRF protocol securely realizes the ideal functionality F_{OPRF} shown in Fig. 2, and thus can be served as a component for constructing the MFA protocol in this paper.

D. Authenticated encryption (AE) [52], [53]

Authenticated encryption (AE) has been a vital operation in cryptography due to its ability to provide confidentiality, integrity, and authenticity at the same time. Traditional encryption-only schemes ensure confidentiality and integrity/authenticity as separate services but were subsequently shown to fail to protect even confidentiality without ensuring integrity. This fact paved the way for the naissance of the notion of authenticated encryption (AE). Encryption ensures the confidentiality of messages under a secret key. The sender encrypts a confidential plaintext message and transmits the ciphertext; the receiver decrypts it, returning the plaintext. The sender calculates a MAC and attaches it to the message for authentication. The receiver employs as the sender does to make sure that the two codes match; if they do match, then he/she is assured that the message is authentic and accepts it; otherwise, a forgery is assumed, and the message is discarded.

An AE scheme has the following steps:

(1) **Key Generation**. $KeyGen(\lambda) \rightarrow (K_e, K_m)$: Given the security parameter λ, it outputs secret key K_e and K_m.

Functionality F_{AE}

The functionality F_{AE} is parameterized by a security parameter κ. It interacts with the sender $\mathcal{S}$, the receiver $\mathcal{R}$, and the simulator $\mathcal{SIM}$ via the following queries.

Key Generation:

- Upon receiving the query $(\text{KEYGEN}, sid, \mathcal{S})$ from R, if $sid \neq (\mathcal{S}/\mathcal{R}, sid')$ or a record $(\text{keyrec}, sid, \mathcal{S}, \mathcal{R}, \cdot, \cdot)$ exists, ignore. Otherwise, send $(\text{KEYGEN}, sid, \mathcal{S}, \mathcal{R})$ to $\mathcal{SIM}$ and wait for a response from $\mathcal{SIM}$:
  - ▷ If receives OK from $\mathcal{SIM}$, create a record $(\text{keyrec}, sid, \mathcal{S}, \mathcal{R}, K_e, K_m)$, send $(\text{KEYCOMPLETE}, sid, \mathcal{S}, \mathcal{R}, K_e, K_m)$ to $\mathcal{S}$ and $\mathcal{R}$, and send $(\text{KEYCOMLETE}, sid, \mathcal{S}, \mathcal{R}, |K_e|, |K_m|)$ to $\mathcal{SIM}$.
  - ▷ If receives $REJECT$ from $\mathcal{SIM}$, return $FAIL$ to $\mathcal{S}$ and $\mathcal{R}$.

Encryption and Authentication:

- Upon receiving the query $(\text{AUTHENC}, sid, \mathcal{R}, M)$ from $\mathcal{S}$ (or from $\mathcal{SIM}$), record $(\text{authenrec}, sid, \mathcal{S}, \mathcal{R}, M, \cdot, \cdot)$, send $(\text{AUTHENC}, sid, |M|)$ to $\mathcal{SIM}$. If there exists no record $(\text{keyrec}, sid, \mathcal{S}, \mathcal{R}, K_e, K_m)$, ignore; Else, if there exists a record $(\text{keyrec}, sid, \mathcal{S}, \mathcal{R}, K_e, K_m)$ and $\mathcal{S}/\mathcal{R}$ is honest, do the following:
  - ▷ If there is no record $(\text{authenrec}, sid, \mathcal{S}, \mathcal{R}, K_e', K_m', M, CT, \sigma)$, generate CT and σ, record $(\text{authenrec}, sid, \mathcal{S}, \mathcal{R}, K_e', K_m', M, CT, \sigma)$, send $(\text{AUTHENCOMPLETE}, sid, \mathcal{S}, CT, \sigma)$ to $\mathcal{R}$ (or $\mathcal{SIM}$).
  - ▷ Else, if there exists a record $(\text{authenrec}, sid, \mathcal{S}, \mathcal{R}, M, CT, \sigma)$ send $(\text{AUTHENCOMPLETE}, sid, \mathcal{S}, CT, \sigma)$ to $\mathcal{R}$ (or $\mathcal{SIM}$). // message M has been queried via encryption and authentication interface

Decryption and Verification:

- Upon receiving the query $(\text{DECVRY}, sid, CT, \sigma)$ from $\mathcal{R}$ (or from $\mathcal{SIM}$), if $sid \neq (\mathcal{R}, sid')$ or there exists no record $(\text{keyrec}, sid, \mathcal{S}, \mathcal{R}, K_e, K_m)$, then output $(\text{DECVRYCOMPLETE}, sid, \cdot, reject)$.to $\mathcal{R}$ (or to $\mathcal{SIM}$) Otherwise, if there exists a record $(\text{authenrec}, sid, M, CT, \sigma)$, output $(\text{DECVRYCOMPLETE}, sid, M, accept)$ to $\mathcal{R}$ or to $\mathcal{SIM}$. If there exists no such record, randomly choose message and authentication tag $M' \in \mathbf{M}$ and $\sigma' \in \boldsymbol{\sigma}$, record $(\text{authenrec}, sid, \mathcal{S}, \mathcal{R}, M', CT, \sigma')$, and output $(\text{DECVRYCOMPLETE}, sid, M', accept)$ to $\mathcal{R}$ (or to $\mathcal{SIM}$).

Fig. 3: The ideal functionality F_{OPRF}. It interacts with $\mathcal{S}$, $\mathcal{R}$,and $\mathcal{SIM}$ via the KEYGEN query, AUTHENC query, and DECVRY query.

(2) **Encrypt and Authenticate**. $aEnc(M, K_e, K_m) \rightarrow (CT, \sigma)$: Given a variable-length message M and secret key K_e and K_m, it outputs the ciphertext CT and the corresponding fixed-length authentication tag σ.

(3) **Decrypt and Verification**. $aDec(CT, \sigma, K_e, K_m) \rightarrow (M, (accept/reject))$ Given the ciphertext CT and secret key K_e, K_m , it outputs the message M, and $accept$ if the decryption is successful and CT is valid; Otherwise, it outputs $reject$.

The ideal functionality of the authenticated encryption scheme is shown in Fig. 3. It involves three parties: the sender $\mathcal{S}$, the receiver $\mathcal{R}$ and the simulator $\mathcal{SIM}$. These parties interact with the ideal functionality F_{AE} as follows. Upon receiving (KEYGEN, sid, $\mathcal{R}/\mathcal{S}$) from party $\mathcal{S}$ or $\mathcal{R}$, F_{AE} checks if this session sid has been activated. If it has been activated and there exists a record (keyrec, sid, $\mathcal{S}$, $\mathcal{R}$, $\cdot$, $\cdot$), F_{AE} will also ignore this query. If this session has not been queried before, then F_{AE} sends (KEYGEN, sid, $\mathcal{S}$, $\mathcal{R}$) to $\mathcal{SIM}$ and waits for a response. If $\mathcal{SIM}$ returns OK, F_{AE} will choose K_e and K_m from the key space randomly and create a record (keyrec, sid, $\mathcal{S}$, $\mathcal{R}$, K_e, K_m), it outputs (KEYCOMPLETE, sid, K_e, K_m) to $\mathcal{S}$ and $\mathcal{R}$, and sends (KEYGEN, sid, $|K_e|$, $|K_m|$) to $\mathcal{SIM}$. If $\mathcal{SIM}$ returns $REJECT$, then output $FAIL$ to $\mathcal{S}$ and $\mathcal{R}$. It indicates that $\mathcal{SIM}$ can learn something about the established session key such as the length of the session key.

When receiving (AUTHENC, sid, $\mathcal{R}$ M) from S (or from $\mathcal{SIM}$), record (authenrec, sid, $\mathcal{S}$, $\mathcal{R}$, M) and send (AUTHENC, sid, $|M|$) to $\mathcal{SIM}$. Note that if $\mathcal{S}$ is honest, the simulator can learn the length of the message M. If there exists a record (keyrec, sid, $\mathcal{S}$, $\mathcal{R}$, K_e, K_m) and $\mathcal{S}$ is honest, F_{AE} will check whether there exists a record (authenrec, sid, $\mathcal{S}$, $\mathcal{R}$, M, CT, σ). If there exists such a record, it denotes that the message M has been queried on the session sid, therfore F_{AE} will send (AUTHENCOMPLETE, sid, $\mathcal{S}$, CT, σ) to $\mathcal{R}$ (or to $\mathcal{SIM}$). Otherwise, F_{AE} generates CT and σ from the ciphertext and credential space randomly, records (authenrec, sid, $\mathcal{S}$, $\mathcal{R}$, M, CT, σ) and sends (AUTHENCOMPLETE, sid, $\mathcal{S}$, CT, σ) to $\mathcal{R}$ (or to $\mathcal{SIM}$).

Upon receiving (DECVRY, sid, CT, σ) from $\mathcal{R}$ (or from $\mathcal{SIM}$), if this session sid is not the matched session or there is no record (keyrec, sid, $\mathcal{S}$, $\mathcal{R}$, K_e, K_m), F_{AE} will output (DECVRYCOMPLETE, sid, $\cdot$, $reject$) to $\mathcal{R}$. Otherwise, if there exists a record (encrec, sid, $\mathcal{S}$, $\mathcal{R}$, M, CT, σ) for (CT, σ), F_{AE} will output (DECVRYCOMPLETE, sid, M, $accept$) to $\mathcal{R}$ or (DECVRYCOMPLETE, sid, $\cdot$, $accept$) to SIM. If there exists no such a record (encrec, sid, $\mathcal{S}$, $\mathcal{R}$, M, CT, σ) for (CT, σ), which denotes that $\mathcal{S}$ or $\mathcal{SIM}$ makes no AUTHECC query to obtain CT and σ, F_{AE} then randomly chooses $M^{'}$ and $\sigma^{'}$, record (authenrec, sid, $\mathcal{S}$, $\mathcal{R}$, $M^{'}$, CT, σ), and outputs (DECVRYCOMPLETE, sid, M, $accept$) to $\mathcal{R}$ or (DECVRYCOMPLETE, sid, $\cdot$, $accept$) to $\mathcal{SIM}$.

A UC secure authenticated encryption scheme is presented by Canetti et al. [53], which is realized using a standard encrypt-then-mac approach combined with a programmable random oracle to achieve adaptive security. According to the security assumption defined in this paper, where the hash functions are treated as random oracles and the ideal

functionality F_{AE} denotes the ideal execution of the authenticated encryption scheme, thus the UC secure authenticated encryption scheme designed in reference [53] can be used to construct the MFA protocol in this paper.

E. Adversary model

Referring to representative works [3], [10], [54], the adversary model is described as follows. Additionally, we have added the adversary model **Adv7** to show the secret question guessing attacks conducted by the adversary when he compromises the smart card and attempts to recover the password.

- Adv1. An adversary can access and enumerate all elements in the attribute and password space efficiently.
- Adv2. An adversary can control communication channels, enabling interception, modification, eavesdropping, and replay of messages transmitted publicly.
- Adv3. An adversary can compromise any two of the three authentication factors (not all of them).
- Adv4. An adversary can compromise the servers, obtain secret values, and try to conduct impersonation attacks.
- Adv5. An adversary can compromise the random value and attempt to reconstruct the session key.
- Adv6. An adversary can register and log in as a legitimate user, aiming to access other secret information.
- Adv7. An adversary can compromise the smart card, perform secret question guessing attacks, and attempt to recover the password.

F. Evaluation criteria

The evaluation criteria outline the security requirements and ideal properties that a well-designed multi-factor authentication scheme should meet. As mentioned earlier, Wang-Wang's evaluation criteria [10] is the most comprehensive and concrete, and it has been widely adopted (see [7], [10], [49], [54]). Here we make some modifications (i.e., dynamic password recovery, no passwords and secret answers exposure, and fine-grained access control) and present as follows.

- C1. **No Password Verifier Table**: The authentication gateway and sensor nodes don't generate password tables.
- C2. **Dynamic Password Recovery and Updating**: It permits users to recover and update passwords dynamically and flexibly.
- C3. **No Password, Attribute and Secret Answers Exposure**: Publicly transmitted messages ensure no password, attribute and secret answers exposure.
- C4. **Resistance to Smart Card Loss Attack**: It is infeasible for adversaries to conduct other known attacks through a stolen smart card.
- C5. **Resistance to Known Attacks**: The scheme maintains security against various attacks, such as password guessing attacks, and node capture attacks.
- C6. **Sound Flexibility**: The scheme allows for the revocation of accounts to exit the session.
- C7. **Key-Agreement Provision**: A session key is collaboratively established between users and servers.
- C8. **No Clock Synchronization**: The scheme operates effectively without synchronized clocks.
- C9. **Timely Detection**: It is configured to respond promptly to malicious behaviour.
- C10. **Mutual Authentication**: Mutual authentication is conducted among all participating entities to ensure communication security.
- C11. **User Anonymity and Untraceability**: It is designed to protect the privacy of all entities, ensuring anonymity and untraceability.
- C12. **Forward Security**: When the long-term secret is compromised, the session key remains secure.
- C13. **Fine-Grained Access Control**: It sets an access policy to achieve access control via users' attributes.

Note that **C2** is enhanced to support dynamic password recovery to prevent account loss. Moreover, **C3** has been modified to ensure that publicly transmitted messages will not reveal any valid information about the answers to secret questions, thus ensuring the security of the password recovery. In addition, **C13** has been added to check whether MFA schemes can provide fine-grained access control.

G. UC Security Definition

The security of a scheme is defined through the computational indistinguishability of a real-world experiment and an ideal-world experiment, which is introduced by the UC security framework [55]–[57]. In the real-world experiment, an adversary interacts with the actual protocol execution, while in the ideal-world experiment, a simulator interacts with an ideal functionality that captures the desired security properties. The environment cannot distinguish between these two experiments with non-negligible probability, ensuring that the protocol achieves composable security.

The following entities are involved in the UC framework: dummy parties, a real-world adversary $\mathcal{A}$, an environment $\mathcal{E}$, an ideal-world functionality F, and an ideal-world simulator $\mathcal{SIM}$. $\mathcal{E}$ gives inputs to and obtains outputs from parties in both real-world and ideal-world experiments. More importantly, $\mathcal{E}$ can arbitrarily interact with $\mathcal{A}$ in the real-world experiment and $\mathcal{SIM}$ in the ideal-world experiment.

In the real-world experiment, there exists an extra adversary $\mathcal{A}$ who controls some corrupted parties and interacts with other parties who run the instance of the scheme π. In the ideal-world experiment, the same group of parties completes the task of scheme π by interacting with $\mathcal{F}$, which is trusted and cannot be compromised. In particular, there exists a simulator $\mathcal{SIM}$ that controls the same set of corrupted parties as $\mathcal{A}$ does in the ideal world. The communication between parties and $\mathcal{F}$ is authenticated and confidential. The key requirement for UC security is that no environment $\mathcal{Z}$ can distinguish whether it is interacting with the real-world experiment or the ideal-world one, meaning $\mathcal{SIM}$ must produce a perfect or computationally close imitation of the real-world adversary's effect.

Definition 1. Given a security parameter κ, a scheme π securely realize a functionality F if for any real-world adversary $\mathcal{A}$, there exists an ideal-world simulator (adversary) $\mathcal{SIM}$ such that $\Pr[\text{Real}_{\mathcal{A},\pi,\mathcal{E}} - 1] \approx \Pr[\text{Ideal}_{\mathcal{SIM},F,\mathcal{E}} - 1]$. Here, $\text{Real}_{\mathcal{A},\pi,\mathcal{E}}$ denotes the output of $\mathcal{E}$ observing the real-world experiment, while $\text{Ideal}_{\mathcal{SIM},F,\mathcal{E}}$ denotes the output of $\mathcal{E}$ observing the ideal-world experiment.

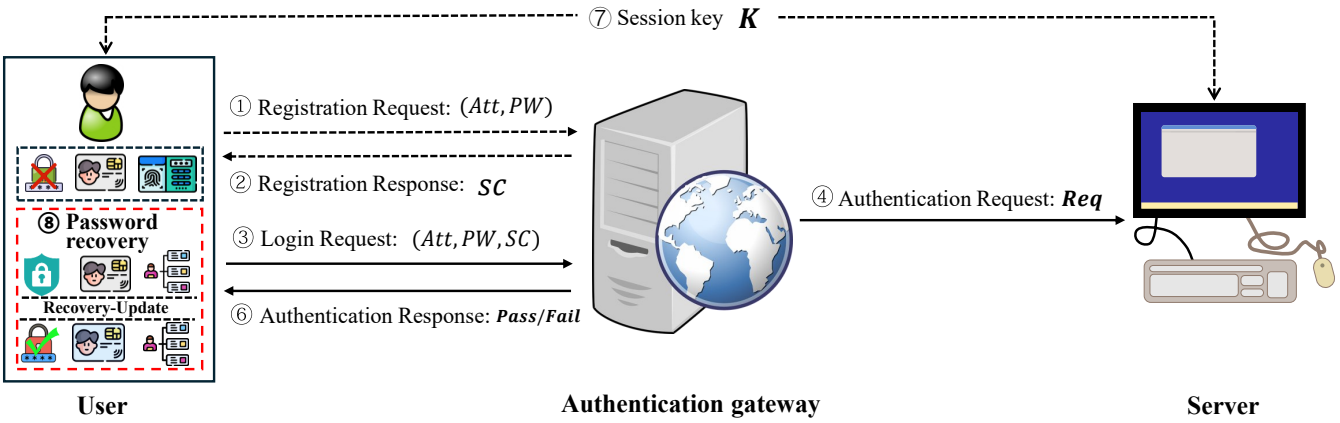

Fig. 4: The system framework of MFA-DPRF. It consists of four phases: registration phase, login and authentication phase, dynamic password recovery phase, and update phase. The user can recover the original password and then reset a new one during the update phase.

III. The Proposed Protocol: MFA-DPRF

This section presents the construction of MFA-DPRF, which includes six phases: system initialization phase, user registration phase, server registration phase, login and authentication phase, password recovery phase, and updating phase. Table I shows symbols and notations in the proposed scheme.

TABLE I: Symbol and notation

Symbol	Notation	Symbol	Notation
UR	User	AGW	Authentication Gateway
SR	Server	PW	Password of UR
Att	Attributes of UR	SC	Smart card of UR
$OPRF$	Oblivous pseudorandom functionality	H_i	Hash funtion
AE	Authenticated encryption	r	Random value
x	Secret chosen by UR	idx_i	Index of x_i
y	Secret chosen by AGW	s	Secret of SR
p	A large prime	KGF	Key generation function
e	Pairing operation	MAC	Message authentication code
$\oplus$	XOR operation	$\|\|$	Concatenation operation
CT	Ciphertext	σ	Authentication tag
k_{gu}, k_{gs}	Secret key	K	Session key
Q	Secret questions	asw	Answers
M, ρ	Access structure	PWC	Ciphertext of PW

A. Design Idea

The framework of the proposed scheme is shown in Fig. 4. In the registration phase, UR inputs attributes Att and runs oblivious pseudorandom functionality $OPRF$ with AGW on inputs ω_i. Upon receiving sk_i from $OPRF$, UR chooses a random value x and generates its shares as x_i; UR further computes $X[idx_i] = AES.ENC(x_i, sk_i)$, where $idx_i = H_3(Hpw_{\phi(att_i)} \,||\, att_i)$, $HPW \rightarrow (Hpw_1 \,||\, Hpw_2 \,||\, ... \,||\, Hpw_n)$. Note that only when UR inputs the correct password and a set of attributes Att^* that meet the access policy can x be successfully reconstructed in the login and authentication phase. Upon successfully reconstructing x, UR computes $C_1^* = H_4(H_1(HPW^*) \,||\, H_1(x) \bmod n_p)$, $PID_u = C_2 \oplus C_1$, and $C_3^* = H_1(H_5(g^{xy}) \,||\, PID_u)$. If $C_3^* = C_3$, UR can extract the secret information from the smart card SC. UR computes (CT_1, σ_1) via the authenticated encryption scheme, and $C_4 = H_1(HPW^* \,||\, PID_u \,||\, H_5(g^{r_{ug}}))$ to AGW as the login request. AGW will pass C_5, CT_2, σ_2, $g^{r_{gs}}$, T_3 to SR if it verifies $C_4^* = C_4$. If $C_5^* = C_5$, SR will compute $K = KGF(g^{r_{us}}, g^{r_{su}}, (g^{r_{us}})^{r_{su}})$ and send σ_3, $g^{r_{su}}$, T_4 to UR. UR will establish the session key K with SR if $\sigma_3^* = \sigma_3$.

Note that the proposed protocol employs an authenticated encryption scheme AE to protect the publicly transmitted message $g^{r_{us}}$ as $(CT_1, \sigma_1) = F_{\text{AE}}.Enc(m_1 = g^{r_{us}}, k_{gu} = g^{r_{ug}y})$. When successfully authenticating UR and verifying the validity of the message, AGW runs an instance of the authenticated encryption scheme to protect $g^{r_{us}}$. When SR authenticates AGW and the validity of the message, it chooses a random value r_{su} and generates the session key $K = KGF(g^{r_{us}}, g^{r_{su}}, (g^{r_{us}})^{r_{su}})$. Upon receiving σ_3, $g^{r_{su}}$, UR computes $K = KGF(g^{r_{us}}, g^{r_{su}}, (g^{r_{su}})^{r_{us}}$ and establishes the correct session key K with SR if $\sigma_3^* = \sigma_3$.

To conduct password recovery, UR inputs attributes Att_2^* and chooses random values $r_i^{''} \in_R \mathbb{Z}_p$, computes $c_1^* = g^{r_i^{''}}$, $c_2^* = H_2(H_3(att_i^* || EML || RID) \mod n_a)^{r_i^{''}}$, and computes the index value $H_3(att_i^* || salt)\}_{i \in |Att^*|}$. If $e(c_1, c_2^*) = e(c_1^*, c_2)$, the attribute att_i^* is valid. If the number of the valid attributes is no less than t, UR can extract the recovery credentials $\{M, \rho, Q, \{t_i\}_{i \in Q}, PWC\}$ from the smart card SC. UR then answers the secret questions $Q_i \rightarrow asw_i$. If UR gets asw_i correctly, she can reconstruct $\eta = \sum_{i \in Q} l_i \times \lambda_i$ and recover $PW = Dec(PWC, \eta)$. In addition, UR can reset secret questions $Q_i \rightarrow Q_i^{new}$, secret value $s \rightarrow s^{new}$, and LSSS parameters $(M, \rho) \rightarrow (M^{new}, \rho^{new})$to regenerate password recovery credentials. Dynamically updating the secret information can resist leakage issues and further enhance the security of password recovery. The password recovery and password update methods presented in this paper enable users to retrieve and reset their passwords. The password update process requires users to first enter their current password before setting a new one. If users forget or lose their password, they can first recover the current password and then decide whether to update it.

B. System Initialization

Given a security parameter κ, it outputs parameters $\{G_1, G_T, e, p\}$, where G_1 and G_T are cyclic groups of order p. p is a large prime whose length is at least 2^κ. g is a random generator of G_1. Five hash functions are defined as follows: $H_1 : \{0,1\}^* \rightarrow \{0,1\}^*$, $H_2 : \{0,1\}^* \rightarrow G_1$, $H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_p$, $H_4 : \{0,1\}^* \rightarrow \{0,1\}^*$, $H_5 : G_1 \rightarrow \{0,1\}^*$. A key generation funciton KGF is chosen, KGF: $G_1 \times G_1 \times G_1 \rightarrow \{0,1\}^*$. A message authentication code is defined: MAC: $\{0,1\}^* \times \{0,1\}^* \times G_1 \times G_1 \rightarrow \{0,1\}^*$. Additionally, a

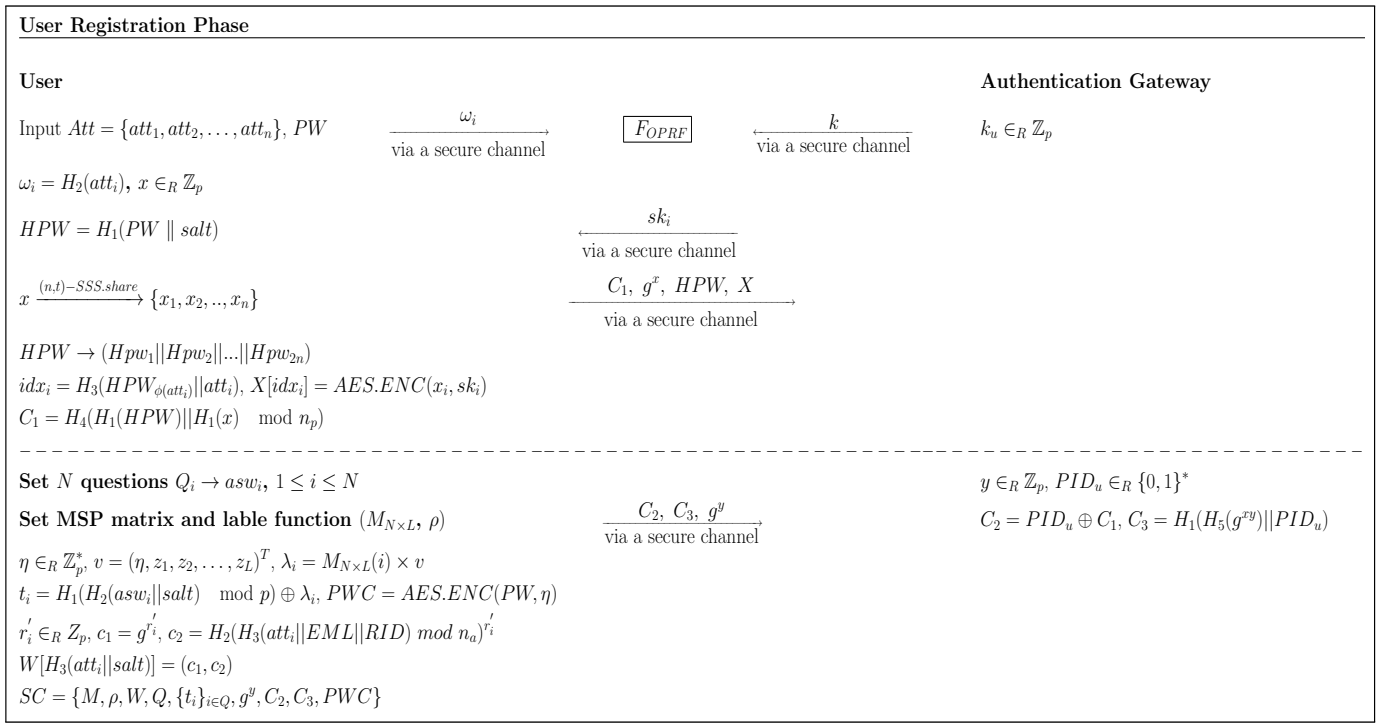

Fig. 5: User registration phase. UR sends registration requests to AGW and obtains partial smart card data as a registration response in this phase.

label function ϕ is chosen, $\phi : \{0,1\}^* \rightarrow \mathbb{Z}_{2n}$; n_p and n_a denotes the size of the password and attribute space. The public parameters are as follows params $= \{G_1, G_T, p, g, n_p, n_a, (H_i)_{1\le i\le 5}, \phi, KGF, MAC\}$.

C. User Registration

This phase is shown in Fig. 5 and described as follows.

UR1. UR first sets attributes $Att = \{att_1, att_2, \ldots, att_n\}$, and password PW, and computes the hash value $HPW = H_1(PW \parallel salt)$, and generates $2n$ parts of HPW as $HPW \rightarrow (Hpw_1||Hpw_2||...||Hpw_{2n})$; computes $\omega_i = H_2(att_i)$; runs $OPRF$ with input ω_i to obtain sk_i. Note that the OPRF protocol is treated as an ideal functionality F_{OPRF} in this paper which is executed securely between UR and AGW. Therefore, sk_i is transmitted securely to UR. Then, UR chooses $x \in_R \mathbb{Z}_p$ and generates shares x_i as $x \xrightarrow{(n,t)-SSS.share} \{x_1, x_2, \ldots, x_n\}$; computes $idx_i = H_3(Hpw_{\phi(att_i)}||att_i)$, $X[idx_i] = AES.ENC(x_i, sk_i)$, $C_1 = H_4(H_1(HPW)||H_1(x) \mod n_p)$ and sends X, C_1 to AGW.

UR2. Note that,AGW chooses a random key $k_u \in_R \mathbb{Z}_p$ and runs $OPRF$ with UR. Upon receiving X and C_1, AGW chooses a random value $y \in_R \mathbb{Z}_p$, $PID_u \in_R \{0,1\}^*$, and computes $C_2 = PID_u \oplus C_1$, $C_3 = H_1(H_5(g^{xy})||PID_u)$. It keeps k_u, PID_u, X, and sends C_2, C_3 to UR.

UR3. Upon receiving C_2 and C_3, UR sets N number-driven secret questions (e.g., "Your last phone number") $Q_i \rightarrow asw_i$, $1 \le i \le N$ and a MSP matrix and lable function $(M_{N\times L}, \rho)$; chooses a random value $\eta \in_R \mathbb{Z}_p^*$ and sets a random vector $v = (\eta, z_1, z_2, \ldots, z_L)^T$, where $z_i \in \mathbb{Z}_p$; then computes $\lambda_i = M_{N\times L}(i) \times v$, $t_i = H_1(H_2(asw_i||salt) \mod p) \oplus \lambda_i$, $PWC = AES.ENC(PW, \eta)$; then UR chooses $r_i^{'} \in_R \mathbb{Z}_p$, computes $c_1 = g^{r_i^{'}}$, $c_2 = H_2(H_3(att_i||EML||RID) \mod n_a)^{r_i^{'}}$ where EML denotes the email address and RID denotes the recvoery identity associated with the specific user; then defines $W[H_3(att_i||salt)] = (c_1, c_2)$. It also sets a threshold num_{rec} which denotes the maximum number of password recovery attempts. Finally, UR sets $SC = \{M, \rho, W, Q, \{t_i\}_{i\in Q}, g^y, C_2, C_3, PWC, num_{rec}\}$.

D. Server Registration

SR1. Upon receiving the registration request from SR, AGW chooses $s \in_R \mathbb{Z}_p^*$, sends s to SR and keeps g^s.

SR2. Upon receiving s, SR keeps it securely.

E. User Login and Authentication

User login and authentication phase is shown in Fig. 6.

UL1. UR inputs Att_2^*, computes $\omega_i^* = H_2(att_i^*)$, then UR runs $OPRF$ with the input ω_i^* and obtains sk_i; UR inputs PW^*, computes $HPW^* = H_1(PW^* \parallel salt)$, $HPW^* \rightarrow (Hpw_1 \parallel Hpw_2 \parallel ... \parallel Hpw_{2n})$, $idx_i = H_3(Hpw_{\phi(att_i^*)} \parallel att_i^*)$, and sends idx_i, T_1 to AGW.

UL2. Upon receiving $\{idx_i\}_{i\in_R Att_2^*}$, if $|T_1^{'} - T_1| \le \Delta T$ and $X[idx_i] \neq \perp$, AGW sets $count_u += 1$. If $count_u \ge t$, it sends $\{X[idx_i]\}_{i\in Att_2^*}$ to UR.

UL3. Upon receiving $\{X[idx_i]\}_{i\in Att_2^*}$, UR performs the decryption to recover $x_i = AES.DEC(X[idx_i], sk_i)$ and recovers $SSS.Reconstruct(x_i) \rightarrow x$. UR inserts SC, computes $C_1^* = H_4(H_1(HPW^*) \parallel H_1(x) \mod n_p)$, $PID_u = C_2 \oplus C_1$, and $C_3^* = H_1(H_5(g^{xy})||PID_u)$. If $C_3^* \neq C_3$, abort; otherwise, chooses $r_{ug}, r_{us} \in_R \mathbb{Z}_p^*$, computes $k_{gu} = (g^y)^{r_{ug}}$, $(CT_1, \sigma_1) = F_{\text{AE}}.Enc(m_1 = g^{r_{us}} \parallel T_2, k_{gu})$, and $C_4 = H_1(HPW^* \parallel PID_u \parallel H_5(g^{r_{ug}}))$. UR sends C_4, CT_1, σ_1, $g^{r_{ug}}$, T_2 to AGW on the public channel.

UL4. Upon receiving C_4, CT_1, σ_1, T_2, if $|T_2^{'} - T_2| \le \Delta T$, AGW computes $k_{gu} = (g^{r_{ug}})^y$, $m_1 = F_{\text{AE}}.Dec(CT_1, k_{gu})$; if

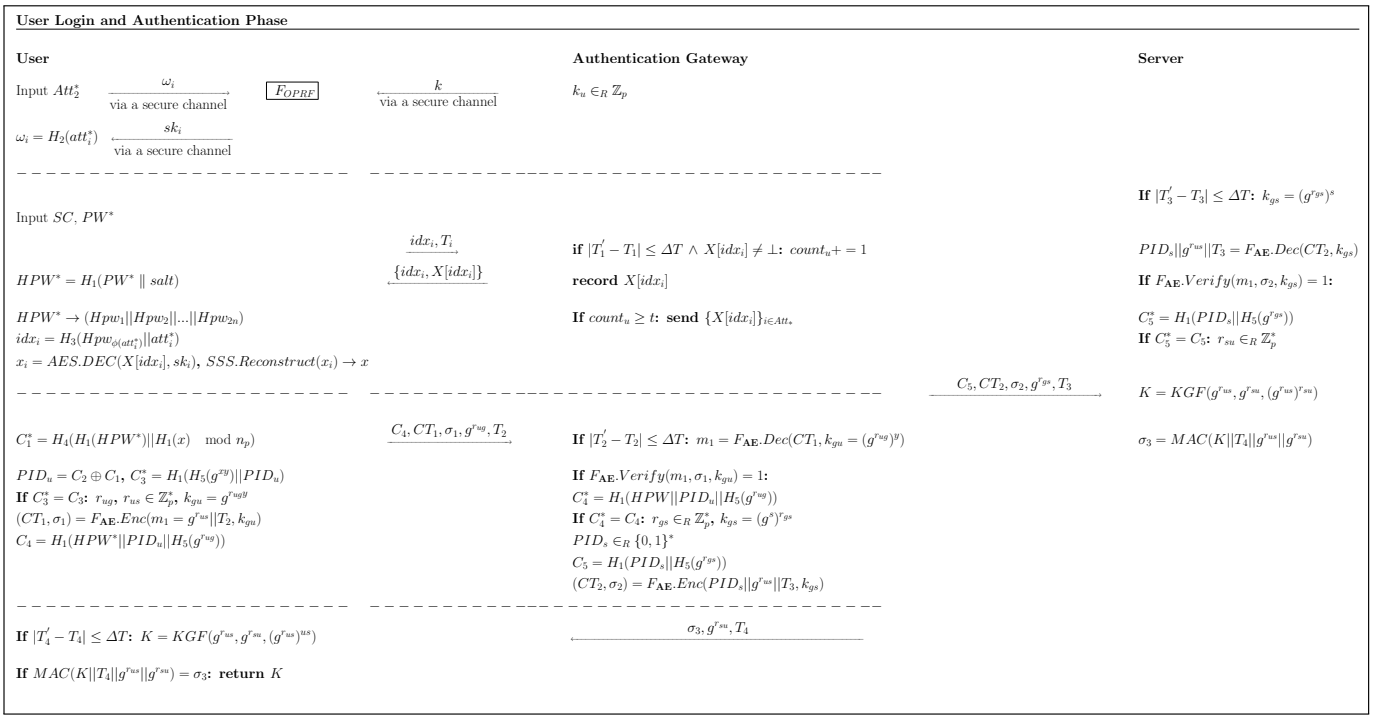

Fig. 6: User login and authentication phase. UR logs in to the system and establishes a session key with SR after mutual authentication in this phase.

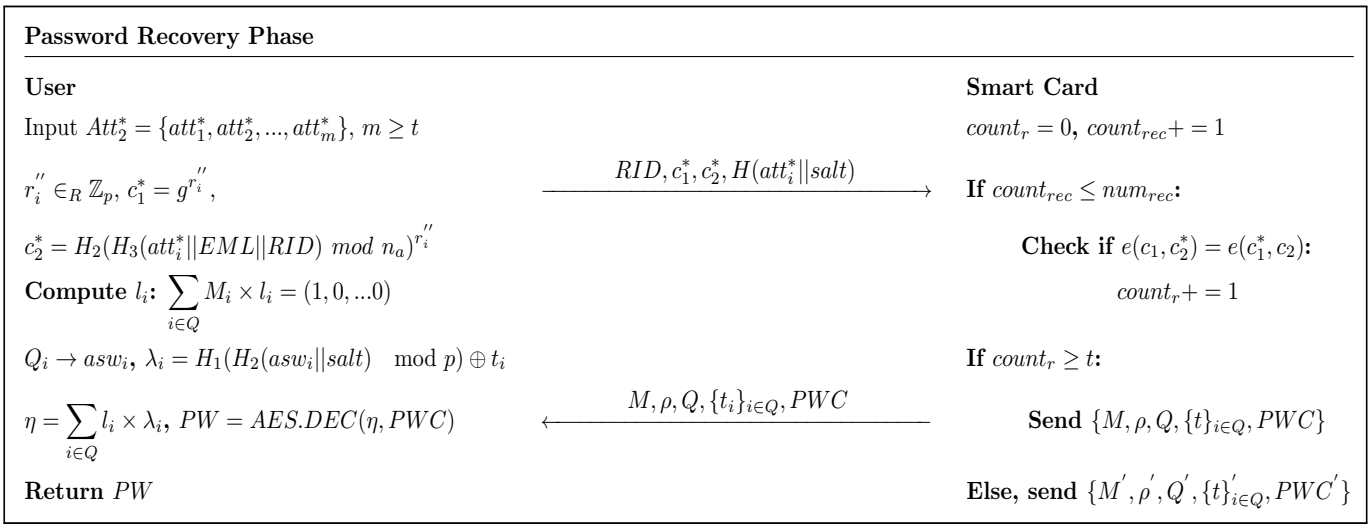

Fig. 7: Password recovery phase. UR can retrieve her previous passwords locally when she forgets or loses password in this phase.

$F_{\text{AE}}.Verify(m_1, \sigma_1, k_{gu}) = 1$, it computes $C_4^* = H_1(HPW \parallel PID_u \parallel H_5(g^{r_{ug}}))$; if $C_4^* = C_4$, it chooses $r_{gs} \in_R \mathbb{Z}_p^*$, $PID_s \in_R \{0,1\}^*$, computes $C_5 = H_1(PID_s \parallel H_5(g^{r_{gs}}))$, $k_{gs} = (g^s)^{r_{gs}}$ and $(CT_2, \sigma_2) = F_{\text{AE}}.Enc(PID_s \parallel g^{r_{us}} \parallel T_3, k_{gs})$. Finally, AGW sends C_5, CT_2, σ_2, $g^{r_{gs}}$, T_3 to SR.

UL5. Upon receiving C_5, CT_2, σ_2, $g^{r_{gs}}$, T_3, if $|T_3^{'} - T_3| \leq \Delta T$, SR computes $k_{gs} = (g^{r_{gs}})^s$ and $PID_s \parallel g^{r_{us}} \parallel T_3 = F_{\text{AE}}.Dec(CT_2, k_{gs})$; if $F_{\text{AE}}.Verify(PID_s \parallel g^{r_{us}} \parallel T_3, \sigma_2, k_{gs}) = 1$, it chooses $r_{su} \in_R \mathbb{Z}_p^*$, computes $K = KGF(g^{r_{us}}, g^{r_{su}}, (g^{r_{us}})^{r_{su}})$ and $\sigma_3 = MAC(K \parallel T_4 \parallel g^{r_{us}} \parallel g^{r_{su}})$. Finally, SR sends σ_3, $g^{r_{su}}$, T_4 to UR.

UL6. Upon receiving σ_3, $g^{r_{su}}$, T_4, if $|T_4^{'} - T_4| \leq \Delta T$, UR computes $K = KGF(g^{r_{us}}, g^{r_{su}}, g^{r_{us}r_{su}})$ and $\sigma_3^* = MAC(K \parallel T_4 \parallel g^{r_{us}} \parallel g^{r_{su}})$; if $\sigma_3^* = \sigma_3$, UR returns K.

F. Password Recovery

This phase is shown in Fig. 7 and described as follows.

PR1. UR inserts SC to the specific device and inputs attributes Att_2^*; UR chooses random values $r_i^{''} \in_R \mathbb{Z}_p$, computes $c_1^* = g^{r_i''}$, $c_2^* = H_2(H_3(att_i^* \parallel EML \parallel RID)\ mod\ n_a)^{r_i''}$, and sends $\{RID, c_1^*, c_2^*, H_3(att_i^*||salt)\}_{i \in Att_2^*}$ to the device.

PR2. Upon receiving $\{RID, c_1^*, c_2^*, H_3(att_i^*||salt)\}_{i \in Att_2^*}$, the device computes the current number of attempts at password recovery $count_{rec} += 1$ which corresponds to the recovery identity RID and checks if $count_{rec} \leq num_{rec}$; if so, it extracts $W[H_3(att_i^*||salt)] = (c_1, c_2)$, set $count_r = 0$, and checks if $e(c_1, c_2^*) = e(c_1^*, c_2)$; if the above equality holds, it sets $count_r += 1$; if $count_r \geq t$, it sends $\{M,$

ρ, Q, $\{t_i\}_{i\in Q}$, $PWC\}$ to UR, and sets $count_{rec}$ to zero; if $count_r < t$, dummy message $\{M^{'}, \rho^{'}, Q^{'}, \{t_i\}^{'}_{i\in Q}, PWC^{'}\}$ following the same distribution will be sent to UR.

PR3. Upon receiving $\{M, \rho, Q, \{t\}, PWC\}$ from SC, UR answers $Q_i \rightarrow asw_i$, and computes $\lambda_i = H_1(H_2(asw_i||salt) \bmod p) \oplus t_i$. Furthermore, UR computes l_i where $\sum_{i\in Q} l_i \times M_i = (1, 0, \ldots, 0)$, and $\eta = \sum_{i\in Q} l_i \times \lambda_i$. UR computes $PW = AES.DEC(PWC, \eta)$.

Specifically, UR is able to change secret questions $Q_i \rightarrow Q_i^{new}$, $(M, \rho) \rightarrow (M^{new}, \rho^{new})$ and chooses a new secret η^{new} and sets $v^{new} = (\eta^{new}, z_1^{new}, z_2^{new}, \ldots, z_L^{new})^T$ to compute $\lambda_i^{new} = M^{new}(i) \times v^{new}$, $t_i^{new} = H_1(asw_i^{new}||salt) \oplus \lambda_i^{new}$, $PWC^{new} = AES.ENC(PW, \eta^{new})$. Dynamically updating these data can prevent privacy leakage and further enhance the security of password recovery. That's why we call it dynamic password recovery.

Since this process is only executed when users want to perform password recovery, the computational overhead generated in this stage will not affect the user's login and authentication process every time; At the same time, as long as the attributes Att_2^* meets the threshold policy (n, t), the user can pass the authentication, so the number of pairing operations executed in the smart card can be compressed to $2t$ times, which reduces the computational cost. Additionally, if the smart card is kept safe, the adversary cannot perform attribute guessing attacks; if the smart card is stolen, due to the application of the verifier technique ($c_2 = H_2(H_3(att_i||EML||RID) \bmod n_a)^{r_i^{'}}$, $c_2^* = H_2(H_3(att_i^*||EML||RID) \bmod n_a)^{r_i^{''}}$), and the rate-limiting policy (allows the adversary to perform password recovery no more than num_{rec} times), the adversary fails to guess attributes or extracts the attributes from the compromised smart card efficiently.

G. Password, Attributes, and Smart Card Update

a. After providing the old password PW, attributes Att and smart card SC, UR recover x and PID_u according to the above description, computes $C_5^* = H_1(H_5(g^{xy})||PID_u)$ and sends C_5^*, T_4 to AGW.

b. Upon receiving C_5, if $|T_4^{'} - T_4| \leq \Delta T \wedge C_5^* = C_5 = H_1(H_5(g^{xy})||PID_u)$, AGW verifies the identity of UR.

c. UR selects the new password PW^{new}, attributes Att^{new}, computes $HPW^{new} = H_1(PW^{new} \parallel salt^{new})$; chooses $r_i^{new} \in_R Z_p^*$, and computes $f_i^{new} = H_1(att_i^{new})$, $\omega_i^{new} = H_2(f_i^{new})^{r_i^{new}}$; runs $OPRF$ with AGW to obtain sk_i^{new}; chooses $x^{new} \in_R \mathbb{Z}_p$ and generates shares x_i^{new} as $x \xrightarrow{(n,t)-SSS.share} \{x_1^{new}, x_2^{new}, \ldots, x_n^{new}\}$; computes $idx_i^{new} = H_3(att_i^{new}||HPW^{new})$, $X[idx_i^{new}] = AES.ENC(x_i^{new}, sk_i^{new})$, $C_1^{new} = H_4(H_1(HPW^{new})||H_1(x^{new}) \bmod n_p)$, and sends X^{new} and $g^{x^{new}}$ to AGW.

d. Upon receiving X^{new} and $g^{x^{new}}$, AGW updates $X \rightarrow X^{new}$ and $g^{xy} \rightarrow g^{x^{new}y}$.

e. UR chooses $(r_i^{'})^{new} \in_R Z_p$, computes $c_1^{new} = g^{(r_i^{'})^{new}}$, $c_2^{new} = H_2(H_3(att_i^{new} \parallel EML \parallel RID)^{(r_i^{'})^{new}}$ and defines $W[H_3(att_i^{new} \parallel salt^{new})] = (c_1^{new}, c_2^{new})$; computes $PWC^{new} = AES.ENC(PW^{new}, s)$, $C_2^{new} = PID_u \oplus H_4(H_1(HPW^{new}) \parallel H_1(x^{new}) \bmod n_p)$, $C_3^{new} = H_1(H_5(g^{x^{new}y}) \parallel PID_u)$; finally, UR updates the smart card as $SC^{new} = \{M, \rho, W^{new}, Q, \{t_i\}_{i\in Q}, g^y, C_2^{new}, C_3^{new}, PWC^{new}, num_{rec}\}$.

Considering the scenario where users prefer to use their previous password rather than resetting a new one, we design password recovery and password update separately. However, enabling a direct password reset within the MFA protocol remains a challenge, as authenticating with a newly chosen password may typically require a full registration process and some potential threats (e.g., password reuse) need to be analyzed. Therefore, designing a MFA protocol with password recovery and reset deserves further research and we will address this issue in future work.

IV. Formal Security Analysis of MFA-DPRF

Theorem 1. Assume that there is a probabilistic polynomial-time adversary $\mathcal{A}$ that executes q_E times of Extract(θ_{UR}, θ_{AGW}, θ_{SR}), q_S times of Send(θ_{UR}, θ_{AGW}, θ_{SR}, M), q_{H_i} times of $H_i(\cdot)_{i\in[1,5]}$, q_{enc} encryption and decryption queries and q_{mac} message authentication code queries. Then $\mathrm{Adv}_{\mathcal{A}}^{\text{MFA-DPR}} \leq \sum_{i\in[1,4]} \frac{q_{H_i}^2 + q_S}{2^{l_{H_i}}} + Adv_{\mathcal{A}}^{AE} + \frac{q_{enc}^2}{2^{l_{enc}+1}} + \frac{q_{mac}^2}{2^{l_{mac}+1}} + 2C^{'}q_S^{\eta^{'}} + \frac{(q_E+q_S)^2+2q_S}{2^p} + \frac{q_{H_1}N\lambda_{asw_i}}{G_{asw_i}} + \sum_{i\geq t, q_S^{'}\leq num_{rec}} \frac{q_S^{'}}{2^{l_{att_i}}} + q_{kgf}\varepsilon_{cdh}$. Referring to works [10], [41], [58] $C^{'}$, $\eta^{'}$, λ_{asw_i} and G_{asw_i} are constants related to the password and private questions, l_i is the length of $H_i(\cdot)_{i\in[1,5]}$, l_{enc} is the length of the ciphertext, l_{mac} denotes the length of MAC, l_{att_i} denotes the length of att_i, and ε_{cdh} denotes the advantage in solving the CDH problem.

The complete security proof is shown in Appendix A.

V. Analysis of Password Recovery Security

Theorem 2. Assuming the existence of an efficient adversary $\mathcal{A}$ who can extract Q, PWC and $\{t_i\}_{i\in Q}$ from SC, the advantage of $\mathcal{A}$ successfully recovering the password is no more than $Adv_{\mathcal{A}}^{AES} + \frac{q_{H_1}N\lambda_{asw_i}}{G_{asw_i}} + \sum_{i\geq t, q_S^{'}\leq num_{rec}} \frac{q_S^{'}}{2^{l_{att_i}}} + \frac{1}{2^{pN}}$, where $Adv_{\mathcal{A}}^{AES}$ denotes the advantage of $\mathcal{A}$ successfully breaking the underlying AES algorithm and $\frac{\lambda_{asw_i}}{G_{asw_i}}$ denotes the advantage of guessing the secret question. See a more detailed analysis of password recovery in Appendix B.

VI. UC Security Analysis of MFA-DPRF

Theorem 3. The proposed scheme securely realizes the F_{MFA} in $\{F_{\text{OPRF}}, F_{\text{AE}}, F_{\text{CRS}}\}$-hybrid model. In other words, given a security parameter κ, for any real-world adversary $\mathcal{A}$, there exists an ideal-world simulator $\mathcal{SIM}$ such that environment $\mathcal{E}$ is unable to distinguish whether it interacts with $\mathcal{A}$ and MFA-DPRF in the real world or with $\mathcal{SIM}$ and F_{MFA} in the ideal world,

$$Pr[\text{Real}_{\mathcal{A},\text{MFA-DPRF},\mathcal{E}}^{hybrid} = 1] \approx Pr[\text{Ideal}_{S,F_{\text{MFA}},\mathcal{E}} = 1].$$

Functionality F_{MFA}

The functionality F_{MFA} is parameterized by a security parameter κ. It sets a password recovery threshold num_{rec} and a value $count_{rec}$ to record the recovery attempts. F_{MFA} interacts with the user $\mathcal{UR}$, the server $\mathcal{SR}$, and the simulator $\mathcal{SIM}$ via the following queries.

Registration:

- Upon receiving the query $(\text{REGIS}, sid, \mathcal{AGW}, PW, Att, Q)$ from $\mathcal{UR}$, if it is the first registration query, record $(\text{regisrec}, sid, \mathcal{UR}, PW, Att, Q, \{asw_i\}_{i\in Q}, SC)$ and make it fresh. If this is the second registration query and there exists a record $(\text{regisrec}, \mathcal{UR}, PW', Att', \{asw_i'\}_{i\in Q}, SC')$, then record $(\text{regisrec}, sid, \mathcal{UR}, PW, Att, Q, \{asw_i\}_{i\in Q}, SC)$. Finally, send $(\text{REGIS}, sid, \mathcal{AGW}, \mathcal{UR}, |PW|, |Att|, |att_i|, |asw_i|)$ to $\mathcal{SIM}$ and wait for response from $\mathcal{SIM}$:
  - ▷ If receives OK from $\mathcal{SIM}$, return $(\text{regisrec}, sid, SC)$ to $\mathcal{UR}$;
  - ▷ If receives $REJECT$ from $\mathcal{SIM}$, return $FAIL$ to $\mathcal{UR}$.
- Upon receiving a query $(\text{REGIS}, sid, \mathcal{AGW}, ID_{SR})$ from $\mathcal{SR}$, if it is the first registration query, choose a random value $s \in_R Z_p^*$, record $(\text{regisrec}, sid, SR, s)$, send $(\text{NewSess}, sid, SR, |PW|, |Att|, |att_i|)$ to $\mathcal{SIM}$ and wait for response from $\mathcal{SIM}$:
  - ▷ If receives OK from $\mathcal{SIM}$, return $(\text{regisrec}, sid, \mathcal{SR}, s)$ to $\mathcal{SR}$;
  - ▷ If receives $REJECT$ from $\mathcal{SIM}$: return $FAIL$ to $\mathcal{SR}$.

Login and Authentication:

- Upon receiving the query $(\text{LOGAUTH}, sid, PW^*, Att^*, SC^*)$ from $\mathcal{UR}$ (or from $\mathcal{SIM}$), first keep the record $(\text{logauthrec}, sid, \mathcal{AGW}, \mathcal{UR}, W^*, Att^*, SC^*)$. Then, send $(\text{LOGAUTH}, sid, |PW^*|, |Att^*|, |att_i^*|)$ to $\mathcal{SIM}$ and wait for a response from $\mathcal{SIM}$:
  - ▷ If receives OK from $\mathcal{SIM}$, check if there exist a record $(\text{regisrec}, sid, \mathcal{UR}, PW, Att, Q, \{asw_i\}_{i\in Q}, SC)$:
    - -If there exists a record, and $PW = PW^* \wedge |Att \cap Att^*| \geq t$, return "successful login and authentication" to $\mathcal{UR}$.
    - -Else, return $FAIL$ to $\mathcal{UR}$.
  - ▷ If receives $REJECT$ from $\mathcal{SIM}$, return $FAIL$ to $\mathcal{UR}$.

Key Generation:

- Upon receiving the query $(\text{NEWKEY}, sid, \mathcal{UR}/\mathcal{SR}, K^*)$ from $\mathcal{SIM}$, if there is a record $(\text{keyrec}, sid, \mathcal{UR}, \mathcal{SR}, \cdot)$ not marked $COMPLETED$:
  - ▷ If this session sid is marked as $COMPROMISED$, or $\mathcal{UR}/\mathcal{SR}$ is corrupted, set $K = K^*$
  - ▷ If this session sid is marked as $FRESH$, and a session key K' has been sent to $\mathcal{UR}/\mathcal{SR}$, set $K = K'$.
  - ▷ Else, set a random session key $K \in_R \{0,1\}^{klen}$.

Password Recovery:

- Upon receiving the query $(\text{RECPW}, sid, Att^*, \{asw_i^*\}_{i\in Q}, SC^*)$ from $\mathcal{UR}$ (or from $\mathcal{SIM}$), set $count_{rec} += 1$ and keep the record $(\text{pwrec}, sid, \mathcal{UR}, Att^*, \{asw_i^*\}_{i\in Q}, SC^*)$ and then send $(\text{RECPW}, sid, |Att^*|, |att_i^*|, |asw_i^*|)$ to $\mathcal{SIM}$.
  - ▷ If there exists no record $(\text{regisrec}, sid, \mathcal{UR}, PW, Att, Q, \{asw_i\}_{i\in Q}, SC)$, ignore.
  - ▷ If $count_{rec} += 1 > num_{rec}$, return $\perp$ to $\mathcal{UR}$. // rate-limiting mechanism.
  - ▷ Else, if $|Att \cap Att^*| \geq t \wedge \{asw_i^*\}_{i\in Q} = \{asw_i\}_{i\in Q} \wedge count_{rec} += 1 \leq num_{rec}$, return PW to $\mathcal{UR}$. // attributes are valid and answers are correct
  - ▷ If $|Att \cap Att^*| < t \vee \{asw_i^*\}_{i\in Q} \neq \{asw_i\}_{i\in Q}$, return a random string $PW' \in_R \{0,1\}^{|PW|}$ to $\mathcal{UR}$. // incorrect answers or invalid attributes will lead to a random password

Active Attacks:

- Upon receiving the query $(\text{TEST}, sid, \mathcal{UR}, PW', Att', \{asw_i'\}_{i\in Q})$ from $\mathcal{SIM}$, if there exists no record $(\text{regisrec}, sid, \mathcal{UR}, PW, Att, Q, \{asw_i\}_{i\in Q}, SC)$, ignore; Else, do the following:
  - ▷ If $PW' = PW \wedge |Att \cap Att'| \geq t$, mark this session sid $COMPROMISED$ and return "correct guess" to $\mathcal{SIM}$. // password and attribute guessing attacks
  - ▷ Else, mark this session sid interrupted and return "wrong guess" to $\mathcal{SIM}$.
  - ▷ If $\{asw_i'\}_{i\in Q} = \{asw_i\}_{i\in Q}$, return "correct answers" to $\mathcal{SIM}$. // secret question guessing attacks
  - ▷ Else, return "wrong answers" to $\mathcal{SIM}$.
- Upon receiving the query $(\text{LEAKSC}, sid, \mathcal{UR})$ from $\mathcal{SIM}$, if there exists no record $(\text{regisrec}, sid, \mathcal{UR}, \cdot, \cdot, \cdot, \cdot, SC)$, ignore; Otherwise, return $leak(state, SC)$ to $\mathcal{SIM}$. // side channel attacks against the smart card

Fig. 8: The ideal functionality F_{MFA}. It interacts with $\mathcal{UR}$, $\mathcal{SR}$, and $\mathcal{SIM}$ via the REGIS query, LOGAUTH query, NEWKEY query, and RECPW query. Furthermore, it captures active attacks, including password guessing attacks, attribute guessing attacks, secret question guessing attacks, and side channel attacks, via the interface TEST and LEAKSC. Upon guessing the password and attributes successfully, the session sid is marked COMPROMISED.

A sequence of hybrid experiments is defined, starting from the real world and ending in the ideal world, where in each experiment we change some aspect of the simulation and show that $\mathcal{E}$ cannot distinguish any two consecutive experiments with non-negligible probability. Finally, we will prove that $\mathcal{E}$ is unable to distinguish the interaction with $\mathcal{A}$ and MFA-DPRF in the real world or with $\mathcal{SIM}$ and F_{MFA} in the ideal world, except with negligible probability.

Fig. 8 shows the ideal functionality F_{MFA}, which represents the desirable security properties of MFA-DPRF. The registration phase is captured by the query REGIS. The login and authentication phase is performed by the LOGAUTH query. The key generation phase is captured by the NEWKEY query and the active attacks such as password guessing attacks is performed by the TEST query.

Low-entropy secret information is always vulnerable to guessing attacks. Consequently, we capture password guessing attacks, attribute guessing attacks and secret questions guessing attacks via the interface $(\text{TEST}, sid, \mathcal{UR}, PW', Att', \{asw_i'\}_{i\in Q})$ in F_{MFA}. $\mathcal{SIM}$ can receive "correct guess" from

TABLE II: Security Comparison among Relevant Authentication schemes

Scheme	Year	Ref.	Authentication factor	Adverary model							Thirteen evaluation criteria												
				Adv1	Adv2	Adv3	Adv4	Adv5	Adv6	Adv7	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
Guo et al. et al.	2014	[21]	Attribute	✓	✓	-	×	×	✓	-	-	-	-	-	×	×	✓	✓	×	×	✓	×	✓
Sucasas et al.	2016	[32]	Attribute	✓	✓	-	×	×	✓	-	-	-	-	-	×	×	✓	✓	×	×	✓	×	✓
Zhang et al.	2018	[59]	Attribute	✓	✓	-	×	×	✓	-	-	-	-	-	×	×	✓	✓	×	×	✓	×	✓
Lu et al.	2019	[60]	Password+Smart card+Biometric	✓	✓	×	×	×	✓	×	✓	×	✓	×	×	✓	✓	×	✓	✓	×	×	×
Srinivas et al.	2021	[61]	Password+Smart card+Biometric	✓	✓	✓	✓	✓	✓	×	✓	×	✓	×	×	✓	✓	×	✓	×	×	×	×
Sahoo et al.	2021	[62]	Password+Smart card+Biometric	✓	✓	✓	✓	✓	✓	×	✓	×	✓	×	×	✓	✓	×	✓	✓	×	×	×
Han et al.	2021	[63]	Attribute	✓	✓	-	×	×	✓	-	-	-	-	-	×	×	✓	✓	×	×	✓	×	✓
Sutrala et al.	2022	[64]	Password+Smart card	✓	✓	✓	✓	✓	✓	×	✓	×	✓	×	×	✓	✓	×	✓	✓	×	✓	×
Braeken et al.	2023	[65]	Password+Smart card+Biometric	✓	✓	✓	✓	✓	✓	×	✓	×	✓	×	×	×	✓	×	✓	✓	×	×	×
Sucasas et al.	2023	[34]	Attribute	✓	✓	-	×	×	✓	-	-	-	-	-	×	×	✓	✓	×	×	✓	×	✓
Shi et al.	2024	[66]	Attribute	✓	✓	-	×	×	✓	-	-	-	-	-	×	×	✓	✓	×	×	✓	×	✓
Song et al.	2024	[35]	Attribute+Password	✓	✓	✓	✓	✓	✓	×	✓	×	✓	-	×	✓	✓	✓	✓	✓	✓	✓	✓
Wang et al.	2025	[67]	Identity+Biometric+Certificate	✓	✓	✓	✓	✓	✓	-	✓	-	-	-	✓	✓	✓	✓	×	✓	✓	×	×
Yang et al.	2025	[68]	Reputation+Certificate	✓	✓	-	✓	×	×	-	✓	-	-	-	✓	✓	×	✓	×	✓	✓	✓	×
Le et al.	2025	[69]	Password+Biometric+Smart card	✓	✓	✓	✓	×	✓	×	✓	×	✓	✓	×	✓	✓	✓	×	✓	✓	×	×
Han et al.	2025	[70]	Identity+Certificate	✓	✓	-	×	×	✓	-	-	-	-	-	×	×	✓	✓	×	×	✓	×	✓
Our scheme	2025	–	Password+Smart card+Attribute	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

†The details of the adversary model Adv1∼Adv6 and criteria C1∼C13 are referred to Section II-E and Section II-F. Note that "✓" means achieving the corresponding goal, while "×" means failing to achieving the corresponding goal, "−" means not applicable.

TABLE III: Performance Comparison among Relevant Authentication schemes

Scheme	Ref.	Computation cost			Communication cost		
		User	Server	Total	User	Server	Total
Guo et al.	[21]	$(27+2n)T_M+(45+n)T_E$	$(5+n)T_M+6T_P$	≈ 429.074 ms	5120 bits	11264bits	16384 bits
Sucasas et al.	[32]	$3T_H+(8+2n)T_M+2T_E+T_P$	$4T_H+(3+2n)T_M+2T_E+5T_P$	≈ 275.171 ms	5120 bits	12288 bits	17408 bits
Zhang et al.	[59]	$(3+3n)T_H+(16+n)T_M+6T_P$	$(1+n)T_H+(10+n)T_M$	≈ 357.477 ms	9728 bits	34816 bits	44544 bits
Lu et al.	[60]	$6T_H+T_S+3T_P+T_B$	$2T_H+2T_S+2T_P$	≈ 56.829 ms	2036 bits	852 bits	2888 bits
Srinivas et al.	[61]	$17T_H+6T_P+T_B$	$8T_H+4T_P$	≈ 107.491 ms	2472 bits	1504 bits	3976 bits
Sahoo et al.	[62]	$8T_H+2T_S+3T_P+T_B$	$4T_H+2T_S+2T_P$	≈ 65.602 ms	2576 bits	1640 bits	4216 bits
Han et al.	[63]	$(1+3n)T_H+(17+8n)T_M+(28+15n)T_E+(10+8n)T_P$	$21T_H+13T_M+28T_E+8T_P$	≈ 2491.283 ms	7680 bits	1176 bits	19456 bits
Sutrala et al.	[64]	$16T_H+T_B+5T_P$	$8T_H+4T_P$	≈ 100.093 ms	2608 bits	992 bits	3600 bits
Braeken et al.	[65]	$3T_H+4T_P+3T_S+T_B$	$4T_H+3T_P+T_S$	≈ 68.659 ms	1012 bits	768 bits	1780 bits
Sucasas et al.	[34]	$27T_H+(8+7n)T_M+(7+8n)T_E+3nT_P$	$2T_H+(9+4n)T_M+(6+5n)T_E+3nT_P$	≈ 2144.513 ms	3,018 bits	4,408 bits	7,426 bits
Shi et al.	[66]	$nT_H+(5n+2)T_M+(4n+13)T_E$	$nT_H+(2n+2)T_M+(n+1)T_E+7T_P$	≈ 915.683 ms	7168 bits	4,408 bits	11576 bits
Song et al.	[35]	$8T_H+(11+2n)T_M+(12+3n)T_E+5T_P$	$(1+4n)T_M+(3+8n)T_E$	≈ 1288.794ms	11776 bits	11776 bits	23552 bits
Wang et al.	[67]	$2T_H+T_B+(l_s+1)T_M+l_sT_F+(2l_s+3)T_E$	$T_H+(l_s+1)T_E$	≈ 167.972 ms	2048 bits	512 bits	2560 bits
Yang et al.	[68]	$3T_H+4T_M+7T_E$	$2T_H+4T_M+6T_E$	≈ 74.036 ms	4608 bits	2560 bits	7168 bits
Le et al.	[69]	$15T_H+T_B+6T_M$	$10T_H+(T+6)T_M$	≈ 89.631 ms	2560 bits	4352 bits	6912 bits
Han et al.	[70]	$(4l_s+12)T_M+(l_s+17)T_E+(2l_s+9)T_P$	$(2l_s+1)T_H+3l_sT_M+(4l_s+4)T_E+T_P$	≈ 1133.513 ms	3,018 bits	4,408 bits	7,426 bits
Our scheme	–	$(2n+4)T_H+(n+3)T_E+T_A+nT_F$	$3T_H+2T_E+T_S$	≈ 323.548 ms	11264 bits	1024 bits	12288 bits

† "n" represents the number of attributes (we set $n=20$); "l_s" represents the number of servers (we set $l_s=5$). T_H, T_S, T_E, T_M, T_B, T_P, T_A, T_F denote the hash operation, symmetric encryption operation, modular exponential operation, multiplication operation, fuzzy extraction operation, bilinear pairing operation, authentication encryption, and oblivious pseudorandom operation, respectively.

F_{MFA} when it successfully guesses the password and attributes. This session will be marked as $COMPROMISED$. Otherwise, $\mathcal{SIM}$ will receive "wrong guess", and this session will be marked as interrupted. The complete analysis under the UC framework can be seen in Appendix C.

VII. Informal Security Analysis of MFA-DPRF

This section presents an informal security analysis of MFA-DPRF, showing that it can resist known attacks such as password guessing attacks, impersonation attacks, node capture attacks, and achieves fine-grained access control, multi-factor security, dynamic password recovery, and forward security.

Resist password guessing attacks. MFA-DPRF employs the fuzzy verifier technique to mix up the distribution of passwords, effectively reducing the risk of password guessing attacks. If the adversary $\mathcal{A}$ intercepts the publicly transmitted message idx_i, C_4, CT_1 and σ_1, since we apply the fuzzy verifier technique to compute $idx_i = H_3(H_1(Hpw_i)||H_1(att_i) \bmod n_p)$ where $HPW \rightarrow (Hpw_1||Hpw_2||...||Hpw_n)$, it is difficult for $\mathcal{A}$ to guess the password PW even if he may provide a valid attribute. Due to the randomness of $g^{r_{ug}}$, r_{gs} and the security of the authenticated encryption scheme AE, it is impossible for $\mathcal{A}$ to extract some valid information about PW from the publicly transmitted message C_4, CT_1 and σ_1.

See the complete informal analysis in Appendix D.

VIII. Performance Analysis

This section conducts an exhaustive comparison of some representative authentication schemes with MFA-DPRF.

Table IV shows that these representative works [21], [32], [34], [35], [59], [63], [66] are attribute-based authentication schemes, while others are multi-factor authentication schemes. None of them employ attributes to construct MFA schemes except Song et al. [35]'s work. However, Song et al. [35] fails to address password forgetting and loss issues and cannot provide composable security. It reveals that, except for MFA-DPRF, other schemes fail to meet evaluation criterion C2, which requires dynamic password recovery and allows users to change their secret values flexibly. Additionally, nearly 80% of these schemes are unable to meet criteria C4 and C5, indicating they are vulnerable to password guessing attacks, node capture attacks, and other known attacks. Furthermore, although schemes [21], [32], [34], [35], [59], [63], [66] support fine-grained access control (see C13), none of these works

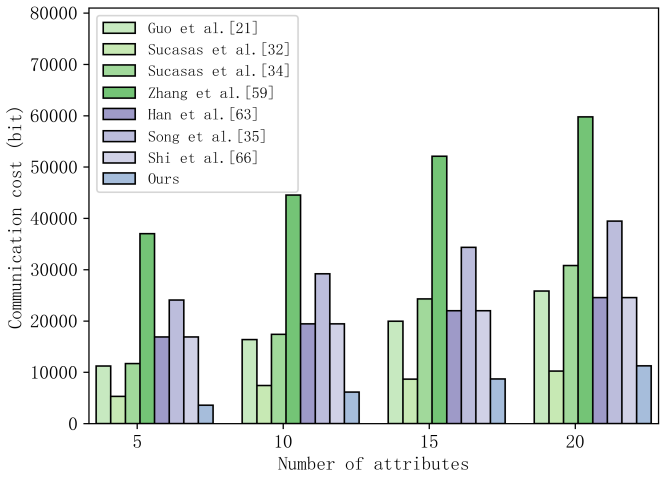

Fig. 9: Comparison of communication cost.

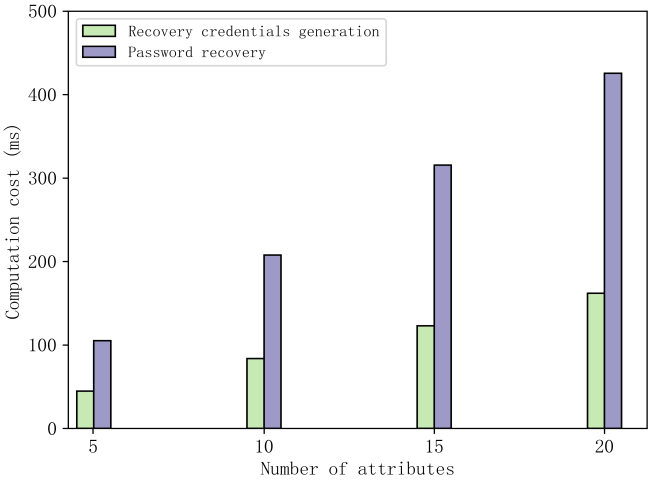

Fig. 11: Computation cost of dynamic password recovery.

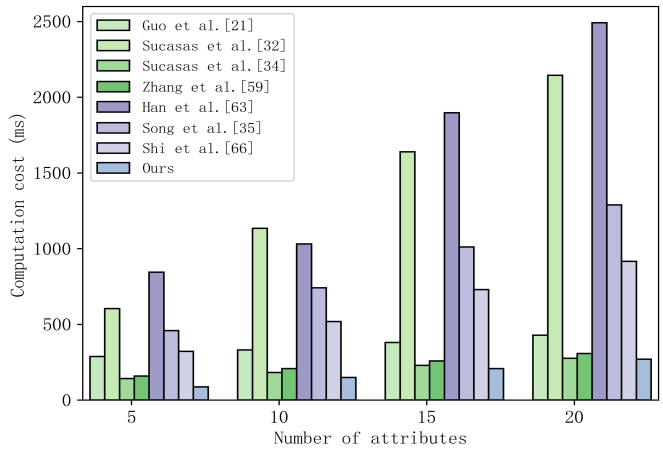

Fig. 10: Comparison of computation cost.

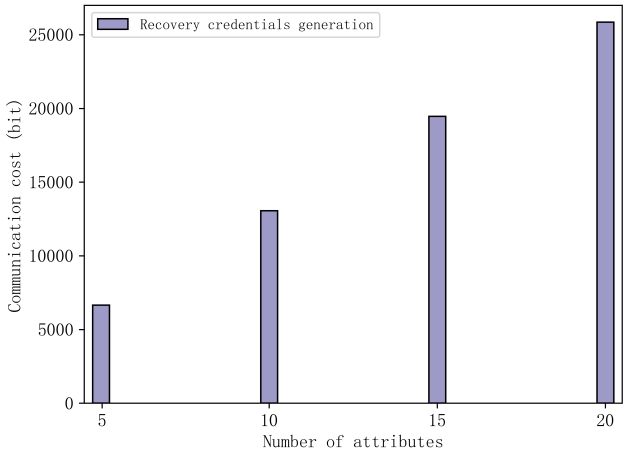

Fig. 12: Communication cost of dynamic password recovery.

TABLE IV: Performance of dynamic password recovery

Process	Computation cost	Communication cost
Generation	$nT_H + T_M + 2nT_E + T_{ENC}$	$2n\|G\| + (n+1)\|\mathbb{Z}_p\|$
Recovery	$nT_H + T_M + 2nT_E + 2nT_P + T_{ENC}$	-

TABLE V: Operation cost and description

Operation Cost	Description	Operation Cost	Description
$T_H \approx 1.413$	one hash operation	$T_F \approx 4.665$	one OPRF operation
$T_P \approx 6.827$	one bilinear paring operation	$T_B \approx 3.251$	one fuzzy extractor operation
$T_M \approx 3.652$	one modular multiplication operation	$T_S \approx 2.172$	one Symmetric Encryption operation
$T_E \approx 2.947$	one modular exponential operation	$T_A \approx 4.537$	one authentication encryption operation

can provide forward security and multi-factor security. MFA-DPRF can not only support dynamic password recovery and forward security but also achieve fine-grained access control.

For simplicity, we assume that the sizes of attributes, passwords, and biometrics are 60 bits, the size of random values, physically unclonable function outputs, and fuzzy extractor values are 160 bits. The size of the hash function and symmetric encryption output are 256 bits. Table III shows that MFA-DPRF is more efficient than Guo et al. [21], Sucasas et al. [32], Zhang et al. [59], Han et al. [63], and Sucasas et al. [34]. Compared to the state-of-the-art works [21], [32], [34], [35], [59], [63], [66], Fig. 9 shows that MFA-DPRF is more suitable for the resource-limited environment.

Additionally, to measure the cost of key operations, including bilinear pairing, hashing, modular exponentiation, multiplication, and symmetric encryption, we have conducted some simulations on a microcomputer equipped with an Intel(R) Core (TM) i5-12500H 2.50 GHz processor based on the Pairing-Based Cryptography (PBC) library. The simulation results are shown in Table V. Compared to some representative works [21], [32], [34], [35], [59], [63], [66], according to Fig. 10, it shows that MFA-DPRF is superior in efficiency.

Table IV presents the computation and communication costs of the dynamic password recovery method. To further enhance security, we perform attribute verification prior to executing the recovery algorithm. Only when the user's attributes meet the access policy can UR obtain recovery credentials from SC. Consequently, the recovery process requires $2t$ pairing operations, where t is the threshold. Fig. 11 and Fig. 12 illustrate the relationship between computation and communication cost and the number of attributes. Although computation and communication costs increase with the number of attributes, it is executed only when users need to recover passwords and will not affect each login and authentication phrase, thus balancing security and efficiency.

IX. Conclusion

To address the issue *" How to construct a UC secure MFA scheme with dynamic password recovery and fine-grained access control?"*, we propose MFA-DPRF in the UC framework. In MFA-DPRF, authentication succeeds only when a user provides the correct password, a valid smart card, and a set of attributes that satisfy the specified access policy, thereby enabling fine-grained access control. It also supports dynamic password recovery, allowing users to recover their passwords and update secret values to prevent privacy leakage. The security can be reduced to the CDH problem in the ROM model and can be proven secure within the UC framework. Compared to the state-of-the-art works, performance analysis shows that MFA-DPRF is superior in security and efficiency.

Acknowledgment

The authors gratefully acknowledge the editor and anonymous reviewers for their constructive and insightful comments, which have greatly enhanced the quality of this work. Additionally, the authors extend their sincere appreciation to Professor Peizhen Hong for her dedicated efforts in refining and strengthening the security analysis of the proposed scheme.

References

[1] M. Qiu, H.-N. Dai, A. K. Sangaiah, K. Liang, and X. Zheng, "Guest editorial: Special section on emerging privacy and security issues brought by artificial intelligence in industrial informatics," *IEEE Trans. Industr. Inform.*, vol. 16, no. 3, pp. 2029–2030, 2020.

[2] C. De Alwis, P. Porambage, K. Dev, T. R. Gadekallu, and M. Liyanage, "A survey on network slicing security: attacks, challenges, solutions and research directions," *IEEE Commun. Surv. Tutor.*, vol. 26, no. 1, pp. 534–570, 2024.

[3] D. Wang, D. He, P. Wang, and C. Chu, "Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment," *IEEE Trans. Dependable Secur. Comput.*, vol. 12, no. 4, pp. 428–442, 2015.

[4] C. C. Chang and H. D. Le, "A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks," *IEEE Trans. Wirel. Commun.*, vol. 15, no. 1, pp. 357–366, 2016.

[5] Q. Xie, D. S. Wong, G. Wang, X. Tan, K. Chen, and L. Fang, "Provably secure dynamic id-based anonymous two-factor authenticated key exchange protocol with extended security model," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 6, pp. 1382–1392, 2017.

[6] P. Gope, A. K. Das, N. Kumar, and Y. Cheng, "Lightweight and physically secure anonymous mutual authentication protocol for real-time data access in industrial wireless sensor networks," *IEEE Trans. Ind. Informatics*, vol. 15, no. 9, pp. 4957–4968, 2019.

[7] Q. Wang, D. Wang, C. Cheng, and D. He, "Quantum2fa: efficient quantum-resistant two-factor authentication scheme for mobile devices," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 1, pp. 193–208, 2023.

[8] R. Madhusudhan and R. C. Mittal, "Dynamic id-based remote user password authentication schemes using smart cards: A review," *J. Netw. Comput. Appl.*, vol. 35, no. 4, pp. 1235–1248, 2012.

[9] A. K. Das, S. Kumari *et al.*, "Provably secure user authentication and key agreement scheme for wireless sensor networks," *Secur. Commun. Netw.*, vol. 9, no. 16, pp. 3670–3687, 2016.

[10] D. Wang and P. Wang, "Two birds with one stone: two-factor authentication with security beyond conventional bound," *IEEE Trans. Dependable Secur. Comput.*, vol. 15, no. 4, pp. 708–722, 2018.

[11] F. Wu, X. Li, A. K. Sangaiah, L. Xu, S. Kumari, L. Wu, and J. Shen, "A lightweight and robust two-factor authentication scheme for personalized healthcare systems using wireless medical sensor networks," *Future Gener. Comput. Syst.*, vol. 82, pp. 727–737, 2018.

[12] A. K. Sutrala, M. S. Obaidat, S. Saha, A. K. Das, M. Alazab, and Y. Park, "Authenticated key agree- ment scheme with user anonymity and untraceability for 5g-enabled softwarized industrial cyber-physical systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2316–2330, 2022.

[13] M. Keith, B. Shao, and P. J. Steinbart, "The usability of passphrases for authentication: An empirical field study," *Int. J. Hum.-Comput. Stud.*, vol. 65, no. 1, pp. 17–28, 2007.

[14] S. Alroomi and F. Li, "Measuring website password creation policies at scale," in *Proc. CCS 2023*, pp. 3108–3122.

[15] "Juggling security: How many passwords does the average person have in 2024?" *NordPass*, https://nordpass.com/blog/how-many-passwords-does-average-person-have, 2024.

[16] N. Frykholm and A. Juels, "Error-tolerant password recovery," in *Proc. CCS 2001*, pp. 1–9.

[17] R. Hranickỳ, M. Holkovič, and P. Matoušek, "On efficiency of distributed password recovery," vol. 11, no. 2, p. 5, 2016.

[18] W. M. Kharudin, N. F. M. Din, and M. Z. Jali, "Password recovery using graphical method," in *Proc. PAISIT 2015*, pp. 11–20.

[19] Y. Li, Z. Chen, H. Wang, K. Sun, and S. Jajodia, "Understanding account recovery in the wild and its security implications," *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 1, pp. 620–634, 2020.

[20] L. Lassak, P. Markert, M. Golla, E. Stobert, and M. Dürmuth, "A comparative long-term study of fallback authentication schemes," in *Proc. CHI 2024*, 2024, pp. 1–19.

[21] L. Guo, C. Zhang, J. Sun, and Y. Fang, "A privacy-preserving attribute-based authentication system for mobile health networks," *IEEE Trans. Mob. Comput.*, vol. 13, no. 9, pp. 1927–1941, 2013.

[22] X. Yao, H. Kong, H. Liu, T. Qiu, and H. Ning, "An attribute credential based public key scheme for fog computing in digital manufacturing," *IEEE Trans. Industr. Inform.*, vol. 15, no. 4, pp. 2297–2307, 2019.

[23] K. Papadamou, S. Zannettou, B. Chifor, S. Teican, G. Gugulea, A. Caponi, A. Recupero, C. Pisa, G. Bianchi, S. Gevers *et al.*, "Killing the password and preserving privacy with device-centric and attribute-based authentication," *IEEE Trans. Inf. Forensic Secur.*, vol. 15, pp. 2183–2193, 2020.

[24] Z. Zhang, W. Huang, Y. Huang, Y. Liao, C. Wu, and S. Zhou, "An attribute-based pre-authenticated secure communication protocol enabling key protection and credential online-upgrading for 5g nr v2x," *IEEE Trans. Intell. Transp. Syst.*, 2025, doi:10.1109/TITS.2025.3558978.

[25] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[26] A. Bhargav-Spantzel, A. Squicciarini, and E. Bertino, "Privacy preserving multi-factor authentication with biometrics," in *Proc. DIM 2006*, 2006, pp. 63–72.

[27] X. Huang, Y. Xiang, A. Chonka, J. Zhou, and R. H. Deng, "A generic framework for three-factor authentication: Preserving security and privacy in distributed systems," *IEEE Trans. Parallel Distributed Syst.*, vol. 22, no. 8, pp. 1390–1397, 2011.

[28] S. Roy, S. Chatterjee, A. K. Das, S. Chattopadhyay, S. Kumari, and M. Jo, "Chaotic map-based anonymous user authentication scheme with user biometrics and fuzzy extractor for crowdsourcing internet of things," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2884–2895, 2017.

[29] A. M. A. Modarres and G. Sarbishaei, "An improved lightweight two-factor authentication protocol for iot applications," *IEEE Trans. Industr. Inform.*, vol. 19, no. 5, pp. 6588–6598, 2022.

[30] L. Zhang, Y. Zhu, W. Ren, Y. Zhang, and K.-K. R. Choo, "Privacy-preserving fast three-factor authentication and key agreement for iot-based e-health systems," *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1324–1333, 2022.

[31] J. Jiang, D. Wang, G. Zhang, and Z. Chen, "Quantum-resistant password-based threshold single-sign-on authentication with updatable server private key," in *Proc. ESORICS 2022*, pp. 295–316.

[32] V. Sucasas, G. Mantas, F. B. Saghezchi, A. Radwan, and J. Rodriguez, "An autonomous privacy-preserving authentication scheme for intelligent transportation systems," *Comput. Secur.*, vol. 60, pp. 193–205, 2016.

[33] G. Alpár, L. Batina, L. Batten, V. Moonsamy, A. Krasnova, A. Guellier, and I. Natgunanathan, "New directions in iot privacy using attribute-based authentication," in *Proc. ICCF 2016*, pp. 461–466.

[34] V. Sucasas, G. Mantas, M. Papaioannou, and J. Rodriguez, "Attribute-based pseudonymity for privacy-preserving authentication in cloud services," *IEEE Trans. Cloud. Comput.*, vol. 11, no. 1, pp. 168–184, 2021.

[35] M. Song and D. Wang, "Ab-pake: Achieving fine-grained access control and flexible authentication," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 6197–6212, 2024.
[36] V. Zimmermann, "From the quest to replace passwords towards supporting secure and usable password creation," *Ph.D. dissertation, Technische University of Darmstadt, Germany, 2021*.
[37] D. Weinshall and S. Kirkpatrick, "Passwords you'll never forget, but can't recall," in *Proc. CHI 2004*, pp. 1399–1402.
[38] C. Ellison, C. Hall, R. Milbert, and B. Schneier, "Protecting secret keys with personal entropy," *Futur. Gener. Comp. Syst.*, vol. 16, no. 4, pp. 311–318, 2000.
[39] M. Just and D. Aspinall, "Personal choice and challenge questions: a security and usability assessment," in *Proc. SOUPS 2009*, pp. 1–11.
[40] J. Bonneau, E. Bursztein, I. Caron, R. Jackson, and M. Williamson, "Secrets, lies, and account recovery: Lessons from the use of personal knowledge questions at google," in *Proc. WWW 2015*, pp. 141–150.
[41] M. Golla and M. Dürmuth, "Analyzing 4 million real-world personal knowledge questions," in *Proc. PASSWORDS 2015*, pp. 39–44.
[42] L. Zhu and D. Wang, "Robust multi-factor authentication for wsns with dynamic password recovery," *IEEE Trans. Inf. Forensic Secur.*, vol. 19, pp. 8398–8413, 2024.
[43] C. Jiang, C. Xu, Y. Han, Z. Zhang, and K. Chen, "Two-factor authenticated key exchange from biometrics with low entropy rates," *IEEE Trans. Inf. Forensic Secur.*, vol. 19, pp. 3844–3856, 2024.
[44] S. A. Chaudhry, A. Irshad, K. Yahya *et al.*, "Rotating behind privacy: An improved lightweight authentication scheme for cloud-based iot environment," *ACM Trans. Internet Techn.*, vol. 21, no. 3, pp. 78:1–78:19, 2021.
[45] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 11, pp. 2776–2791, 2017.
[46] Z. Hou and D. Wang, "New observations on zipf's law in passwords," *IEEE Trans. Inf. Forensic Secur.*, vol. 18, pp. 517–532, 2023.
[47] E. Stobert and R. Biddle, "The password life cycle," *ACM Trans. Priv. Secur*, vol. 21, no. 3, pp. 1–32, 2018.
[48] R. Little, L. Qin, and M. Varia, "Secure account recovery for a privacy-preserving web service," in *Proc. USENIX 2024*, 2024, pp. 1993–2010.
[49] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 2, pp. 1338–1351, 2022.
[50] S. Jarecki and X. Liu, "Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection," in *Proc. TCC 2009*. Springer, pp. 577–594.
[51] 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.
[52] P. Rogaway, "Authenticated-encryption with associated-data," in *Proc. CCS 2002*, 2002, pp. 98–107.
[53] R. Canetti, P. Jain, M. Swanberg, and M. Varia, "Universally composable end-to-end secure messaging," in *Proc. CRYPTO 2022*, pp. 3–33.
[54] D. Wang, W. Li, and P. Wang, "Measuring two-factor authentication schemes for real-time data access in industrial wireless sensor networks," *IEEE Trans. Ind. Informatics*, vol. 14, no. 8, pp. 4081–4092, 2018.
[55] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. FCS 2001*, pp. 136–145.
[56] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai, "Universally composable two-party and multi-party secure computation," in *Proc. STC 2002*, pp. 494–503.
[57] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. MacKenzie, "Universally composable password-based key exchange," in *Proc. EUROCRYPT 2005*, pp. 404–421.
[58] J. Bonneau, E. Bursztein, I. Caron, R. Jackson, and M. Williamson, "Secrets, lies, and account recovery: Lessons from the use of personal knowledge questions at google," in *Proc. WWW 2015*, pp. 141–150.
[59] Q. Zhang, Y. Gan, L. Liu, X. Wang, X. Luo, and Y. Li, "An authenticated asymmetric group key agreement based on attribute encryption," *J Netw. Comput, Appl.*, vol. 123, pp. 1–10, 2018.
[60] Y. Lu, G. Xu, L. Li, and Y. Yang, "Anonymous three-factor authenticated key agreement for wireless sensor networks," *Wirel. Networks*, vol. 25, no. 4, pp. 1461–1475, 2019.
[61] J. Srinivas, 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.*, vol. 8, no. 9, pp. 7727–7744, 2020.
[62] S. S. Sahoo, S. Mohanty, and B. Majhi, "A secure three factor based authentication scheme for health care systems using iot enabled devices," *J Ambient. Intell. Humaniz. Comput.*, vol. 12, no. 1, pp. 1419–1434, 2021.
[63] J. Han, L. Chen, S. Schneider, H. Treharne, and S. Wesemeyer, "Privacy-preserving electronic ticket scheme with attribute-based credentials," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1836–1849, 2019.
[64] A. K. Sutrala, M. S. Obaidat, S. Saha, A. K. Das, M. Alazab, and Y. Park, "Authenticated key agreement scheme with user anonymity and untraceability for 5g-enabled softwarized industrial cyber-physical systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2316–2330, 2021.
[65] A. Braeken, "Highly efficient bidirectional multi-factor authentication and key agreement for real-time access to sensor data," *IEEE Internet Things J.*, 2023.
[66] R. Shi, Y. Yang, Y. Li, H. Feng, G. Shi, H. H. Pang, and R. H. Deng, "Double issuer-hiding attribute-based credentials from tag-based aggregatable mercurial signatures," *IEEE Trans. Dependable Secur. Comput.*, vol. 21, no. 4, pp. 2585–2602, 2024.
[67] L. Wang, J. Xu, B. Qin, M. Wen, and K. Chen, "An efficient fuzzy certificateless signature-based authentication scheme using anonymous biometric identities for vanets," *IEEE Trans. Dependable Secur. Comput.*, vol. 22, no. 1, pp. 292–307, 2024.
[68] X. Yang, F. Zhu, X. Yang, J. Luo, X. Yi, J. Ning, and X. Huang, "Secure reputation-based authentication with malicious detection in vanets," *IEEE Trans. Dependable Secur. Comput.*, vol. 22, no. 1, pp. 359–372, 2024.
[69] T.-V. Le, M.-H. Yang, K. Mahmood, H. Taherdoost, C.-T. Li, C.-C. Lee, A. Ghani, and C.-T. Chen, "An enhanced multi-factor device authentication protocol in iolt healthcare environment," *IEEE Trans. Netw. Sci. Eng.*, vol. 12, no. 2, pp. 571–583, 2024.
[70] J. Han, L. Chen, and W. Susilo, "Privacy-preserving decentralized signature-based access control," *IEEE Trans. Dependable Secur. Comput.*, 2025, doi: 10.1109/TDSC.2025.3565210.

Liufu Zhu received the MS degree in information security from the Fujian Normal University, Fuzhou, P. R. China, in Jun. 2023. He is currently working toward the PhD degree from the College of Cyber Science, Nankai University, Tianjin, P. R. China. His research interests include public-key cryptography and cryptographic protocol.

Ding Wang received his Ph.D. degree in Information Security at Peking University in 2017, and was supported by the "Boya Postdoctoral Fellowship" in Peking University from 2017 to 2019. Currently, he is a Full Professor at Nankai University. As the first author (or corresponding author), he has published more than 120 papers at venues like IEEE S&P, ACM CCS, NDSS, USENIX Security, IEEE TDSC and IEEE TIFS. His research has been reported by over 200 media like Daily Mail, Forbes, IEEE Spectrum and Communications of the ACM, appeared in the Elsevier 2017 "Article Selection Celebrating Computer Science Research in China", and resulted in the revision of the authentication guideline NIST SP800-63. He has been involved in the community as a PC Chair/TPC member for over 60 international conferences such as NDSS 2023-2025, ACM CCS 2022, PETS 2022-2024, ACSAC 2020-2024, RAID 2024/2023, ACM AsiaCCS 2027/2025/2022/2021, FC 2026/2025, IFIP SEC 2018-2026, ICICS 2018-2026, SPNCE 2020-2022. He has received the "ACM China Outstanding Doctoral Dissertation Award", the Best Paper Award at INSCRYPT 2018, the Outstanding Youth Award of China Association for Cryptologic Research, the Young Scientist Nomination Award for Powerful Nation, and the First Prize of Natural Science Award of Ministry of Education. His research interests focus on the security of the secret key and provable security.

UC-Secure Multi-Factor Authentication with Dynamic Password Recovery and Fine-Grained Access Control (Appendix File)

Liufu Zhu and Ding Wang

Abstract—This appendix file consists of four parts. Appendix A shows the formal security analysis of MFA-DPRF under the ROM model, Appendix B proves the security of the password recovery, Appendix C provides UC security of MFA-DPRF, and Appendix D presents the informal analysis, such as forward security and dynamic password recovery.

Index Terms—Appendix file.

APPENDIX A: GAME-BASED SECURITY ANALYSIS UNDER THE ROM MODEL

Theorem 1. Assume that there is a probabilistic polynomial-time adversary $\mathcal{A}$ that executes q_E times of Extract(θ_{UR}, θ_{AGW}, θ_{SR}), q_S times of Send(θ_{UR}, θ_{AGW}, θ_{SR}, M), q_{H_i} times of $H_i(\cdot)_{i\in[1,5]}$, q_{enc} encryption and decryption queries and q_{mac} message authentication code queries. Then $\mathrm{Adv}_{\mathcal{A}}^{\text{MFA-DPRF}} \leq \sum_{i\in[1,4]} \frac{q_{H_i}^2 + q_S}{2^{l_{H_i}}} + Adv_{\mathcal{A}}^{AE} + \frac{q_{enc}^2}{2^{l_{enc}+1}} + \frac{q_{mac}^2}{2^{l_{mac}+1}} + 2C' q_S^{\eta'} + \frac{(q_E+q_S)^2 + 2q_S}{2^p} + \frac{q_{H_1} N \lambda_{asw_i}}{G_{asw_i}} + \sum_{i\geq t} \frac{q_S}{2^{l_{att_i}}} + q_{kgf}\varepsilon_{cdh}$.

C', η', λ_{asw_i} and G_{asw_i} are constants related to the password and private questions, l_i is the length of $H_i(\cdot)_{i\in[1,5]}$, l_{enc} is the length of the ciphertext, l_{att_i} is the length of att_i, and ε_{cdh} is the advantage in solving the CDH problem.

Security Model. Under the ROM model, MFA-DPRF is secure against various attacks. Let $\theta_{UR}, \theta_{AG}, \theta_{SR}$ denote the instance of the user, authentication gateway, and server, respectively. Suppose the existence of an efficient adversary $\mathcal{A}$ who will use $\Delta = \{\Omega_{UR}, \Omega_{SR}, \Omega_{AGW}\}$ to perform the following queries to break the security of the protocol.

(1) **Extract($\theta_{UR}, \theta_{AGW}, \theta_{SR}$):** This query simulates passive attacks, where $\mathcal{A}$ can eavesdrop on public channels and intercept communication data.

(2) **Send($\theta_{UR}, \theta_{AGW}, \theta_{SR}, M$):** This query simulates active attacks, where $\mathcal{A}$ can send M to θ_{UR}, θ_{AGW}, and θ_{SR}, and obtain some response about M during each step of the scheme.

(3) **Reveal(θ_{UR}, θ_{SR}):** When θ_{UR} and θ_{SR} successfully establish a session key K now, this oracle query sends it to $\mathcal{A}$ if θ_{UR} or θ_{SR} is compromised in the current session; otherwise, it returns a null symbol $\perp$.

(4) **Corrupt(θ_{UR}, b):** This oracle query allows sending any two authentication factors to $\mathcal{A}$. In addition, $\mathcal{A}$ also attempts to perform secret question guessing attacks when he compromises the smart card.
- **Corrupt($\theta_{UR}, 1$):** $\mathcal{A}$ obtains PW and SC.
- **Corrupt($\theta_{UR}, 2$):** $\mathcal{A}$ obtains Att and PW.
- **Corrupt($\theta_{UR}, 3$):** $\mathcal{A}$ obtains SC and Att.

(5) **Hash($\cdot$):** When executing this query, $\mathcal{A}$ receives random values. Lists $List_{h_i}$ will be created to store these queries and responses.

(6) **Test(θ_{UR}, θ_{SR}):** When θ_{UR} and θ_{SR} faithfully execute the protocol and establish a session key K, this query performs based on the outcome of a coin toss b. If $b = 1$, K is sent to $\mathcal{A}$; if $b = 0$, a random string $K^{'}$ is sent to $\mathcal{A}$. It requires that Corrupt(θ_{UR}, b) and Reveal(θ_{UR}, θ_{SR}) haven't been executed on θ_{UR} and θ_{SR}.

Semantic Security. The adversary $\mathcal{A}$, through the aforementioned queries, guesses the value of b via the Test(θ_{UR}, θ_{SR}) query as b'. At this point, the advantage of $\mathcal{A}$ winning the above interactive game can be defined as $\mathrm{Adv}_{\mathcal{A}}^{\text{MFA-DPRF}} = 2\Pr[b' = b] - 1$. Then, assume that $\mathcal{A}$ can perform **Execute(Δ)**, **Send(Δ, M)**, **Corrupt(Δ, b)**, **Hash($\cdot$)**, and **Test(Δ)** queries and then attempts to win the game with negligible advantage $\mathrm{Adv}_{\mathcal{A}}^{\text{MFA-DPRF}}$.

Proof. We define a series of games and specifies that $\mathcal{A}$ wins Game_i if he correctly guesses the coin flip b. Let $Pr[G\_i]$ represent the probability that $\mathcal{A}$ wins $Game\_i$.

Game_1. Game_1 simulates the attacks on the real scheme. Therefore, $\mathrm{Adv}_{\mathcal{A}^{\text{MFA-DPRF}}} = 2\Pr[G\_1] - 1$.

Game_2. In Game_2, $\mathcal{A}$ can perform Extract($\theta_{UR}, \theta_{AGW}$, θ_{SR}) queries. Then the message transmitted publicly can be sent to $\mathcal{A}$. After obtaining these messages, $\mathcal{A}$ outputs a guess b'. However, without knowing the secret values r_{us}, r_{gs} and r_{su} of θ_{UR}, θ_{AGW}, θ_{SR}, $\mathcal{A}$ cannot extract

Received on September 14th, 2024; revised on April 14th, 2026; accepted on May 17th, 2026. This work is supported in part by Tianjin Natural Science Foundation Project (Grant No. 25JCJQJC00230) and by the Fundamental Research Funds for the Central Universities, Nankai University (Grant No. 63263258). (Corresponding author: Ding Wang.)

Liufu Zhu and Ding Wang are with the College of Cryptology and Cyber Science, Nankai University, Tianjin 300350, China, with the State Key Laboratory of Cryptology, Beijing 100878, China, with Key Laboratory of Data and Intelligent System Security, Nankai University, Ministry of Education, Tianjin 300350, China, with the Tianjin Key Laboratory of Network and Data Security Technology, Nankai University, Tianjin 300350, China; also with the Beijing Key Laboratory of Post-Quantum Cryptography, Beijing 100083, China. (Email: zhuliufu@mail.nankai.edu.cn; wangding@nankai.edu.cn)

any valid information about b from the public message. Therefore, Game_2 is computationally indistinguishable from Game_1, with $\Pr[G\_2] = \Pr[G\_1]$.

Game_3. We initialize five hash lists $List_{h_i}$ to simulate the hash oracles $H_i(\cdot)$. These lists record the hash queries and responses activated by the adversary $\mathcal{A}$. When receiving encryption queries, an encryption and decryption list $List_{enc}$ will be created to record encryption queries and decryption responses. In addition, when receiving a message authentication code (MAC) query, a list $List_{mac}$ will be built to store MAC queries and responses. Notice that Game_3 is indistinguishable from Game_2, we have that $\Pr[G\_3] = \Pr[G\_2]$.

Game_4. Game_3 will terminate if a collision occurs in query responses. It includes the following cases:

(1). Collisions in the hash values when querying random oracles $H_i(\cdot)$;

(2). Collisions in other communication messages retrieved through queries: CT_1, σ_1, CT_2, σ_2, σ_3,.

Since (CT_1, σ_1) and (CT_2, σ_2) are generated using authenticated encryption scheme AES, and σ_3 is generated using the message authentication code MAC, We have that $|\Pr[G\_3] - \Pr[G\_2]| \leq \sum_{i\in[1,5]} \frac{q_{H_i}^2}{2^{l_{H_i}}} + \frac{q_{enc}^2}{2^{l_{enc}+1}} + \frac{q_{mac}^2}{2^{l_{mac}+1}} + \frac{(q_E+q_S)^2}{2^p}$.

Game_5. In Game_5, if $\mathcal{A}$ can successfully guess hash values without querying oracles H_i, this game will terminate. That is $|\Pr[G\_5] - \Pr[G\_4]| \leq \sum_{i\in[1,5]} \frac{q_S}{2^{l_{H_i}}}$.

Game_6. In Game_6, if $\mathcal{A}$ can successfully guess the random values r_{us}, r_{gs}, r_{su}, this game will terminate. We have that $|\Pr[G\_6] - \Pr[G\_5]| \leq \frac{q_S}{2^p}$.

Game_7. In Game_7, $\mathcal{A}$ performs a Corrupt(θ_{UR}, b) query. When executing Corrupt(θ_{UR}, 1), $\mathcal{A}$ attempts to guess valid attributes Att with probability $\sum_{i\geq t} \frac{q_S}{2^{l_{att_i}}}$; when executing Corrupt(θ_{UR}, 2), $\mathcal{A}$ attempts to break the private questions with probability $\frac{q_{H_1} N \lambda_{asw_i}}{G_{asw_i}}$ and guess the secret values with probability $\frac{q_S}{2^p}$; when executing Corrupt(θ_{UR}, 3), $\mathcal{A}$ attempts to guess the password PW with probability $C' q_S^{\eta'}$. Note that $\mathcal{A}$ can guess the secret message stored in SC via secret question guessing attacks in case Corrupt(θ_{UR}, 2). Accordig to the recent works, we have that the advantage of $\mathcal{A}$ successfully guessing private questions when the adversary performs q_{H_1} times H_1 hash queries is $\frac{q_{H_1} N \lambda_{asw_i}}{G_{asw_i}}$, where λ_{Ans_i} and G_{Ans_i} are constants related to the guessing entropy of the private question, q_H denotes the number of H_2 and H_3 hash queries. As a result, $|\Pr[G\_7] - \Pr[G\_6]| \leq 2C' q_S^{\eta'} + \frac{q_{H_1} N \lambda_{asw_i}}{G_{asw_i}} + \sum_{i\geq t} \frac{q_S}{2^{l_{att_i}}} + \frac{q_S}{2^p}$.

Game_8. In Game_8, we simulate the protocol execution via a computational Diffie-Hellman (CDH) problem instance defined in group G_1. In addition, we assume the existence of an adversary $\mathcal{B}$ who is given a CDH problem (g, g^α, g^β) and interacts with $\mathcal{A}$ as follows. $\mathcal{B}$ will emulate the authenticated encryption scheme as ideal functions F_{AE}, and then sets $g^{r_{us}} = g^\alpha$ and $g^{r_{su}} = g^\beta$. Specifically, $\mathcal{B}$ emulates the ideal functionality F_{AE} to generate CT_2 and σ_2. $\mathcal{B}$ also generates σ_3 randomly and sends CT_2, σ_2, σ_3 and g^β to $\mathcal{A}$. If $\mathcal{A}$ can successfully guess the value b in Game_8, we have that $\mathcal{A}$ has already queried the random oracle $KGF(\cdot)$ with input $(g^{r_{us}}, g^{r_{su}}, g^{r_{us}r_{su}}) = (g^\alpha, g^\beta, g^{\alpha\beta})$. By picking randomly from the list $List_{kgf}$, the challenger $\mathcal{B}$ gets the intended record with probability $\frac{1}{q_{kgf}}$. Finally, $\mathcal{B}$ can successfully solve the CDH problem with probability $\varepsilon_{cdh} \geq \frac{Adv(\mathcal{A})}{q_{kgf}}$. Since $\mathcal{A}$ knows $g^{r_{us}} = g^\alpha$ according to the ciphertexts CT_2, we have that $|\Pr[G\_8] - \Pr[G\_7]| \leq q_{kgf}\varepsilon_{cdh} + Adv_{\mathcal{A}}^{AE}$.

from **Game_1** to **Game_8**, the advantage of $\mathcal{A}$ in the above games is

$$\text{Adv}_{\mathcal{A}}^{MFA-DPRF} \leq \sum_{i\in[1,4]} \frac{q_{H_i}^2 + q_S}{2^{l_{H_i}}} + Adv_{\mathcal{A}}^{AE} + \frac{q_{enc}^2}{2^{l_{enc}+1}} + \frac{q_{mac}^2}{2^{l_{mac}+1}} + 2C' q_S^{\eta'} + \frac{(q_E+q_S)^2 + 2q_S}{2^p} + \frac{q_{H_1} N \lambda_{asw_i}}{G_{asw_i}} + \sum_{i\geq t} \frac{q_S}{2^{l_{att_i}}} + q_{kgf}\varepsilon_{cdh}.$$

APPENDIX B: SECURITY PROOF OF DYNAMIC PASSWORD RECOVERY

Theorem 2. Assuming the existence of an efficient adversary $\mathcal{A}$ who attempts to extract $\{Q_i\}_{1\leq N}$, PWC and $\{t_i\}_{i\in Q}$ from SC and recover the password, the advantage of $\mathcal{A}$ successfully recovering the password is no more than $Adv_{\mathcal{A}}^{AES} + \frac{q_{H_1} N \lambda_{asw_i}}{G_{asw_i}} + \frac{1}{2^{pN}} + \sum_{i\geq t, q_S^{'}\leq num_{rec}} \frac{q_S^{'}}{2^{l_{att_i}}}$, where $Adv_{\mathcal{A}}^{AES}$ denotes the advantage of $\mathcal{A}$ successfully breaking the AES algorithm; $\frac{\lambda_{asw_i}}{G_{asw_i}}$ denotes the advantage of guessing the secret question; q_{H_1} denotes the number of H_1 hash queries; $q_S^{'}$ denotes the number of **Send(Δ)**; $\frac{q_S^{'}}{2^{l_{att_i}}}$ denotes the advantage of guessing the attributes.

Proof. First, we assume that the smart card SC is kept secure. The adversary $\mathcal{A}$ needs to pass the authentication through attribute guessing attacks. Due to the application of fuzzy verifier technique and the rate-limit policy, after q_h times H_2 and H_3 hash queries, the advantage of the $\mathcal{A}$ successfully extracting the secret values is $\sum_{i\geq t, q_S^{'}\leq num_{rec}} \frac{q_S^{'}}{2^{l_{att_i}}}$; Suppose that $\sum_{i\geq t, q_S^{'}\leq num_{rec}} \frac{q_S^{'}}{2^{l_{att_i}}}$ has compromised the smart card and obtained the secret values $\{Q_i\}_{1\leq N}$, PWC and $\{t_i\}_{i\in Q}$. According to the definition, $\{M_{N\times L}\}$ is random matrix defined in $\mathbb{Z}_q^{n_2\times k_2}$. $v = (s, z_1, z_2, \ldots, z_L)$ is randomly distributed in $\mathbb{Z}_q^{1\times L}$. Let $\{v^{'}\}$ be a set of all possible matrix definetd in $\mathbb{Z}_q^L$ and $\{v^*\}$ be a set of matrixes related to t_i, where $\{v^{'}\} \subseteq \{v^*\}$. When achieving

t_i, the maximum advantage of $\mathcal{A}$ recovering the password depends on $\{v^*\}$. Define the maximum advantage

$$Pr_{\mathcal{A}}^{Max}[s] = \frac{Pr[H_2(asw_i||salt)]}{\sum_{H_2(asw_i^{'}||salt)\leftrightarrow t_i} Pr[H_2(asw_i^{'}||salt))]},$$

where $Pr[H_2(asw_i||salt)]$ represents the probability of $H_2(asw_i||salt)$ being inferred from t_i. When treating H_1 as a random oracle, $Pr[H_2(asw_i||salt)]$ is uniformally distributed and different from $Pr[H_2(asw_j||salt)]$ if $i \neq j$. Therefore, $Pr_{\mathcal{A}}^{Max}[\delta] \approx 1/|\{t_i\}| \approx 1/|\{v^{'}\}|$, which denotes that $\mathcal{A}$ successfully reconstructs $v = (s, z_1, z_2, \ldots, z_L)$, where $z_i \in [0, p-1]$. If $\mathcal{A}$ reconstructs v successfully, z_i has been guessed successfully. The probability for successfully guessing the secret s is $Pr_{\mathcal{A}}^{Max}[t_i] \leq 1/2^p$.

In order to analyze the maximum and minimum bound of $Pr[H_2(asw_i||salt)]$, we define Min_i and Max_i as follows,

$$Min_i = Min_{H_2(asw_i^{'}||salt)\leftrightarrow t_i} Pr[H_2(asw_i||salt)], and$$

$$Max_i = Max_{H_2(asw_i^{'}||salt)\leftrightarrow t_i} Pr[H_2(asw_i||salt)].$$

Then we have the following inequality

$$\sum Pr[H_2(asw_i^{'}||salt)] \geq \frac{Min_{H_2(asw_i^{'}||salt)\leftrightarrow t_i} Pr[H_2(asw_i)]}{|\{t_i\}|}$$

$$\approx \frac{1}{2^p} \prod_{i=1}^{n_2} Min_i.$$

Due to the independence of each question, $Pr_{\mathcal{A}}^{Max}[t_i] \leq \frac{1}{2^{pN}} \prod_{i=1}^{N} \frac{Max_i}{Min_i}$. It is infeasible for $\mathcal{A}$ to distinguish $H_2(asw_i||salt)$ and $H_2(asw_j||salt)$ under the random oracle model, which infers $Min_i \approx Max_i$. When achieving $\{t_i\}$, the maximum advantage $Pr_{\mathcal{A}}^{Max}[t_i]$ for $\mathcal{A}$ to recover the password is less than $\frac{1}{2^{pN}}$.

Note that, if $\mathcal{A}$ fails to achieve any useful information from $\{t_i\}$, it extracts PWC from SC and attempts to decrypt PWC. If $\mathcal{A}$ breaking the underlying AES algorithm successfully, the advantage is nearly $Adv_{\mathcal{A}}^{AES}$.

Assume that $\mathcal{A}$ can extract $\{Q_i\}_{1\leq N}$ from SC, it attempts to perform secret question guessing attacks to retrieve the correct answers. The advantage of $\mathcal{A}$ successfully guessing private questions is no more than $\frac{q_{H_1} N \lambda_{asw_i}}{G_{asw_i}}$.

Above all, the advantage of $\mathcal{A}$ successfully recovering the password is less than

$$Adv_{\mathcal{A}}^{AES} + \frac{q_{H_1} N \lambda_{asw_i}}{G_{asw_i}} + \sum_{i\geq t, q_S^{'} \leq num_{rec}} \frac{q_S^{'}}{2^{l_{att_i}}} + \frac{1}{2^{pN}}.$$

APPENDIX C: UC SECURITY OF MFA-DPRF

Theorem 4. The proposed protocol securely realizes the F_{MFA} in $(F_{\text{OPRF}}, F_{\text{AE}}, F_{\text{CRS}})$-hybrid model under the UC framework. Given a security parameter κ, for any real-world adversary $\mathcal{A}$, there exists an ideal-world simulator SIM where the environment $\mathcal{E}$ cannot distinguish whether it interacts with $\mathcal{A}$ and MFA-DPRF in the $(F_{\text{OPRF}}, F_{\text{AE}}, F_{\text{CRS}})$-hybrid model or with SIM and F_{MFA} in the ideal world,

$$Pr[\text{Real}_{\mathcal{A},\text{MFA-DPRF},\mathcal{E}}^{hybrid} = 1] \approx Pr[\text{Ideal}_{S,F_{\text{MFA}},\mathcal{E}} = 1].$$

Fig. 1 shows the ideal functionality F_{MFA}, which represents the desirable security properties of MFA-DPRF. The registration phase is captured by the query (REGIS, sid, $\mathcal{AGW}$, PW, Att, Q) from $\mathcal{UR}$, and (REGIS, sid, $\mathcal{AGW}$, ID_{SR}) from $\mathcal{SR}$. When receivig these queries, F_{MFA} records (regrec, sid, $\mathcal{UR}$, PW, Att, $\{asw_i\}_{i\in Q}$, SC) and (regrec, sid, $\mathcal{SR}$, s). Additionally, F_{MFA} sends (REGIS, sid, $\mathcal{AGW}$, $\mathcal{UR}$, $|PW|$, $|Att|$, $|att_i|$, $|asw_i|$) and (REGIS, sid, $\mathcal{AGW}$, $\mathcal{SR}$, $|s|$) to $\mathcal{SIM}$, denoting the leakage during the registration phase.

When receiving the login and authentication query (LOGAUTH, sid, $\mathcal{AGW}$, PW^*, Att_2^*, SC^*) from $\mathcal{UR}$ or $\mathcal{SIM}$, F_{MFA} records (logauthrec, sid, $\mathcal{UR}$, PW^*, Att^*, SC^*) and sends (LOGAUTH, sid, $\mathcal{AGW}$, $\mathcal{UR}$, $|PW^*|$, $|Att_2^*|$, $|att_i^*|$, $leak(state, SC^*)$) to $\mathcal{SIM}$. $|PW^*|$, $|Att_2^*|$, and $|att_i^*|$ denote the leakage of the password and attributes, $leak(state, SC^*)$ is the leakage of the smart card. If $\mathcal{SIM}$ has corrupted $\mathcal{UR}$ or $\mathcal{SR}$, it can receive the above values via the interface of the simulated adversary who communicates with the environment $\mathcal{E}$. $\mathcal{SIM}$ can control the network used to transmit the login and authentication request. Therefore, if receiving $REJECT$ from $\mathcal{SIM}$, F_{MFA} will return FAIL to $\mathcal{UR}$. Otherwise, F_{MFA} checks whether there is a record (regisrec, sid, $\mathcal{UR}$, PW, Att, SC, $\cdot$). If there exists such a record and $PW^* = PW \wedge |Att \cap Att_2^*| \geq t$, F_{MFA} returns "$Successful$" to $\mathcal{UR}$. If $PW^* \neq PW \vee |Att \cap Att_2^*| \geq t$, F_{MFA} returns "$Failed$" to $\mathcal{UR}$. Otherwise, ignore this query.

At the end of F_{MFA}, $\mathcal{UR}$ and $\mathcal{SR}$ obtain the same high-entropy session key K if $\mathcal{UR}$ inputs a correct password, valid attributes, and a smart card. Otherwise, the session key is set independently and randomly. When $\mathcal{UR}$ and $\mathcal{SR}$ are honest, $\mathcal{A}$ learns nothing about the session key. If $\mathcal{A}$ inputs(guesses) correct PW and Att and smart card SC, or corrupts $\mathcal{SR}$, $\mathcal{SIM}$ can determine the session key.

When receiving the password recovery query (RECPW, sid, Att_2^*, $\{asw_i^*\}_{i\in Q}$, SC^*) from $\mathcal{UR}$ or $\mathcal{SIM}$ ($\mathcal{SIM}$ has obtained compromised SC), F_{MFA} records (pwrec, sid, $\mathcal{UR}$, Att_2^*, $\{asw_i^*\}_{i\in Q}$, SC^*) and sends (RECPW, sid, $\mathcal{UR}$, $|Att_2^*|$, $|att_i^*|$, $|asw_i^*|$) to $\mathcal{SIM}$. If there exists no record (regrec, sid, $\mathcal{UR}$, PW, Att, $\{asw_i\}_{i\in Q}$, SC) stored during the registration phase, F_{MFA} will ignore this query. Otherwise, if $count_{rec} \leq num_{rec}$, $|Att_2^* \cup Att| \geq t$, and $\{asw_i^*\}_{i\in Q} = \{asw_i\}_{i\in Q}$, F_{MFA} will return PW to $\mathcal{UR}$ (or $\mathcal{SIM}$). Else if $count_{rec} > num_{rec}$, this query will be rejected. If $|Att_2^* \cup Att| < t$, or $\{asw_i^*\}_{i\in Q} \neq \{asw_i\}_{i\in Q}$, F_{MFA} will send a random $PW^{'}$ to $\mathcal{UR}$ (or $\mathcal{SIM}$).

Low-entropy secret information is always vulnerable to guessing attacks. Consequently, we capture password guessing attacks, attribute guessing attacks and private questions guessing attacks via the interface (TEST, sid, $\mathcal{UR}$, $PW^{'}$, $Att_2^{'}$, $\{asw_i^{'}\}_{i\in Q}$) in F_{MFA}. $\mathcal{SIM}$ can receive "$correctguess$" from F_{MFA} when it successfully guesses the password and attributes. Furthermore, this session will be marked as $COMPROMISED$. Otherwise, $\mathcal{SIM}$ will receive "$wrongguess$", and this session will be marked as interrupted. Additionally, if $\mathcal{SIM}$ has obtained some information about the private questions stored in the smart card through side channel attacks, it forwards $\{asw_i^{'}\}_{i\in Q}$ to F_{MFA} to conduct private questions guessing attacks.

Since the proposed MFA protocol applies the smart card to store some secret information, we capture side channel attacks using the interface (LEAKSC, sid, UR) where the adversary can obtain some leakage about the smart card through function $leak(state, SC)$.

Description of the Simulator. The simulator $\mathcal{SIM}$ starts by internally invoking an adversary $\mathcal{A}$, it runs a simulated interaction with the environment $\mathcal{E}$ and dummy parties. Specifically, $\mathcal{SIM}$ proceeds as follows:

Simulating the communication with $\mathcal{E}$: Every message that $\mathcal{SIM}$ receives from $\mathcal{E}$ will be given to $\mathcal{A}$. Each value that $\mathcal{A}$ outputs will be sent to $\mathcal{SIM}$ and then given to $\mathcal{E}$.

Simulating adversaries $\mathcal{A}$ who try to break the security of MFA-DPRF, $\mathcal{SIM}$ simulates as follows.

When initialized with security parameter κ, $\mathcal{SIM}$ chooses dummy $PW^{'}$ and $Att_2^{'}$ at random from space $|PW|$ and $|Att|$. Then $\mathcal{SIM}$ initializes the real-world adversary $\mathcal{A}$, giving $\mathcal{A}$ the well-designed (H_i, $KDF(\cdot)$, MAC, g, ϕ) as the common reference string, and interacts with $\mathcal{E}$, F_{MFA}, and $\mathcal{A}$. Specifically, SIM creates list $List_{h_i}$ to record corresponding queries and responses to $H_i(\cdot)$. For the most part, this interaction is implemented by $\mathcal{SIM}$ following MFA-DPRF on behalf of honest parties. The simulated parties use the dummy password and attributes and only receive (send) messages from (to) F_{MFA} or $\mathcal{SIM}$. We assume that the registration process is conducted through a secure channel, but it may leak some prior knowledge about the account to $\mathcal{SIM}$. Therefore, upon receiving (REGIS, sid, $\mathcal{AGW}$, $\mathcal{UR}$, $|PW|$, $|Att|$, $|att_i|$, $|asw_i|$) from F_{MFA}, $\mathcal{SIM}$ will create a record to store these information, satisfying $|PW^*| = |PW^{'}|$, $|Att_2^*| = |Att^{'}|$ and $|att_i^*| = |att_i^{'}|$, to simulate the honest $\mathcal{UR}$. $\mathcal{SIM}$ will emulate the functionality F_{OPRF}, and send OPRF value ω_i to $\mathcal{A}$. Additionally, $\mathcal{SIM}$ also emulates the functionality F_{AE} to generate ciphertexts (CT_i, σ_i) for $\mathcal{A}$.

During the login and authentication phase, if a user $\mathcal{UR}^{'}$ (may be legal or illegal) inputs $PW^{'}$ and $Att_2^{'}$ to attempt to login the system, $\mathcal{SIM}$ will receive receiving (LOGAUTH, sid, $\mathcal{AGW}$, $\mathcal{UR}$, $|PW^*|$, $|Att^*|$, $|att_i^*|$, $leak(state, SC^*)$) from F_{MFA}, learning the leakage $|PW^*|$, $|Att_2^*|$, $|att_i^*|$, $leak(state, SC^*)$ and extracting PW^* and attributes Att_2^*. Furthermore, $\mathcal{SIM}$ will make (TEST, sid, $\mathcal{UR}$, PW^*, Att_2^*) to F_{MFA} to determine whether PW^* and Att_2^* are correct. If these values are correct, $\mathcal{SIM}$ will replace the dummy values with the correct ones. Otherwise, it makes no change. If $\mathcal{A}$ modifies the publicly transmitted message during the login and authentication phase, $\mathcal{SIM}$ can also extract the malicious $PW^{'}$ and $Att_2^{'}$ and make a $TEST$ query to F_{MFA}.

Related Message. If a received message is formatted differently from what is expected, $\mathcal{SIM}$ will abort the session and notify $\mathcal{A}$. Otherwise, $\mathcal{SIM}$ will process as follows.

(1). Assuming that $\mathcal{UR}^{'}$ is malicious and attempts to login the system with inputs $PW^{'}$ and $Att_2^{'}$, $\mathcal{SIM}$ emulates the functionality F_{OPRF} and receives ω_i from $\mathcal{UR}^{'}$. $\mathcal{SIM}$ chooses a random oprf key k_u and returns the (EVALCOMPLETE, sid, $\mathcal{UR}$, sk_i) to $UR^{'}$. Additionally, $\mathcal{SIM}$ will search the list $List_{h_1}$ to determine the attribute $att_i^{'}$ chosen by $\mathcal{UR}^{'}$.

(2). When receiving idx_i, T_i from $\mathcal{UR}^{'}$, $\mathcal{SIM}$ creates X randomly where $X[idx_i] \in_R$ **CT**, it records idx_i and sends $X[idx_i]$ to $\mathcal{UR}^{'}$. Suppose that, $\mathcal{SIM}$ has obtained $att_i^{'}$ in (1), it can search $List_{h_3}$ to determine $Hpw_{\phi(att_i^{'})}$. If the number of attribute $|Att^{'}| = t$, $HPW^{'}$ can be recontrusted as $HPW^{'} = Hpw_1 \,||\, Hpw_2 \,||\, ... \,||\, Hpw_n$. Therefore, $\mathcal{SIM}$ can search $List_{h_1}$ to determine $PW^{'}$. $\mathcal{SIM}$ will make a query (TEST, $sid, \mathcal{UR}$, $PW^{'}$, $Att^{'}$, $\perp$) to F_{MFA}.

(3). In our analysis, we treat g^y as a public parameter that can be viewed as the public key of $\mathcal{AGW}$. Therefore, $\mathcal{SIM}$ can choose a random $y \in_R Z_p^*$ and send g^y to $\mathcal{UR}^{'}$. When receiving $C_4, CT_1, \sigma_1, g^{r_{ug}}, T_2$ from $\mathcal{UR}^{'}$, $\mathcal{SIM}$ generates the decryption key $k_{gu} = (g^{r_{ug}})^y$ and emulates F_{AE} to obtain $HPW \,||\, g^{us} \,||\, T_2$. $\mathcal{SIM}$ will search $List_{h_1}$ to determine $PW^{'}$ and make a query (TEST, $sid, \mathcal{UR}$, $PW^{'}$, $Att_2^{'}$, $\perp$) to F_{MFA}.

(4). Suppose that an adversary $\mathcal{A}$ is able to conduct side-channel attacks; therefore, it can extract secret information $\{Q, \{t_i\}_{i\in Q}, c_1, c_2, PWC\}$ from the smart card SC. When receiving λ_i from $\mathcal{A}$, $\mathcal{SIM}$ computes $H_1(H_2(asw_i^{'}||salt) \mod p) = \lambda_i \oplus t_i$ and searches $List_{h_3}$ to determine $asw_i^{'}$. Furthermore, $\mathcal{SIM}$ is able to make a query (TEST, $sid, \mathcal{UR}$, $\perp$, $\perp$, $\{asw_i^{'}\}_{i\in Q}$) to $asw_i^{'}$ to check whether $\{asw_i^{'}\}_{i\in Q}$ is correct.

(5). Other messages. All other messages are handled by $\mathcal{SIM}$ following the protocol. $\mathcal{SIM}$ returns an outcome to $\mathcal{A}$ when a session is terminated or aborted. If the session terminates with a session key K, then S makes a $NEWKEY$ query to F_{MFA}, specifying the session key K. (Unless the session is compromised, F_{MFA} will ignore the key specified by $\mathcal{SIM}$.)

Proof of the Indistinguishability. Let $\mathcal{A}$ be an adaptive adversary running MFA-DPRF. We show that for every $\mathcal{A}$, there is an ideal-world adversary $\mathcal{SIM}$ interacting with dummy parties and F_{MFA} such that no environment $\mathcal{E}$ can distinguish between the interaction with $\mathcal{A}$ running MFA-DPRF in the hybrid model or with $\mathcal{SIM}$ running F_{MFA} in the ideal world.

Accordingly, we define a sequence of hybrid experiments, starting from the real world and ending in the simulated world, where in each experiment we change some aspect of the simulation and show that $\mathcal{E}$ cannot distinguish any two consecutive experiments with non-negligible probability. The hybrid experiments are presented as follows:

Exp0: Let Exp0 be the real-world execution.

Exp1: Exp1 is the same as Exp0, except that it changes the generation of public parameters as follows: it creates four lists $List_{h_i}$, $1 \le i \le 4$, recording queries and responses to simulate hashing functions $H_i(\cdot)$. Additionally, it chooses a random value $y \in_R \mathbb{Z}_p^*$, and sends g^y to $\mathcal{A}$ as the public parameters. We have that Exp0 and Exp1 are indistinguishable, where $|Pr[Exp_0 = 1] - Pr[Exp_1 = 1]| \approx 0$.

Exp2: Exp2 is identical to Exp1 except that it calculates ω_i, idx_i, C_4, CT_1, and σ_1 with dummy values $PW^{'}$ and $Att_2^{'}$. Specifically, it chooses random values r_{ug} and r_{us}, generates $k_{gu} = (g^y)^{r_{ug}}$ and computes ω_i, idx_i, C_4, CT_1, and σ_1 following the protocol. Due to the randomness of r_{ug}, the security of authenticatic encryption AE and the random

Functionality F_{MFA}

The functionality F_{MFA} is parameterized by a security parameter κ. It sets a password recovery threshold num_{rec} and a value count_{rec} to record the recovery attempts. F_{MFA} interacts with the user $\mathcal{UR}$, the server $\mathcal{SR}$, and the simulator $\mathcal{SIM}$ via the following queries.

Registration:

- Upon receiving the query $(\text{REGIS}, sid, \mathcal{AGW}, PW, Att, Q)$ from $\mathcal{UR}$, if it is the first registration query, record $(\text{regisrec}, sid, \mathcal{UR}, PW, Att, Q, \{asw_i\}_{i\in Q}, SC)$ and make it fresh. If this is the second registration query and there exists a record $(\text{regisrec}, \mathcal{UR}, PW', Att', \{asw'_i\}_{i\in Q}, SC')$, then record $(\text{regisrec}, sid, \mathcal{UR}, PW, Att, Q, \{asw_i\}_{i\in Q}, SC)$. Finally, send $(\text{REGIS}, sid, \mathcal{AGW}, \mathcal{UR}, |PW|, |Att|, |att_i|, |asw_i|)$ to $\mathcal{SIM}$ and wait for response from $\mathcal{SIM}$:
  - ▷ If receives OK from $\mathcal{SIM}$, return $(\text{regisrec}, sid, SC)$ to $\mathcal{UR}$;
  - ▷ If receives $REJECT$ from $\mathcal{SIM}$, return $FAIL$ to $\mathcal{UR}$.
- Upon receiving a query $(\text{REGIS}, sid, \mathcal{AGW}, ID_{SR})$ from $\mathcal{SR}$, if it is the first registration query, choose a random value $s \in_R Z_p^*$, record $(\text{regisrec}, sid, SR, s)$, send $(\text{NewSess}, sid, SR, |PW|, |Att|, |att_i|)$ to $\mathcal{SIM}$ and wait for response from $\mathcal{SIM}$:
  - ▷ If receives OK from $\mathcal{SIM}$, return $(\text{regisrec}, sid, \mathcal{SR}, s)$ to $\mathcal{SR}$;
  - ▷ If receives $REJECT$ from $\mathcal{SIM}$: return $FAIL$ to $\mathcal{SR}$.

Login and Authentication:

- Upon receiving the query $(\text{LOGAUTH}, sid, PW^*, Att^*, SC^*)$ from $\mathcal{UR}$ (or from $\mathcal{SIM}$), first keep the record $(\text{logauthrec}, sid, \mathcal{AGW}, \mathcal{UR}, W^*, Att^*, SC^*)$. Then, send $(\text{LOGAUTH}, sid, |PW^*|, |Att^*|, |att_i^*|)$ to $\mathcal{SIM}$ and wait for a response from $\mathcal{SIM}$:
  - ▷ If receives OK from $\mathcal{SIM}$, check if there exist a record $(\text{regisrec}, sid, \mathcal{UR}, PW, Att, Q, \{asw_i\}_{i\in Q}, SC)$:
    - -If there exists a record, and $PW = PW^* \wedge |Att \cap Att^*| \geq t$, return "successful login and authentication" to $\mathcal{UR}$.
    - -Else, return $FAIL$ to $\mathcal{UR}$.
  - ▷ If receives $REJECT$ from $\mathcal{SIM}$, return $FAIL$ to $\mathcal{UR}$.

Key Generation:

- Upon receiving the query $(\text{NEWKEY}, sid, \mathcal{UR}/\mathcal{SR}, K^*)$ from $\mathcal{SIM}$, if there is a record $(\text{keyrec}, sid, \mathcal{UR}, \mathcal{SR}, \cdot)$ not marked $COMPLETED$:
  - ▷ If this session sid is marked as $COMPROMISED$, or $\mathcal{UR}/\mathcal{SR}$ is corrupted, set $K = K^*$
  - ▷ If this session sid is marked as $FRESH$, and a session key K' has been sent to $\mathcal{UR}/\mathcal{SR}$, set $K = K'$.
  - ▷ Else, set a random session key $K \in_R \{0,1\}^{klen}$.

Password Recovery:

- Upon receiving the query $(\text{RECPW}, sid, Att^*, \{asw_i^*\}_{i\in Q}, SC^*)$ from $\mathcal{UR}$ (or from $\mathcal{SIM}$), set $count_{rec}$+= 1 and keep the record $(\text{pwrec}, sid, \mathcal{UR}, Att^*, \{asw_i^*\}_{i\in Q}, SC^*)$ and then send $(\text{RECPW}, sid, |Att^*|, |att_i^*|, |asw_i^*|)$ to $\mathcal{SIM}$.
  - ▷ If there exists no record $(\text{regisrec}, sid, \mathcal{UR}, PW, Att, Q, \{asw_i\}_{i\in Q}, SC)$, ignore.
  - ▷ If $count_{rec}$+= 1 $> num_{rec}$, return $\perp$ to $\mathcal{UR}$. // rate-limiting mechanism
  - ▷ Else, if $|Att \cap Att^*| \geq t \wedge \{asw_i^*\}_{i\in Q} = \{asw_i\}_{i\in Q} \wedge count_{rec}$+= 1 $\leq num_{rec}$, return PW to $\mathcal{UR}$. // attributes are valid and answers are correct
  - ▷ If $|Att \cap Att^*| < t \vee \{asw_i^*\}_{i\in Q} \neq \{asw_i\}_{i\in Q}$, return a random string $PW' \in_R \{0,1\}^{|PW|}$ to $\mathcal{UR}$. // incorrect answers or invalid attributes will lead to a random password

Active Attacks:

- Upon receiving the query $(\text{TEST}, sid, \mathcal{UR}, PW', Att', \{asw'_i\}_{i\in Q})$ from $\mathcal{SIM}$, if there exists no record $(\text{regisrec}, sid, \mathcal{UR}, PW, Att, Q, \{asw_i\}_{i\in Q}, SC)$, ignore; Else, do the following:
  - ▷ If $PW' = PW \wedge |Att \cap Att'| \geq t$, mark this session sid $COMPROMISED$ and return "correct guess" to $\mathcal{SIM}$. // password and attribute guessing attacks
  - ▷ Else, mark this session sid interrupted and return "wrong guess" to $\mathcal{SIM}$.
  - ▷ If $\{asw'_i\}_{i\in Q} = \{asw_i\}_{i\in Q}$, return "correct answers" to $\mathcal{SIM}$. // secret question guessing attacks
  - ▷ Else, return "wrong answers" to $\mathcal{SIM}$.
- Upon receiving the query $(\text{LEAKSC}, sid, \mathcal{UR})$ from $\mathcal{SIM}$, if there exists no record $(\text{regisrec}, sid, \mathcal{UR}, \cdot, \cdot, \cdot, \cdot, SC)$, ignore; Otherwise, return $leak(state, SC)$ to $\mathcal{SIM}$. // side channel attacks against the smart card

Fig. 1: The ideal functionality F_{MFA}. It interacts with $\mathcal{UR}$, $\mathcal{SR}$, and $\mathcal{SIM}$ via the REGIS query, LOGAUTH query, NEWKEY query, and RECPW query. Furthermore, it captures active attacks, including password guessing attacks, attribute guessing attacks, secret question guessing attacks, and side channel attacks, via the interface TEST and LEAKSC. Upon guessing the password and attributes successfully, the session *sid* is marked COMPROMISED.

oracle H_1, we have that $|Pr[Exp_0 = 1] - Pr[Exp_1 = 1]| \leq Adv_{\mathcal{A}}(AE) + \frac{1}{2^p} + \frac{1}{2^{l_{h_1}}}$.

Exp3: A random session key K will be chosen for UR that receives a oracle-generated σ_3, $g^{r_{su}}$ from a peer session party SR. If UR and SR are matching session party, and SR has received a oracle-generated message C_5, CT_2, σ_2, $g^{r_{gs}}$, the same session key K is distributed to SR. We have that $|Pr[Exp_3 = 1] - Pr[Exp_2 = 1]| = 0$.

Exp4: Upon receiving idx_i, C_4, CT_1,$g^{r_{ug}}$, and σ_1 that are different from the previously generated, it checks whether the underlying PW^* and Att_2^* are valid. It emulates the instance of OPRF with the input ω_i^*. It search $List_{h_1}$ to determine the attribute att_i^*. It computes $k_{gu} = g^{r_{ug}y}$ and emulates the instance of the authenticated encryption AE to obtain $HPW||g^{r_{us}}||T_2 = AE.Dec(CT_2, k_{gu})$. It then searches $List_{h_1}$ to determine PW^*. When obtaining (PW^*, Att_2^*), it makes a query (TEST, sid, UR, $PW^{'}$, $Att_2^{'}$, $\cdot$) to F_{MFA}. If receives "correct guess" from F_{MFA}, it records (PW^*, Att_2^*) and replace dummy values $(PW^{'}, Att_2^{'})$ with the correct ones. Otherwise, it makes no change. We have that $|Pr[Exp_4 = 1] - Pr[Exp_3 = 1]| = 0$.

Exp5: A random session key K will be chosen for SR that receives an oracle-generated C_5, CT_2, σ_2, $g^{r_{gs}}$ from AGW, where AGW received a peer-oracle-generated C_4, CT_1, σ_1, $g^{r_{us}}$ from UR. If SR and AGW are matching session parties, then the session key of SR is set randomly (see Exp3) and not changed in this experiment. We have that $|Pr[Exp_5 = 1] - Pr[Exp_4 = 1]| = 0$.

Exp6: It checks whether σ_3, $g^{r_{us}}$ is valid for UR. To perform this validity test, it searches the record r_{us} (see Exp2) and computes $K = KGF(g^{r_{us}}||g^{r_{su}}||g^{r_{us}r_{su}})$. If $\sigma_3, g^{r_{su}}$ is valid, then the same session key K is outputted ot UR. Otherwise, a random session key will be sent for UR. Due to the difficulty of the computational Diffie-Hellman problem in G_1, we have that $|Pr[Exp_6 = 1] - Pr[Exp_5 = 1]| \leq Adv_{\mathcal{A}}(CDH)$.

Exp7: When receiving λ_i from UR, it computes $H_3(asw_i) = \lambda_i \oplus t_i$ and then search for $List_{h_3}$ to find asw_i^*. Furthermore, it will make a query (TEST, sid, UR, $\cdot$, $\cdot$, $\{asw_i^*\}_{i \in Q}$) to F_{MFA}. If it receives "correct answers" from F_{MFA}, it can recover the correct password PW and send it to UR. Otherwise, it chooses a random value $PW^{'} \in |PW|$ and forwards it to UR. We have that $|Pr[Exp_7 = 1] - Pr[Exp_6 = 1]| = 0$.

Exp8: Exp8 is the simulated world. The change from Exp7 to Exp8 is merely conceptual, with the challenger split into the OPRF functionality F_{OPRF}, the authenticated encryption functionality F_{AE}, and the simulator SIM. We have that $|Pr[Exp_8 = 1] - Pr[Exp_7 = 1]| = 0$.

APPENDIX D: INFORMAL SECURITY ANALYSIS ON MFA-DPRF

This section presents an informal analysis of MFA-DPRF, showing that it can resist known attacks such as password guessing attacks, user impersonation attacks, node capture attacks, and achieves multi-factor security, dynamic password recovery, and forward security.

(1) Resist password guessing attacks. MFA-DPRF employs the fuzzy verifier technique to mix up the distribution of passwords, effectively reducing the risk of password guessing attacks. If the adversary $\mathcal{A}$ intercepts the publicly transmitted message idx_i, C_4, CT_1 and σ_1, since $idx_i = H_3(Hpw_{\phi(att_i)} \,||\, att_i)$ where $HPW \rightarrow (Hpw_1 \,||\, Hpw_2 \,||\, ... \,||\, Hpw_{2n})$, it is difficult for $\mathcal{A}$ to guess the password PW even if he may provide a valid attribute because he cannot determine all shares Hpw_i. Due to the randomness of $g^{r_{ug}}$, r_{gs} and the security of the authenticated encryption scheme AE, it is impossible for $\mathcal{A}$ to extract information about PW from C_4, CT_1 and σ_1.

(2) Resist node capture attacks. In this attack, $\mathcal{A}$ obtains the SR's secret s and attempts to recompute the previously generated session keys by UR and SR. Since AGW sends CT_2, σ_2, $g^{r_{gs}}$, T_3 to SR in a public manner, $\mathcal{A}$ can generate $g^{sr_{gs}}$ and decrypt CT_2 to obtain $g^{r_{us}}$. Similarly, $\mathcal{A}$ can achieve $g^{r_{su}}$ from the publicly transmitted message σ_3, $g^{r_{su}}$, T_4 from SR to UR. However, due to the difficulty of CDH problem in G_1, it is infeasible for $\mathcal{A}$ to compute $g^{r_{us}r_{su}}$ from $g^{r_{us}}$ and $g^{r_{su}}$. As a result, $\mathcal{A}$ fails to reconstruct the previously generated session key K.

(3) Resist insider attacks. $\mathcal{A}$ acts as an insider who can achieve the registration information $X[idx_i] = AES.ENC(x_i,\ sk_i)$, $C_1 = H_4(H_1(HPW) \,||\, H_1(x) \ mod\ n_p)$ $C_2 = PID_u \oplus C_1$, $C_3 = H_1(H_5(g^{xy}) \,||\, PID_u)$, PID_u, g^x and g^y but cannot obtain any information about the password PW, attributes Att. $\mathcal{A}$ attempts to recover PW and Att. Since $idx_i = H_3(Hpw_i \,||\, att_i)$, it is difficult for $\mathcal{A}$ to determine whether Hpw_i is the intended share of PW and att_i is a valid attribute. Without the knowledge of x, $\mathcal{A}$ cannot perform guessing attacks to determine PW through C_1. In addition, due to the difficulty of the CDH problem in G_1, it is infeasible for $\mathcal{A}$ to compute the secret g^{xy} even with the knowledge of g^x and g^y to compute $C_3 = H_1(H_5(g^{xy}) \,||\, PID_u)$. Additionally, $\mathcal{A}$ is unable to compute HPW and fails to compute $C_4 = H_1(HPW \,||\, PID_u \,||\, H_5(g^{r_{ug}}))$ to impersonate UR using PID_u.

(4) Resist impersonation attacks. According to (3), it is infeasible for $\mathcal{A}$ to impersonate a valid user UR even if $\mathcal{A}$ knows the registration information and publicly transmitted message. To impersonate a server, $\mathcal{A}$ needs to achieve secret s. According to the construction of the proposed protocol, the publicly transmitted message CT_2, σ_2, $g^{r_{gs}}$, σ_3, and $g^{r_{su}}$ reveal nothing about the secret, so $\mathcal{A}$ fails to decrypt CT_2 to obtain $g^{r_{us}}$. Therefore, it is impossible for $\mathcal{A}$ to impersonate SR to establish the session key with UR.

(5) Resist key compromise impersonation (KCI) attacks. On the one hand, we assume that the long-term secret y of AGW is compromised and $\mathcal{A}$ attempts to impersonate UR. However, without the knowledge of x and PID_u, it is difficult to compute $C_3 = H_1(H_5(g^{xy})||PID_u)$. On the other hand, we assume that the long-term secret s of SR is compromised. In this case, $\mathcal{A}$ attempts to establish a session key with UR. According to analysis in (2), the session key is generated as $K = KGF(g^{r_{us}}, g^{r_{su}}, g^{r_{us}r_{su}})$. Due to the difficulty of CDH problem in G_1, it is infeasible for $\mathcal{A}$ to compute $g^{r_{us}r_{su}}$ from $g^{r_{us}}$ and $g^{r_{su}}$. As a result, $\mathcal{A}$ fails to reconstruct the previously generated session key. Therefore, the proposed protocol can resist KCI attacks.

(6) Resist desynchronization attacks. During the update phase, any modifications to PW or Att necessitate successful verification by SC, significantly lowering the risk

of unauthorized access and attacks. During the authentication and login phase, even if $\mathcal{A}$ intercepts the initial message, he cannot require modifications to secret values in SR's database. So these interceptions don't compromise the integrity or consistency of future communications. Furthermore, subsequent interceptions of message flows don't disrupt the ongoing communication between UR and SR, maintaining the continuity and stability of interactions despite potential security challenges.

(7) Resist man-in-the-middle attacks. Assume that $\mathcal{A}$ can intercept the publicly transmitted message C_4, CT_1, σ_1, CT_2, σ_2, $g^{r_{gs}}$, σ_3, and $g^{r_{ug}}$. Since the authenticated encryption scheme is applied to generate the ciphertexts and authentication tags CT_1, σ_1, CT_2, σ_2, and a message authentication code is used to generate σ_3, it is infeasible for $\mathcal{A}$ to modify these messages. Additionally, without the knowledge of PW and Att, $\mathcal{A}$ cannot generate the correct C_4^*. Therefore, the proposed protocol can resist man-in-the-middle attacks.

(8) Resist message replay attacks. According to the construction, the introduction of the timestamp T_2, T_3,T_4 in each session for computing $(CT_1, \sigma_1) = F_{\text{AE}}.Enc(m_1 = g^{r_{us}}||T_2, k_{gu} = g^{xy})$, $(CT_2, \sigma_2) = F_{\text{AE}}.Enc(m_1||T_3, k_{gs})$ and $\sigma_3 = MAC(K \mid\mid T_4 \mid\mid g^{r_{us}} \mid\mid g^{r_{su}})$ ensures freshness, thus preventing $\mathcal{A}$ from performing message replay attacks.

(9) Mutual Authentication. During the login and authentication phase, AGW computes $C_4 = H_1(HPW \mid\mid PID_u \mid\mid H_5(g^{r_{ug}}))$ and checks if $C_4^* = C_4$. SR computes $k_{gs} = (g^{r_{gs}})^s$, $PID_s||g^{r_{us}}||T_3 = F_{\text{AE}}.Dec(CT_2, k_{gs})$ and checks if $C_5^* = C_5$. UR computes $K = KGF(g^{r_{us}}, g^{r_{su}}, g^{r_{us}r_{su}})$ and checks if $\sigma_3^* = \sigma_3$. If the above mutual authentication succeeds, a session key K will be established.

(10) Real-time verification. During the login and authentication phase, if $C_3^* = H_1(H_5(g^{xy})||PID_u) = C_3$, UR can extract the information stored in SC. If $|T_1^{'} - T_1| \leq \Delta T$, and $count_u \geq t$,and $C_4^* = H_1(HPW \mid\mid PID_u \mid\mid H_5(g^{r_{ug}})) = C_4$, AGW will succussfully verifies the identity of UR.

(11) Multi-factor security. $\mathcal{A}$ can compromise two factors from Att, PW, and SC but not all of them.

a. If $\mathcal{A}$ has compromised PW and SC, she attempts to compromise Att. According to the protocol, $idx_i = H_3(Hpw_{\phi(att_i)} \mid\mid att_i)$, $c_1 = g^{r_i^{'}}$, and $c_2 = H_2(H_3(att_i \mid\mid EML \mid\mid RID) \bmod n_a)^{r_i^{'}}$ are related to Att that can be obtained by $\mathcal{A}$; On the one hand, idx_i transmitted publicly during the login and authentication phase may be generated by some invalid attributes att_i^* in Att_2^*. Therefore, it is difficult to determine which attribute is valid. On the other hand, due to the randomness of $r_i^{'}$, $\mathcal{A}$ cannot compute att_i from c_1 and c_2 if the smart card is kept secure. In addtion, according to Appendix B, the advantage of $\mathcal{A}$ correctly guessing the attributes is less than $\sum_{i \geq t, q_S^{'} \leq num_{rec}} \frac{q_S^{'}}{2^{l_{att_i}}}$ when the smart card is compromised because of the application of the fuzzy verifier technique and the rate-limit policy.

b. If $\mathcal{A}$ has compromised Att and PW, she attempts to compromise SC. Assuming that private questions are secure against guessing attacks, $\mathcal{A}$ cannot compute $\{t_i\}_{i \in Q}$. Therefore, $\mathcal{A}$ cannot compromise SC.

c. If $\mathcal{A}$ has compromised Att and SC, she attempts to compromise PW. First, $\mathcal{A}$ can obtain $idx_i = H_3(Hpw_{\phi(att_i^*)} \mid\mid att_i^*)$ transmitted during the login and authentication phase. Note that idx_i may be generated by some invalid attributes att_i^* in Att_2^* so that $\mathcal{A}$ may recover some incorrect shares of HPW, making it hard to recover PW. On the other hand, due to the security of private questions, it is difficult for $\mathcal{A}$ to recover η through $\{t_i\}_{i \in Q}$. Therefore, $\mathcal{A}$ cannot decrypt PWC to recover PW.

(12) Anonymity and fine-grained access control. By employing attributes Att instead of the actual identity of the user, the proposed protocol ensures anonymity because users who can provide valid attributes may pass the authentication; additionally, fine-grained access control is achieved since if and only if users whose attributes satisfy the access policy can succeed in authentication.

(13) Dynamic password recovery. If UR forgets or loses her password, she can recover it locally without interacting with other parties. It requires UR to insert a valid smart card SC, whose attributes satisfy the access policy. After successful verification, UR is required to correctly answer private questions Q in order to compute secret values η from $\{t_i\}_{i \in Q}$. ur then performs decryption $AES.DEC(PWC, \eta) = PW$ to recover PW. Furthermore, UR is able to change private questions $\{Q_i\} \rightarrow \{Q_i\}^{new}$ and update secret values $\eta \rightarrow \eta^{new}$ dynamicaly: UR replaces $Q_i \rightarrow Q_i^{new}$, modify $(M, \rho) \rightarrow (M^{new}, \rho^{new})$; chooses a new secret η^{new} and $v^{new} = (\eta^{new}, z_1^{new}, z_2^{new}, \ldots, z_L^{new})^T$, and computes $\lambda_i^{new} = M^{new}(i) \times v^{new}$, $t_i^{new} = H_1(asw_i^{new}||salt) \oplus \lambda_i^{new}$, $PWC^{new} = AES.ENC(PW, \eta^{new})$.

(14) Forward Security. In each session, UR, AGW, and SR independently choose new random values r_{us}, r_{gs}, r_{su}. This ensures that even if the long-term secret key y is compromised and obtain $g^{r_{us}}$ and $g^{r_{su}}$, due to the difficulty of the CDH problem in G_1, it is infeasible for $\mathcal{A}$ to compute $g^{r_{us}r_{su}}$ to reconstruct session key K.