entropy

MDPI

Article

Post-Quantum Secure Multi-Factor Authentication Protocol for Multi-Server Architecture

Yunhua Wen *, Yandong Su and Wei Li

School of Computer Science and Technology, Donghua University, Shanghai 201620, China; 2222821@mail.dhu.edu.cn (Y.S.); weili@dhu.edu.cn (W.L.)
* Correspondence: yhwen@dhu.edu.cn

Abstract

The multi-factor authentication (MFA) protocol requires users to provide a combination of a password, a smart card and biometric data as verification factors to gain access to the services they need. In a single-server MFA system, users accessing multiple distinct servers must register separately for each server, manage multiple smart cards, and remember numerous passwords. In contrast, an MFA system designed for multi-server architecture allows users to register once at a registration center (RC) and then access all associated servers with a single smart card and one password. MFA with an offline RC addresses the computational bottleneck and single-point failure issues associated with the RC. In this paper, we propose a post-quantum secure MFA protocol for a multi-server architecture with an offline RC. Our MFA protocol utilizes the post-quantum secure Kyber key encapsulation mechanism and an information-theoretically secure fuzzy extractor as its building blocks. We formally prove the post-quantum semantic security of our MFA protocol under the real or random (ROR) model in the random oracle paradigm. Compared to related protocols, our protocol achieves higher efficiency and maintains reasonable communication overhead.

Keywords: multi-factor authentication; fuzzy extractor; real or random model; post-quantum security; key encapsulation mechanism; lattice-based cryptography

Academic Editor: Giuliano Benenti

Received: 24 May 2025
Revised: 28 June 2025
Accepted: 7 July 2025
Published: 18 July 2025

Citation: Wen, Y.; Su, Y.; Li, W. Post-Quantum Secure Multi-Factor Authentication Protocol for Multi-Server Architecture. *Entropy* **2025**, *27*, 765. https://doi.org/10.3390/e27070765

1. Introduction

With the rapid development of network technology, online services such as information inquiry, social entertainment, business transactions, and government affair handling have become integral to daily life. However, as more sensitive data are transmitted over public channels, ensuring secure access to these services has become increasingly critical. Remote authentication protocols are essential to prevent unauthorized access, eavesdropping, and tampering by malicious entities. Traditional single-factor or two-factor authentication mechanisms [1,2] are no longer sufficient due to vulnerabilities such as password guessing and lost smart card attacks [3]. As a result, multi-factor authentication (MFA) protocols [4] combining passwords, smart cards, and biometric data have been widely adopted for stronger security.

Most existing MFA [5–7] protocols, however, are designed for single-server architectures, requiring users to register separately with each service provider—a cumbersome process that leads to significant management overhead. To address this, multi-server MFA protocols have been proposed [8], allowing users to access multiple servers after a single registration. The multi-sever MFA protocol consists of a set of users, a set of service severs and a registration center (RC), as shown in Figure 1. All users and service severs need to

register on the RC. After registration, the user can access different service servers with the same password, smart card and biometric data.

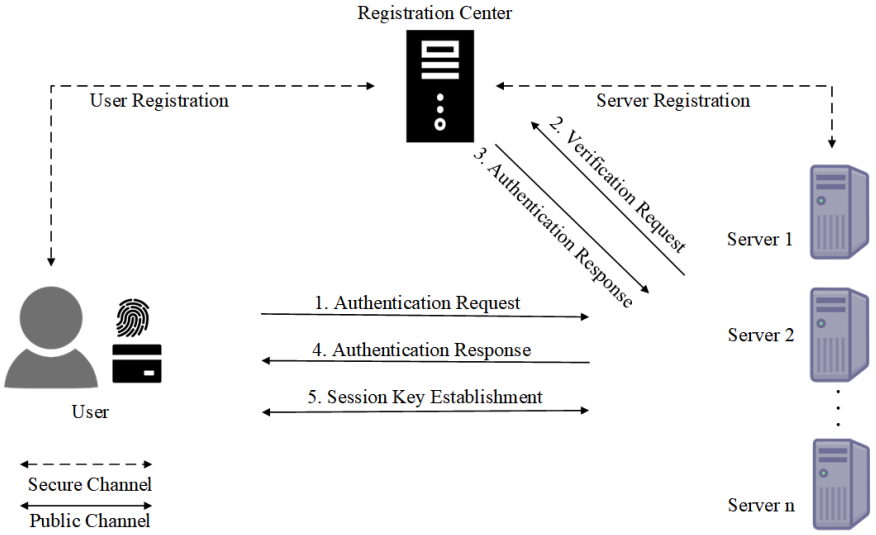

Figure 1. MFA for the multi-server architecture with online RC.

Some multi-sever protocols employ an online registration center (RC). The integration of an RC to authenticate each user–server pair increases communication overhead on the channel, introduces computational bottlenecks, and creates single-point failure issues at the RC.

To address the computational bottleneck problem of the online RC, a multi-server MFA protocol using an offline RC was proposed [9]. As shown in Figure 2, in an offline RC architecture, the RC distributes long-term information to both the user and the service server during the registration phase. This enables the two entities to mutually authenticate each other without involving the RC over a public channel. Offline RC-based solutions often lack support for post-quantum security or suffer from other vulnerabilities, such as denial-of-service (DoS) attacks, lack of user anonymity, or inefficient revocation mechanisms.

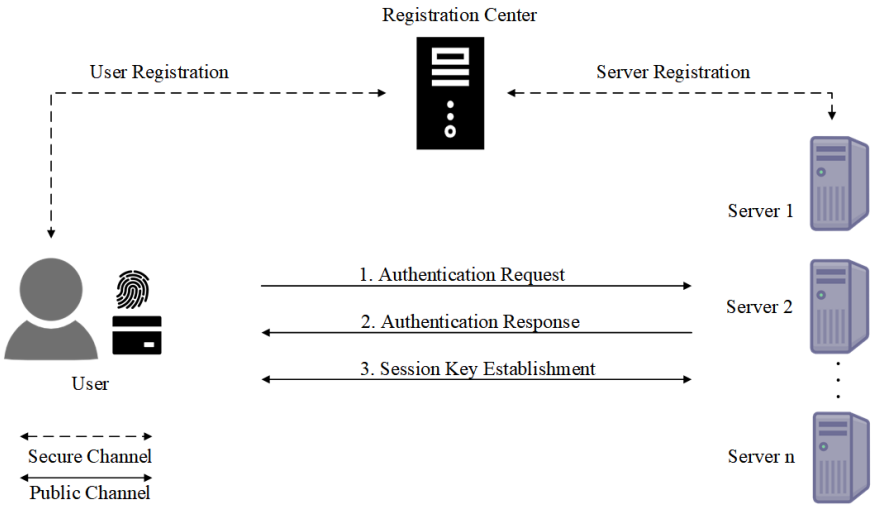

Figure 2. MFA for a multi-server architecture with an offline RC.

Given the threat posed by quantum computing to traditional cryptographic schemes like elliptic curve cryptography (ECC), it is crucial to develop MFA protocols that are not only efficient and secure in a multi-server environment but also resilient to quantum attacks. Despite numerous efforts in this domain, there remains a gap: no existing protocol

simultaneously supports multi-server architectures, employs an offline RC, ensures user anonymity, and guarantees post-quantum security. This work aims to fill this gap by proposing a novel lattice-based MFA protocol tailored for multi-server environments with an offline RC, addressing both current and future security challenges.

Our Contributions

In this paper, we propose a multi-factor authentication protocol that leverages a fuzzy extractor (FE) and a Kyber key encapsulation mechanism (KEM). Our protocol ensures post-quantum security, supports an offline registration center (RC), and operates effectively in a multi-server architecture.

- In our protocol, both users and servers must initially register with the RC via a secure channel. After registration, any user can conduct mutual authentication with registered servers over an insecure channel, effectively negotiating a secure session key even when the RC is offline. Furthermore, our protocol supports users in updating their password and revoking their smart cards as needed.
- To achieve post-quantum security, our MFA protocol incorporates the post-quantum secure Kyber KEM and an information-theoretically secure fuzzy extractor as core components. The fuzzy extractor addresses minor variations in biometric inputs and prevents the storage of raw biometric data on the smart card, thereby reducing the risk of potential information leakage.
- We have proven the semantic security of our MFA protocol under the ROR (real or random) model. Furthermore, we conducted a comparative analysis with related protocols, demonstrating that our protocol achieves higher efficiency while maintaining reasonable communication overhead.

2. Related Work

Since Lamport's pioneering one-time password protocol in 1981 [1], authentication mechanisms have evolved significantly. The introduction of two-factor authentication combining passwords and smart cards improved resistance against direct password guessing [2,10]. However, these protocols remained vulnerable to stolen verifier and smart card loss attacks. To enhance security further, researchers introduced biometrics as a third factor, giving rise to multi-factor authentication (MFA) protocols [11–14].

A major limitation of early MFA protocols was their design for single-server environments, where users must re-register with each server individually an inconvenient and insecure approach when managing multiple accounts. To mitigate this, multi-server MFA protocols were developed [5–7], enabling a single registration to grant access to multiple services. Kumari et al. [8] proposed a protocol using elliptic curve cryptography (ECC) and biohashing for multi-server systems, but Feng et al. [15] later exposed its vulnerabilities, including susceptibility to man-in-the-middle attacks and lack of user anonymity. Barman et al. [16] proposed a remote authentication protocol using fuzzy commitment, but this protocol cannot prevent insider attacks, and the smart card revocation phase is insecure. Barman subsequently proposed an improved protocol [17] to address these issues. Unfortunately, the new protocol does not ensure user anonymity. Analyzing the vulnerabilities in Barman et al.'s protocol [17], Ali et al. [18] proposed an enhanced three-factor symmetric key authentication protocol for multi-server environments. However, this improved protocol exhibits excessive computational overhead and fails to achieve post-quantum security. Yoon et al. [4] introduced another ECC-based solution, but its reliance on an online RC led to increased communication overhead and vulnerability to single-point failures.

To overcome the limitations of an online RC, several protocols adopted an offline RC model, where the RC distributes long-term keys during registration, enabling mutual authentication

between users and servers without involving the RC in every session. Chuang et al. [9] proposed a lightweight anonymous multi-server MFA protocol using an offline RC and employing only nonce and hash functions. Mishra et al. [19] demonstrated that Chuang's protocol [9] is unable to defend against user impersonation, denial of service (DoS) attacks and server spoofing attacks. Then, they proposed an improved protocol that overcomes the weaknesses of the original protocol while retaining its advantages. Chaturvedi et al. [20] proposed a biometric-based protocol that utilizes biohashing, modular exponentiation, and hash functions, along with an offline RC mechanism. Unfortunately, Chaturvedi's protocol [20] fails to offer user anonymity and smart card revocation capabilities. Luo et al. [21] and Shukla et al. [22] presented provably secure ECC-based protocols for multi-server settings with an offline RC; however, they remain vulnerable in the face of quantum threats.

As quantum computing advances threaten classical cryptographic algorithms, lattice-based cryptography has emerged as a promising alternative. Lattice-based problems such as Learning With Errors (LWE) and Ring-LWE offer strong hardness guarantees even under quantum attacks [23–26]. Several recent works have explored lattice-based authentication protocols. For instance, Bahache et al. [27] proposed a Kyber-based framework for cloud healthcare applications, while Ahmad et al. [28] and Basker et al. [29] introduced post-quantum MFA protocols for medical IoT systems. However, none of these protocols are suitable for multi-server environments with an offline RC.

3. Preliminaries

This section introduces definitions and concepts that may be utilized in this paper.

3.1. Notation

In this paper, we use bold uppercase letters to represent matrices and bold lowercase letters to represent vectors. The transpose of $\mathbf{a}$ is written as $\mathbf{a}^\top$. "PPT" is short for probabilistic polynomial time.

3.2. Module-LWE

Definition 1 (Binomial Distribution)**.** *The centered binomial distribution B_η for some integer η is defined as follows: Sample $\{(a_i, b_i)\}_{i=1}^{\eta} \leftarrow (\{0,1\}^2)^\eta$ and output $\sum_{i=1}^{\eta}(a_i - b_i)$.*

If v is an element of R, $v \leftarrow \beta_\eta$ means that $v \in R$ is generated from a distribution where each of its coefficients is generated according to β_η.

Definition 2 (Module-LWE)**.** *Let n be a positive integer. The Module-LWE hard problem is to distinguish uniform samples $(\mathbf{a}_i, b_i) \leftarrow R_q^n \times R_q$ from samples $(\mathbf{a}_i, b_i) \leftarrow R_q^n \times R_q$, where $\mathbf{a}_i \leftarrow R_q^n$ is uniform and $b_i = \mathbf{a}_i^\top \mathbf{s} + e_i$, with $\mathbf{s} \leftarrow \beta_\eta^n$ being common to all samples and $e_i \leftarrow \beta_\eta$ being fresh for every sample. More precisely, the advantage of an adversary $\mathcal{A}$ solving M-LWE problem is defined as*

$$\begin{aligned} Adv_{m,n,\eta}^{\mathsf{mlwe}}(\mathcal{A}) \quad = \quad & \left| \Pr\left[b' = 1 : \begin{array}{l} \mathbf{A} \leftarrow R_q^{m \times n}; (\mathbf{s}, \mathbf{e}) \leftarrow \beta_\eta^n \times \beta_\eta^m; \\ \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}; b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b}) \end{array} \right] \right. \\ & \left. - \Pr\left[b' = 1 : \mathbf{A} \leftarrow R_q^{m \times n}; \mathbf{b} \leftarrow R_q^m; b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b}) \right] \right|. \end{aligned}$$

The Module-LWE problem is hard if for all PPTs, adversary $\mathcal{A}$, $Adv_{m,n,\eta}^{\mathsf{mlwe}}(\mathcal{A})$ is negligible.

3.3. Key Encapsulation Mechanism

Definition 3 (Key Encapsulation Mechanism)**.** *A key encapsulation mechanism (KEM) consists of a triplet of probabilistic algorithms (KeyGen, Encaps, Decaps), a ciphertext space $\mathcal{C}$ and a KEM key space $\mathcal{K}$.*

- *KeyGen*(1^λ) *algorithm: upon input the security parameter* 1^λ*, outputs a public key pk and a secret key sk.*
- *Encaps*(pk) *algorithm: inputs pk and outputs a ciphertext c and a key* $K \in \mathcal{K}$*.*
- *Decaps*(sk, c) *algorithm: inputs sk and c and outputs either a key* $K \in \mathcal{K}$ *or a special symbol* $\perp$*, indicating failure.*

  We define a KEM as $(1-\delta)$*-correct if*

$$\Pr[(c, K) \leftarrow Encaps(pk) \wedge K = Decaps(sk, c)] \geq 1 - \delta,$$

where $(pk, sk) \leftarrow KeyGen(1^\lambda)$.

The standard security notion for the key encapsulation mechanism is indistinguishability under chosen ciphertext attacks. The advantage of an adversary $\mathcal{A}$ is defined as $Adv^{\text{cca}}_{\text{KEM}}(\mathcal{A}) = |\Pr[b = b' : (pk, sk) \leftarrow \text{KeyGen}(1^\lambda); b \leftarrow \{0,1\}; (c^*, K_0^*) \leftarrow \text{Encaps}(pk); K_1^* \leftarrow \mathcal{K}$; and $b' \leftarrow \mathcal{A}^{\text{DECAPS}(\cdot)}(pk, c^*, K_b^*)] - \frac{1}{2}|$, where the DECAPS oracle is defined as $\text{DECAPS}(\cdot) := \text{Decaps}(sk, \cdot)$. $\mathcal{A}$ is not allowed to query $\text{DECAPS}(\cdot)$ with the challenge ciphertext c^*.

Our protocol will use the Kyber KEM introduced by Bos [26] as a building block. Kyber KEM includes three algorithms, (Kyber.KenGen, Kyber.Encaps, and Kyber.Decaps). The security of Kyber KEM is based on the hardness of Module-LWE in the classical and quantum random oracle models [26].

3.4. Metric Spaces and Min-Entropy

Definition 4 (Metric Spaces)**.** *A metric space is a set* $\mathcal{M}$ *equipped with a distance function* $\text{dis} : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}^+ \cup \{0\}$*, which defines the distance between its elements (e.g., Hamming distance, which denotes the number of positions at which two strings of equal length differ).*

Definition 5 (Min-Entropy)**.** *For a random variable X, the min-entropy of X is defined as* $H_\infty(X) = -\log(\max_{x \in \mathcal{X}} \Pr[X = x])$.

3.5. Fuzzy Extractor

Biometric data are not uniformly distributed and may vary slightly during different samples. Fuzzy extractors enable the extraction of stable and nearly uniform randomness from noisy biometric data.

Definition 6 (Fuzzy Extractor)**.** *An* $(\mathcal{M}, m, \mathcal{R}, \mathtt{t}, \varepsilon)$*-fuzzy extractor (FE) consists of two PPT algorithms* $FE = (Gen, Rep)$ *with the following properties:*

- *Gen*(w)*: on input of an element* $w \in \mathcal{M}$*, it outputs a public helper string P and an extracted string* $R \in \mathcal{R}$*.*
- *Rep*(w', P)*: on input of an element* $w' \in \mathcal{M}$ *and the public helper string P, it outputs an extracted string R or* $\perp$*.*
- *Correctness. If* $dis(w, w') \leq \mathtt{t}$*, then for all* $(R, P) \leftarrow Gen(w)$*, we have* $R = Rep(w, P)$*.*
- (m, l, ε)*-Security. For any distribution W over* $\mathcal{M}$ *such that* $H_\infty(W) \geq m$*, for any adversary* $\mathcal{A}$*, it follows that*

$$Adv_{FE}(\mathcal{A}) = |\Pr[\mathcal{A}(R, P) \Rightarrow 1] - \Pr[\mathcal{A}(U, P) \Rightarrow 1]| \leq \varepsilon,$$

  where $(R, P) \leftarrow Gen(w)$ *and* $U \leftarrow \{0,1\}^l$.

Our protocol will use the fuzzy extractor proposed in [30], which is information-theoretically secure.

Remark 1. *The fuzzy extractor proposed by Dodis et al. [30] employs an efficient linear error-correcting code to handle errors in noisy biometric data. The encoding and decoding algorithms are efficient and publicly accessible. Furthermore, it has been proven to be information-theoretically secure [30].*

3.6. Collision-Resistant Hash Function

A collision-resistant hash function is a mathematical function that takes an arbitrary-length message M in space $\mathcal{M}$ as input and produces a fixed-length string $H(M)$. The collision-resistant hash function has the following property: it is computationally infeasible to find two distinct messages M and M' such that $H(M) = H(M')$.

3.7. Adversary Model

We assume that the Dolev–Yao (DY) threat model [31] is applicable in our scheme as the standard threat model. Under this model, the adversary $\mathcal{A}$ has the following capabilities:

- $\mathcal{A}$ is familiar with the protocol's operation and public parameters. Any legitimate user can pretend to be $\mathcal{A}$, and vice versa.
- $\mathcal{A}$ can eavesdrop, replay, modify, and forge messages communicated over the insecure public channel.
- $\mathcal{A}$ can attempt offline password guessing attacks using information extracted from smart cards or other sources.
- $\mathcal{A}$ cannot simultaneously compromise a legitimate user's smart card, biometric and password. However, $\mathcal{A}$ can compromise any two of three factors and attempt to guess the third factor.
- Through side-channel attacks, such as power analysis, $\mathcal{A}$ can extract sensitive information stored on a smart card.

4. Proposed Protocol

In this section, we propose a multi-factor authentication protocol suitable for multi-server architectures, which utilizes an offline RC. The proposed protocol involves three entities: the registration center RC, the user U_i and the service server S_j. The protocol comprises four phases: initializaiton phase, registration phase, login and authentication phase and update phase. All the notations in this section are summarized in Table 1.

Table 1. Notations' summary.

Notation	Description
RC	Registration Center
U_i	ith user
UID_i, PW_i, BIO_i	Identity, password, and biometric of U_i
S_j	jth server
SID_j	Identity of S_j
K	Private key of RC
sk_{S_j}, pk_{S_j}	Private key, public key of the Kyber KEM for S_j
sk_{U_i}, pk_{U_i}	Private key, public key of the Kyber KEM for U_i
SK	Session key between S_j and U_i
T_1, T_2, T_3	Current time stamps
$h(\cdot)$	Hash function
$\oplus$, $\|$	Bitwise XOR, concatenation
ΔT	Acceptable transmission delay threshold
SC_i	Smart card of U_i
$\mathcal{A}$	Adversary

4.1. Initializaiton Phase

In this phase, the RC selects and publicly discloses certain information as follows:

1. *RC* selects a hash function h: $\{0,1\}^* \rightarrow \{0,1\}^n$.
2. *RC* selects the *Gen* and *Rep* algorithms for the fuzzy extractor.
3. *RC* selects security parameters for the Kyber KEM.
4. *RC* randomly generates a private key K, publishes $\{$Kyber.KeyGen, Kyber.Encaps, Kyber.Decaps, $h, Gen, Rep\}$ and securely stores K.

4.2. Registration Phase

In this phase, service servers and users register with the registration center (*RC*) via a secure channel.

4.2.1. Server Registration

Assuming there are k initial servers ready to be registered before any user's registration, additional k' server slots are reserved to accommodate potential new servers in the future. As depicted in Table 2, the detailed steps of the server registration are outlined below.

Table 2. Server registration.

Server (S_j)		***RC***
		Select k' unique SID_j
Select SID_j Compute $(pk_{S_j}, sk_{S_j}) \leftarrow$ Kyber.KeyGen(1^λ) Store sk_{S_j}		
	$\xrightarrow[\text{secure channel}]{\{SID_j, pk_{S_j}\}}$	
		Check SID_j Compute $PSR_j = h(SID_j \| \| K)$ Publish $\{SID_j, pk_{S_j}\}$
	$\xleftarrow[\text{secure channel}]{\{SID_j, PSR_j\}}$	
		Store PSR_j
Compare SID_j Store PSR_j		

1. *RC* first assumes that k initial servers will complete their registration before any user's registration, and additional k' servers will complete their registration thereafter. Then, *RC* selects k' unique SID'_j for servers that may register in the future and computes $PSR'_j = H(SID'_j||K)$.
2. If a server $S_j (1 \leq j \leq k + k')$ intends to register, it first chooses its unique ID, denoted as SID_j. Then, the server invokes $(pk_{S_j}, sk_{S_j}) \leftarrow$ Kyber.KeyGen(1^λ), stores sk_{S_j} securely, and sends pk_{S_j} along with SID_j to *RC*.
3. Upon receiving the message $\{SID_j, pk_{S_j}\}$ from S_j, *RC* will do the following:
   - If no user has registered, *RC* will start to check the SID_j. If the SID_j has been occupied, *RC* will returns a rejection message. Otherwise, *RC* computes $PSR_j = h(SID_j||K)$ and publishes $\{SID_j, pk_{S_j}\}$ as the public information of server S_j and securely stores PSR_j. Finally, *RC* transmits $\{SID_j, PSR_j\}$ to the server S_j.
   - If a user has completed their registration, the *RC* will disregard the SID_j received from the server. Instead, *RC* will use its pre-selected SID'_j as the server's ID, setting $SID_j = SID'_j, PSR_j = PSR'_j$. Then *RC* publishes $\{SID_j, pk_{S_j}\}$ as the public information of server S_j and securely stores PSR_j. Finally, *RC* sends $\{SID_j, PSR_j\}$ to server S_j.

4. After receiving $\{SID_j, PSR_j\}$ from RC, server S_j will use SID_j as its own ID and securely store PSR_j for future authentication purposes.

4.2.2. User Registration

The user's information includes their ID, password and biometric data. Since biometric data are immutable, directly storing them for authentication purposes poses significant security risks in the event of a data leak. Additionally, biometric data can show slight variations each time they are captured. To address these issues, our protocol uses a fuzzy extractor to handle minor variations while preventing raw biometric data from being stored. During the user registration phase, the user's biometric data BIO_i are input into the fuzzy extractor. This process generates two outputs: a random string R and a public helper string P. Only the public helper string P needs to be stored by the user. As shown in Table 3, the detailed steps of the user registration phase are outlined below.

Table 3. User registration.

User (U_i)		**RC**
Select UID_i, PW_i Imprint BIO_i $(R_i, P_i) \leftarrow Gen(BIO_i)$ Generate a random nonce a $PWU_i = h(UID_i \| \| PW_i \| \| a)$ $AID_i = h(UID_i \| \| R_i)$ $A_i = AID_i \oplus a$	$\xrightarrow[\text{secure channel}]{\{AID_i, PWU_i\}}$	
		Check whether the AID_i is unique if unique, store AID_i in the database for $1 \leq j \leq k + k'$, compute $V_{ij} = h(AID_i \| \| PSR_j) \oplus PWU_i$ $P_{ij} = h(SID_j \| \| PSR_j) \oplus PWU_i$ store (SID_j, V_{ij}, P_{ij}) in a smart card SC_i
	$\xleftarrow[\text{secure channel}]{SC_i = \{(SID_j, V_{ij}, P_{ij}) \mid 1 \leq j \leq k+k'\}}$	
Store P_i, PWU_i, A_i in SC_i		

1. User U_i selects its own user ID UID_i and password PW_i. Then, U_i inputs biometric data BIO_i and invokes the generation algorithm $(R_i, P_i) \leftarrow Gen(BIO_i)$. U_i generates a random nonce a and computes $PWU_i = h(UID_i \| \| PW_i \| \| a)$ and $AID_i = h(UID_i \| \| R_i)$, $A_i = AID_i \oplus a$. Finally, U_i sends $\{AID_i, PWU_i\}$ to RC.
2. When RC receives the registration request from the user, it first verifies whether the AID_i is unique. If the AID_i is already in use, RC returns a rejection message. Otherwise, RC stores AID_i in its database for future use. For $1 \leq j \leq k + k'$, RC computes $V_{ij} = h(AID_i \| \| PSR_j) \oplus PWU_i$ and $P_{ij} = h(SID_j \| \| PSR_j) \oplus PWU_i$ and stores (SID_j, V_{ij}, P_{ij}) in a smart card SC_i. Finally, it securely delivers SC_i to the user.
3. After receiving the smart card SC_i, user U_i stores PWU_i, P_i and A_i in the smart card SC_i. Ultimately, the smart card contains $\{PWU_i, P_i, A_i \{SID_j, V_{ij}, P_{ij} | 1 \leq j \leq k + k'\}\}$.

4.3. Login and Authentication Phase

This phase encompasses two components: user login and mutual authentication. As depicted in Table 4, the detailed steps for establishing a session key are outlined below.

4.3.1. User Login

All users are required to pass the login verification process to access a server S_j.

1. User U_i inserts the smart card SC_i to a smart card scanner and inputs UID'_i, PW'_i and biometric data BIO'_i.
2. SC_i invokes the Rep algorithm to reproduce $R'_i = Rep(BIO'_i, P_i)$ and computes $AID'_i = h(UID'_i||R'_i)$, $a' = A_i \oplus AID'_i$, $PWU'_i = h(UID'_i||PW'_i||a')$. Then, SC_i verifies whether $PWU'_i = PWU_i$. If it is true, SC_i continues the process. Otherwise, it rejects the login request.

Table 4. Login and authentication.

User (U_i)		**Server (S_j)**
U_i inserts SC_i inputs UID'_i, PW'_i, BIO'_i Compute $R'_i = Rep(BIO'_i, P_i)$ $AID'_i = h(UID'_i\|\|R'_i)$ $a' = A_i \oplus AID'_i$ $PWU'_i = h(UID'_i\|\|PW'_i\|\|a')$ If $PWU_i \neq PWU'_i$ SC_i ends the session Else continue Compute $(c_1, K_1) \leftarrow$ Kyber.Encaps(pk_{S_j}) $(pk_{U_i}, sk_{U_i}) \leftarrow$ Kyber.KeyGen(1^λ) $AID'_i = h(UID'_i\|\|R'_i)$ $M_1 = PWU'_i \oplus V_{ij}$ $M_2 = PWU'_i \oplus P_{ij}$ $M_3 = AID'_i \oplus M_2 \oplus K_1$ $M_4 = h(M_1\|\|K_1\|\|T_1)$	$\xrightarrow[public\ channel]{Msg1=\{c_1, M_3, M_4, T_1, pk_{U_i}\}}$	
		If $T'_1 - T_1 \leq \Delta T$, continue Compute $K'_1 =$ Kyber.Decaps(sk_{S_j}, c_1) $M_5 = h(SID_j\|\|PSR_j)$ $M_6 = M_5 \oplus M_3 \oplus K'_1$ $M_7 = h(M_6\|\|PSR_j)$ $M_8 = h(M_7\|\|K'_1\|\|T_1)$ If $M_8 = M_4$, continue Compute $(c_2, K_2) \leftarrow$ Kyber.Encaps(pk_{U_i}) $SK_{ij} = h(M_5\|\|M_7\|\|K'_1\|\|K_2\|\|T_2)$ $M_9 = h(SK_{ij}\|\|M_7\|\|T_2)$
	$\xleftarrow[public\ channel]{Msg2=\{M_9, T_2, c_2\}}$	
If $T'_2 - T_2 \leq \Delta T$, continue Compute $K'_2 =$ Kyber.Decaps(sk_{U_i}, c_1) $SK'_{ij} = h(M_2\|\|M_1\|\|K_1\|\|K'_2\|\|T_2)$ $M_{10} = h(SK'_{ij}\|\|M_1\|\|T_2)$ If $M_{10} = M_9$, continue Compute $M_{11} = h(SK'_{ij}\|\|M_1\|\|K'_2\|\|T_3)$		
	$\xrightarrow[public\ channel]{Msg3=\{M_{11}, T_3\}}$	
		If $T'_3 - T_3 \leq \Delta T$, continue. Compute $M_{12} = h(SK_{ij}\|\|M_7\|\|K_2\|\|T_3)$ If $M_{12} = M_{11}$, the session key SK_{ij} is established.

4.3.2. Mutual Authentication

After the user U_i successfully logs in, U_i selects the server S_j they wish to access and sends the initial message. Upon receiving the initial message, the server S_j verifies the user's identity. Then, the server S_j sends a response message to the user U_i. User U_i uses the response message to authenticate the server's identity. Following this mutual authentication process, both parties will collaboratively establish a session key for use in their subsequent communications. All messages are transmitted over a public channel. The detailed mutual authentication and key establishment processes are outlined below.

1. SC_i invokes $(c_1, K_1) \leftarrow \text{Kyber.Encaps}(pk_{S_j})$ and $(pk_{U_i}, sk_{U_i}) \leftarrow \text{Kyber.KeyGen}(1^\lambda)$. Then, SC_i generates a time stamp T_1 and computes the following messages:

$$\begin{gathered} AID_i' = h(UID_i' || R_i'); \\ M_1 = PWU_i' \oplus V_{ij}; \\ M_2 = PWU_i' \oplus P_{ij}; \\ M_3 = AID_i' \oplus M_2 \oplus K_1; \\ M_4 = h(M_1 || K_1 || T_1); \end{gathered}$$

   then, it transmits $Msg1 = \{c_1, M_3, M_4, T_1, pk_{U_i}\}$ to S_j.
2. Upon receiving message $Msg1$, the server S_j first records the current timestamp T_1' and checks if $T_1' - T_1 \leq \Delta T$. If it does not hold, S_j ends the session. Otherwise, S_j invokes $K_1' \leftarrow \text{Kyber.Decaps}(c_1, sk_{S_j})$. Then, S_j uses the PSR_j stored in the registration phase to compute

$$\begin{gathered} M_5 = h(SID_j || PSR_j), \\ M_6 = M_5 \oplus M_3 \oplus K_1', \\ M_7 = h(M_6 || PSR_j), \\ M_8 = h(M_7 || K_1' || T_1). \end{gathered}$$

   If $M_8 = M_4$ holds, S_j invokes the $(c_2, K_2) \leftarrow \text{Kyber.Encaps}(pk_{U_i})$. Then, S_j generates the time stamp T_2 and computes

$$\begin{gathered} SK_{ij} = h(M_5 || M_7 || K_1' || K_2 || T_2), \\ M_9 = h(SK_{ij} || M_7 || T_2). \end{gathered}$$

   Finally, server S_j transmits $Msg2 = \{M_9, T_2, c_2\}$ to user U_i.
3. Upon receiving message $Msg2$, SC_i generates the current time stamp T_2' and then checks if $T_2' - T_2 \leq \Delta T$. If $T_2' - T_2 \leq \Delta T$, SC_i computes

$$\begin{gathered} K_2' = \text{Kyber.Decaps}(sk_{U_i}, c_1), \\ SK_{ij}' = h(M_2 || M_1 || K_1 || K_2' || T_2), \\ M_{10} = h(SK_{ij}' || M_1 || T_2). \end{gathered}$$

   If $M_{10} = M_9$ holds, SC_i generates the time stamp T_3, computes $M_{11} = h(SK_{ij}' || M_1 || K_2' || T_3)$ and transmits $Msg3 = \{M_{11}, T_3\}$ to server S_j.
4. Upon receiving the message $Msg3$, server S_j records the current timestamp T_3' and checks if $T_3' - T_3 \leq \Delta T$. If it does not hold, S_j ends the session. Otherwise, S_j computes $M_{12} = h(SK_{ij} || M_7 || K_2 || T_3)$. If $M_{12} = M_{11}$, S_j acknowledges the correctness of the session key. The session key SK_{ij} is established between U_i and S_j for subsequent communication.

4.4. Update Phase

4.4.1. Password Update Phase

In this phase, users can update their passwords without further contacting the *RC*. Before updating the password, the user must complete the login process. The detailed steps of the password update are outlined below.

1. User U_i inserts the smart card SC_i to a smart card scanner and inputs UID_i', PW_i' and biometric data BIO_i'.
2. SC_i invokes the Rep algorithm to reproduce $R_i' = Rep(BIO_i', P_i)$ and computes $AID_i' = h(UID_i'||R_i')$, $a' = A_i \oplus AID_i'$ and $PWU_i' = h(UID_i'||PW_i'||a')$. Then, SC_i verifies whether $PWU_i' = PWU_i$. If it is true, SC_i continues the process. Otherwise, it rejects the login request.
3. U_i provides a new password PW_i^{new}.
4. SC_i computes

$$PWU_i^{new} = h(UID_i||PW_i^{new}||a').$$

For $1 \leq j \leq k + k'$, SC_i computes

$$V_{ij}^{new} = V_{ij} \oplus PWU_i \oplus PWU_i^{new},$$
$$P_{ij}^{new} = P_{ij} \oplus PWU_i \oplus PWU_i^{new}.$$

5. SC_i updates information as $\{PWU_i^{new}, P_i, A_i, \{SID_j, V_{ij}^{new}, P_{ij}^{new} | 1 \leq j \leq k + k'\}\}$.

4.4.2. Smart Card Revocation Phase

In this phase, the user can revoke the smart card by securely communicating with the RC.

1. User U_i inserts the old smart card, selects new UID_i^{new}, PW_i^{new}, a^{new}, inputs biometric data BIO_i' and computes $R_i' = Rep(BIO_i', P_i)$. Then, U_i computes $AID_i' = h(UID_i'||R_i')$, $PWU_i^{new} = h(UID_i^{new}||PW_i^{new}||a^{new})$, $AID_i^{new} = h(UID_i^{new}||R')$ and $A_i^{new} = AID_i^{new} \oplus a^{new}$, and then transmits $\{AID_i', AID_i^{new}, PWU_i^{new}\}$ to RC along with a revocation request.
2. RC verifies the legitimacy of U_i by comparing AID_i' with all the AIDs stored in the database.
3. RC replaces the old AID_i with the AID_i^{new}. Then, RC computes $V_{ij}^{new} = h(AID_i^{new}||PSR_j) \oplus PWU_i^{new}$ and $P_{ij}^{new} = h(SID_j||PSR_j) \oplus PWU_i^{new}$ for $1 \leq j \leq k + k'$.
4. For $1 \leq j \leq k + k'$, RC stores all the $SID_j, V_{ij}^{new}, P_{ij}^{new}$ in a smart card SC_i^{new} and delivers SC_i^{new} to the user through a secure channel.
5. U_i stores PWU_i^{new}, A_i^{new} and P_i in SC_i^{new}.

4.5. Formal Security Analysis

This section presents the formal security analysis of our proposed MFA using the real or random (ROR) model.

4.5.1. ROR Model

The following provides the main components of the ROR model:

- Participants: U_i^a and S_j^b represent the ath instance of user U_i and the bth instance of server S_j, respectively.
- Adversary: $\mathcal{A}$ denotes a probabilistic polynomial time adversary.

The adversary $\mathcal{A}$ is allowed to query the following oracles:

- $Execute(U_i^a, S_j^b)$: This oracle outputs the protocol messages of an honest protocol execution between U_i^a and S_j^b. By querying this oracle, the adversary launches passive attacks.
- $Send(U_i^a / S_j^b, m)$: This query simulates an active attack by the adversary $\mathcal{A}$, where $\mathcal{A}$ can send message m to the instance U_i^a or S_j^b and receive the response message from U_i^a or S_j^b.

- $Corrupt(U_i, d)$: By querying this oracle, the adversary $\mathcal{A}$ can obtain one or two types of U_i's secret information.
  $d = 0$: $\mathcal{A}$ compromises U_i's UID_i and PW_i.
  $d = 1$: $\mathcal{A}$ compromises all the information stored in U_i's smart card.
  $d = 2$: $\mathcal{A}$ compromises U_i's BIO_i.
- $Test(U_i^a / S_j^b)$: If SK_{ij} is established, the oracle returns SK_{ij} if $c = 1$, and it returns a uniformly random key if $c = 0$, where c is uniformly chosen from $\{0, 1\}$.
- Random Oracle: We model the collision-resistant hash function $h(\cdot)$ as a random oracle, denoted by H, which is publicly available to all participants, including the adversary $\mathcal{A}$.

Definition 7 (Semantic Security). *A PPT adversary $\mathcal{A}$ is allowed to query $Execute(U_i^a, S_j^b)$, $Send(U_i^a / S_j^b, m)$, $Corrupt(U_i^a, d)$, $Test(U_i^a / S_j^b)$ and a random oracle. Finally, $\mathcal{A}$ provides a guessed bit c'. Given a protocol $\mathcal{P}$, the advantage of $\mathcal{A}$ in the polynomial time t is defined as $Adv_{\mathcal{P}}^{\mathcal{A}}(t) = |2 \cdot \Pr[c' = c] - 1|$. We say that $\mathcal{P}$ satisfies the semantic security if $Adv_{\mathcal{P}}^{\mathcal{A}}(t) = |2 \cdot \Pr[c' = c] - 1|$ is negligible.*

4.5.2. Security Proof

Theorem 1. *Suppose the PPT adversary $\mathcal{A}$ can execute at most q_{exe} Execute queries, q_{send} Send queries, q_{test} Test queries and q_{hash} Hash queries. Let l_h denote the bit length of the output of the hash function, l_b denote the length of the output R of fuzzy extractor, and l_K denote the bit length of private key K selected by RC. Let $|U|$ denote the size of user ID space and $|\mathcal{K}|$ denote the size of encapsulation key space of Kyber.KEM. If the PPT $\mathcal{A}$ attempts to break the proposed protocol $\mathcal{P}$ in the polynomial time t, we have*

$$\begin{aligned} Adv_{\mathcal{P}}^{\mathcal{A}}(t) \quad \leq \quad & 4q_{exe} \cdot Adv_{Kyber}^{cca}(\mathcal{A}) + \frac{q_{hash}^2}{2^{l_h}} + \frac{q_{exe}^2}{|\mathcal{K}|} + \frac{q_{send}}{2^{l_b - 1}} \\ & + 2q_{send} \cdot Adv_{FE}(\mathcal{A}) + \frac{q_{send}}{2^{l_k - 1}} + \frac{2q_{send}}{|U|} + \frac{q_{send}}{2^{l_h - 2}}. \end{aligned}$$

Proof. Our proof consists of a sequence of hybrid games, starting with the real attack against MFA and ending in a game in which the adversary's advantage is 0. Let $\Pr[Succ_i]$ denote the probability that $\mathcal{A}$ successfully guesses the hidden bit c involved in Game_i.

- Game_0: This game corresponds to the real attack. According to the definition of semantic security, we have

$$Adv_{\mathcal{P}}^{\mathcal{A}}(t) = 2 \Pr[Succ_0] - 1. \tag{1}$$

- Game_1: This game is identical to Game_0, except that the simulator modifies the simulation of the $Execute$ query by replacing the key K_1 and K_2 used in SK_{ij} with random strings K_1^R and K_2^R of the same length. Recall that in our protocol, $SK_{ij} = h(M_5||M_7||K_1||K_2||T_2)$, where $(c_1, K_1) \leftarrow \text{Kyber.Encaps}(pk_{S_j})$ and $(c_2, K_2) \leftarrow \text{Kyber. Encaps}(pk_{U_i})$. In this game, $SK_{ij} = h(M_5||M_7||K_1^R||K_2^R||T_2)$. Messages that $\mathcal{A}$ can obtain from a $Execute$ query are $Msg1 = \{c_1, M_3, M_4, T_1, pk_{U_i}\}$, $Msg2 = \{M_9, T_2, c_2\}$, $Msg3 = \{M_{11}, T_3\}$. By the security of Kyber.KEM, with ciphertext c_1, c_2 and public key pk_{U_i}, pk_{S_j}, the probability that $\mathcal{A}$ can distinguish K_1 and K_2 from K_1^R and K_2^R is smaller than $2Adv_{\text{Kyber}}^{\text{cca}}(\mathcal{A})$. Since $\mathcal{A}$ can make at most q_{exe} $Execute$ queries, we have

$$|\Pr[Succ_1] - \Pr[Succ_0]| \leq 2q_{exe} \cdot Adv_{\text{Kyber}}^{\text{cca}}(\mathcal{A}). \tag{2}$$

- Game$_2$: In this game, the simulator will stop if a collision exists in hash queries or a collision exists in K_1^R or K_2^R. Based on the birthday paradox, the maximum probability of a collision in the output of a hash function is $\frac{q_{hash}^2}{2^{l_h+1}}$. The probability of finding a collision in K_1^R or K_2^R through Execute queries is at most $\frac{q_{exe}^2}{2|\mathcal{K}|}$. Thus, we have

$$|\Pr[Succ_2] - \Pr[Succ_1]| \leq \frac{q_{hash}^2}{2^{l_h+1}} + \frac{q_{exe}^2}{2|\mathcal{K}|}. \tag{3}$$

- Game$_3$: In this game, the simulator will reject all the *Send* queries.

We claim that in Game$_2$, the *Send* query sent by $\mathcal{A}$ will be rejected with overwhelming probability.

- Upon receiving the *Send* query $Send(S_j, Msg1)$, the simulator will check if $T_1' - T_1 \leq \Delta T$. If $T_1' - T_1 > \Delta T$, the simulator will reject the *Send* query; otherwise, the simulator will compute $K_1' = \text{Kyber.Decaps}(sk_{S_j}, c_1)$, $M_5 = h(SID_j||PSR_j)$, $M_6 = M_5 \oplus M_3 \oplus K_1'$, $M_7 = h(M_6||PSR_j)$, $M_8 = h(M_7||K_1'||T_1)$, and it will verify if $M_8 = M_4$. If $M_8 \neq M_4$, the simulator will reject the *Send* query.
  - Suppose that $\mathcal{A}$ has already executed $Corrupt(U_i, 0)$ and $Corrupt(U_i, 1)$ before executing the *Send* query. In this case, $\mathcal{A}$ obtains the smart card SC_i, password PW_i and user identity UID_i and does not know the biometric date BIO_i. In order to pass the verification, $\mathcal{A}$ needs to compute $AID_i = h(UID_i||R_i)$ where $(R_i, P_i) \leftarrow Gen(BIO_i)$. Note that as long as biometric data BIO_i has enough entropy, R_i is almost uniform. More precisely, for all adversary $\mathcal{A}$, the probability of $\mathcal{A}$ to distinguish R_i from a random string R is smaller than $Adv_{FE}(\mathcal{A})$. As a result, the probability of $\mathcal{A}$ guessing R_i is smaller than $\frac{q_{send}}{2^{l_b}} + q_{send} \cdot Adv_{FE}(\mathcal{A})$.
  - Suppose that $\mathcal{A}$ has already executed $Corrupt(U_i, 0)$ and $Corrupt(U_i, 2)$ before executing the *Send* query. In this case, $\mathcal{A}$ obtains the biometric data BIO_i, password PW_i and UID_i, but $\mathcal{A}$ does not obtain the smart card SC_i. In order to pass the verification, $\mathcal{A}$ needs to compute $AID_i = h(UID_i||R_i)$ and PSR_j, and then it computes $h(AID_i||PSR_j)$ and $h(SID_j||PSR_j)$. Recall that $PSR_j = h(SID_j||K)$ and K is the secret key of RC. If $\mathcal{A}$ can correctly guess the key of RC and P_i simultaneously, $\mathcal{A}$ can pass the verification. Since the key of RC is uniformly random, the probability of $\mathcal{A}$ guessing the key of RC and P_i simultaneously is less than $\frac{q_{send}}{2^{l_k}}$.
  - Suppose that $\mathcal{A}$ has already executed $Corrupt(U_i, 1)$ and $Corrupt(U_i, 2)$ before executing the *Send* query. In this case, $\mathcal{A}$ obtains the smart card SC_i and the biometric data BIO_i, but $\mathcal{A}$ does not obtain the password PW_i and UID_i. With the smart card, $\mathcal{A}$ only needs to compute AID_i to pass the verification. If $\mathcal{A}$ can correctly guess the UID_i, $\mathcal{A}$ would be able to pass the verification. Recall that $PWU_i = h(UID_i||PW_i||a)$, $AID_i = h(UID_i||R_i)$ and $A_i = AID_i \oplus a$, where a is a random nonce. Since a is random, AID_i is masked by a. As a result, $\mathcal{A}$ cannot launch the offline user ID guessing attack. Therefore, $\mathcal{A}$ can guess UID_i with probability $\frac{q_{send}}{|U|}$.
- Upon receiving the $Send(U_i, Msg2)$, the simulator checks if $T_2' - T_2 \leq \Delta T$. Then, the simulator computes $K_2' = \text{Kyber.Decaps}(sk_{U_i}, c_1)$, $SK_{ij}' = h(M_2||M_1||K_1||K_2'||T_2)$, $M_{10} = h(SK_{ij}'||M_1||T_2)$, and verifies if $M_{10} = M_9$. If $\mathcal{A}$ wants to pass the verification, $\mathcal{A}$ needs to compute M_{10}. Since the adversary $\mathcal{A}$ does not know the secret key sk_{S_j} and PSR_j, it is hard for $\mathcal{A}$ to compute M_2, M_1 and K_1. As a result, $\mathcal{A}$ can guess M_{10} with a probability no larger than $\frac{q_{send}}{2^{l_h}}$.

- Upon receiving $Send(S_j, Msg3)$, the simulator checks if $T_3' - T_3 \leq \Delta T$. Then, the simulator computes $M_{12} = h(SK_{ij}||M_7||K_2||T_3)$ and verifies if $M_{12} = M_{11}$. Since $\mathcal{A}$ does not know sk_{U_i}, it cannot decapsulate the ciphertext to obtain K_2. As a result, $\mathcal{A}$ can guess the M_{11} with a probability no larger than $\frac{q_{send}}{2^{l_h}}$.

From the above analysis, we can see that in Game_2, the *Send* query sent by $\mathcal{A}$ would not be rejected with a probability smaller than

$$\frac{q_{send}}{2^{l_b}} + q_{send} \cdot Adv_{FE}(\mathcal{A}) + \frac{q_{send}}{2^{l_k}} + \frac{q_{send}}{|U|} + \frac{q_{send}}{2^{l_h}} + \frac{q_{send}}{2^{l_h}}.$$

As a result,

$$|\Pr[Succ_3] - \Pr[Succ_2]| \leq \frac{q_{send}}{2^{l_b}} + q_{send} \cdot Adv_{FE}(\mathcal{A}) + \frac{q_{send}}{2^{l_k}} + \frac{q_{send}}{|U|} + \frac{q_{send}}{2^{l_h-1}}. \quad (4)$$

Note that in Game_3, all *Send* queries will be rejected. The session key SK_{ij} in the *Execute* query is generated from random strings K_1^R and K_2^R. Due to the security of the hash function, SK_{ij} is uniformly random in the view of $\mathcal{A}$. As a result, we have

$$\Pr[Succ_3] = \frac{1}{2}. \quad (5)$$

Combing Equations (1)–(5) together, we have

$$\begin{aligned} Adv_{\mathcal{P}}^{\mathcal{A}}(t) \quad \leq \quad & 4q_{exe} \cdot Adv_{\text{Kyber}}^{\text{cca}}(\mathcal{A}) + \frac{q_{hash}^2}{2^{l_h}} + \frac{q_{exe}^2}{|\mathcal{K}|} + \frac{q_{send}}{2^{l_b-1}} \\ & + 2q_{send} \cdot Adv_{FE}(\mathcal{A}) + \frac{q_{send}}{2^{l_k-1}} + \frac{2q_{send}}{|U|} + \frac{q_{send}}{2^{l_h-2}}. \end{aligned}$$

□

4.6. Informal Security Analysis

- **user anonymity:** To preserve user anonymity, in the registration phase, the user U_i computes masked identity $AID_i = h(UID_i||R_i)$, where R_i is generated by $Gen(BIO_i) = (R_i, P_i)$. In the authentication phase, U_i sends $Msg1 = \{c_1, M_3, M_4, T_1, pk_{U_i}\}$ to the server S_j; the masked identity AID_i is involved in $Msg1$. Note that the message related to UID_i transmitted to the *RC* and server S_j is the masked identity AID_i. This ensures that the actual user identity remains protected. More precisely, due to the security of fuzzy extractor, as long as the biometric data BIO_i have enough entropy, R_i is almost uniformly distributed. Given the security properties of the hash function, it is infeasible for the *RC*, server S_j, or adversary $\mathcal{A}$ to guess UID_i. This ensures user anonymity with respect to *RC*, server S_j and adversary $\mathcal{A}$.
- **user untraceability:** The user untraceability ensures that it is difficult for an adversary $\mathcal{A}$ to trace or link different sessions to the same user by eavesdropping. Recall that, in each session, the messages transmitted over the public channel are $Msg1 = \{c_1, M_3, M_4, T_1, pk_{U_i}\}$, $Msg2 = \{M_9, T_2, c_2\}$ and $Msg3 = \{M_{11}, T_3\}$. Note that, in each session, user U_i invokes $(pk_{U_i}, sk_{U_i}) \leftarrow \text{Kyber.KeyGen}(1^\lambda)$ and $(c_1, K_1) \leftarrow \text{Kyber.Encaps}(pk_{S_j})$ and sets $M_3 = AID_i' \oplus M_2 \oplus K_1$ and $M_4 = h(M_1||K_1||T_1)$. As a result, each parameter $Msg1$ is randomized. Similarly, every parameter in both $Msg2$ and $Msg3$ is randomized. Thus, the absence of transmission of static parameters ensures user untraceability.
- **replay attack:** In the authentication phase, the message $Msg1 = \{c_1, M_3, M_4, T_1, pk_{U_i}\}$ sent by user U_i involves a random key K_1 and a timestamp T_1, and the message $Msg2 = \{M_9, T_2, c_2\}$ sent by the server S_j involves a random key K_2 and a timestamp

T_2. The involvement of random keys and timestamps helps our protocol resist replay attacks, specifically if an attacker eavesdrops $Msg1 = \{c_1, M_3, M_4, T_1, pk_{U_i}\}$ and performs a replay attack using $Msg1$. Upon receiving $Msg1$, S_j will check whether both $T_1' - T_1 \stackrel{?}{\leq} \Delta T$ and $M_8 \stackrel{?}{=} h(M_7||K_1'||T_{\mathcal{A}})$ are true. If not, it ends the session. By the security of the underlying hash function, random key K_1 and timestamp T_1, the server S_j will end the session with overwhelming probability. Similar analysis can be applied to $Msg2$.

- **privileged insider attack:** In our protocol, in the registration phase, user U_i transmits $AID_i = h(UID_i||R_i)$ and $PWU_i = h(UID_i||PW_i||a)$ to the RC. Note that, R_i is generated by $Gen(BIO_i) = (R_i, P_i)$ and a is a random nonce. As long as the biometric data BIO_i contain sufficient entropy, R_i is uniformly random. Additionally, the hash function has the property of pre-image resistance. Therefore, it is hard for an insider to derive $\{UID_i, PW_i, BIO_i\}$, even if the registration message $\{AID_i, PWU_i\}$ is leaked.
- **offline password guessing attack:** Suppose that the adversary has obtained the information $\{PWU_i, P_i, A_i, \{SID_j, V_{ij}, P_{ij}|1 \leq j \leq k + k'\}\}$ stored on the smart card and is attempting to guess the user's password PW_i offline. Note that $PWU_i = h(UID_i||PW_i||a)$ contains the information of PW_i. If an attacker attempts to guess PW_i from PWU_i, it is required to guess UID_i, PW_i and a simultaneously. Recall that $AID_i = h(UID_i||R_i)$, $A_i = AID_i \oplus a$, $Gen(BIO_i) = (R_i, P_i)$ and a is a random nonce. The fuzzy extractor guarantees that as long as BIO_i has enough entropy, R_i is almost uniformly distributed even if P_i is public. Due to the security of the hash function, AID_i is pseudorandom. a is pseudorandom-conditioned on A_i. Therefore, the adversary cannot obtain PW_i from PWU_i. Thereby, our protocol can resist offline password guessing attacks.
- **stolen smart card attack:** Suppose that the smart card is stolen by an adversary $\mathcal{A}$, who can extract all the information stored in the smart card. If $\mathcal{A}$ wants to log in, $\mathcal{A}$ needs to compute $PWU_i = h(UID_i, PW_i, a)$, where a is a random nonce. a is concealed by AID_i and $A_i = AID_i \oplus a$. Recall that $AID_i = h(UID_i||R_i)$ and $Gen(BIO_i) = (R_i, P_i)$. This means that if $\mathcal{A}$ wants to compute $PWU_i = h(UID_i, PW_i, a)$, $\mathcal{A}$ needs to guess UID_i, PW_i and BIO_i simultaneously. Since BIO_i has enough entropy, it is hard for $\mathcal{A}$ to guess BIO_i. As a result, it is difficult for $\mathcal{A}$ to log in just with a smart card.
- **user impersonation attack:** A user impersonation attack refers to when an attacker pretends to be a legitimate user to illegally obtain information services from servers. Impersonating a legitimate user U_i, the attacker $\mathcal{A}$ must be able to forge a valid login request. In our protocol, this means that $\mathcal{A}$ needs to successfully compute $AID_i = h(UID_i||R_i)$ where $(R_i, P_i) \leftarrow Gen(BIO_i)$.
  - Through eavesdropping, $\mathcal{A}$ can easily obtain $\{Msg1, Msg2, Msg3\}$. In our protocol, AID_i is masked by K_1. By the security of Kyber.KEM, K_1 is pseudorandom. As a result, $\mathcal{A}$ cannot obtain AID_i from eavesdropping.
  - Assume that $\mathcal{A}$ obtains the smart card SC_i and biometric data BIO_i. Recall that $PWU_i = h(UID_i||PW_i||a)$, $AID_i = h(UID_i||R_i)$ and $A_i = AID_i \oplus a$, where a is a random nonce. Since a is random, AID_i is masked by a. As a result, $\mathcal{A}$ cannot launch a user ID guessing attack. Therefore, $\mathcal{A}$ cannot compute $AID_i = h(UID_i||R_i)$.
  - Assume that $\mathcal{A}$ obtains the smart card SC_i and password PW_i and user identity UID_i. As long as biometric data BIO_i have enough entropy, R_i is almost uniform. As a result, $\mathcal{A}$ cannot compute $AID_i = h(UID_i||R_i)$.
  - Assume that $\mathcal{A}$ obtains the biometric data BIO_i and password PW_i and UID_i. If $\mathcal{A}$ wants to compute $AID_i = h(UID_i||R_i)$, $\mathcal{A}$ needs to invoke $R_i = Rep(BIO_i, P_i)$ to

obtain R_i. Since P_i is stored in the smart card, $\mathcal{A}$ cannot invoke $R_i = Rep(BIO_i, P_i)$. As a result, $\mathcal{A}$ cannot compute $AID_i = h(UID_i||R_i)$.

- **man-in-the-middle attack:** The adversary $\mathcal{A}$ pretends to be the server in a conversation with the user and also pretends to be the user in a conversation with the server.
  - Every legal server has its own sk_{S_j}. Without the sk_{S_j}, $\mathcal{A}$ cannot compute $K_1' = \text{Kyber.Decaps}(sk_{S_j}, c_1)$ and forge M_9. Therefore, $\mathcal{A}$ cannot make U_i believe that $\mathcal{A}$ is the specific legitimate server.
  - Every legal user has her/his own UID_i, PW_i, BIO_i and SC_i. As long as the adversary $\mathcal{A}$ does not obtain all three factors of the user U_i who they want to impersonate, $\mathcal{A}$ cannot compute $AID_i = h(UID_i||R_i)$. As a result, verification of the server would not be passed.
- **backward and forward secrecy:** In our protocol, even if a participant's long-term secret is compromised, it remains difficult for an adversary $\mathcal{A}$ to derive either previously generated or future possible keys. Note that in each session, the user U_i will invoke $(pk_{U_i}, sk_{U_i}) \leftarrow \text{Kyber.KeyGen}(1^\lambda)$ and send pk_{U_i} to the server S_j. The server will invoke $(c_2, K_2) \leftarrow \text{Kyber.Encaps}(pk_{U_i})$. The session key is $SK_{ij} = h(M_2||M_1||K_1||K_2||T_2)$. Although c_2, pk_{U_i} are transmitted over the public channel, due to the the security of the Kyber KEM, it is hard for $\mathcal{A}$ to obtain K_2. The forward and backward secrecy of the protocol is ensured.
- **key compromise impersonation attack:** The multi-server architecture comprises multiple cloud servers and user entities. A key compromise impersonation attack occurs when an adversary $\mathcal{A}$ obtains the long-term secrets of some participants and attempts to impersonate another user U_i or server S_j. Assuming that $\mathcal{A}$ comprises a subset of users $U_\mathcal{A} = \{U_{i1}, \cdots, U_{im}\}$ and a subset of servers $S_\mathcal{A} = \{S_{j1} \cdots, S_{jn}\}$, our protocol ensures that $\mathcal{A}$ cannot impersonate any uncompromised user $U_i \notin U_\mathcal{A}$ or server $S_j \notin S_\mathcal{A}$.
  - The adversary $\mathcal{A}$ cannot impersonate any user $U_i \notin U_\mathcal{A}$. To impersonate a legitimate user U_i, the adversary $\mathcal{A}$ must forge a valid login request. In our protocol, this requires $\mathcal{A}$ to successfully compute $AID_i = h(UID_i||R_i)$, where $(R_i, P_i) \leftarrow Gen(BIO_i)$. Since $\mathcal{A}$ does not comprise user U_i, $\mathcal{A}$ gains no information about BIO_i. Given that BIO_i has enough entropy, the security of fuzzy extractor ensures that R_i is nearly uniformly distributed. As a result, $\mathcal{A}$ cannot compute AID_i.
  - The adversary $\mathcal{A}$ cannot impersonate any server $S_j \notin S_\mathcal{A}$. To impersonate a legitimate server S_j, the adversary $\mathcal{A}$ must compute the correct message M_9 to pass the verification, where $M_9 = h(SK_{ij}||M_7||T_2)$, $SK_{ij} = h(M_5||M_7||K_1'||K_2||T_2)$, $M_5 = h(SID_j||PSR_j)$ and $K_1' = \text{Kyber.Decaps}(sk_{S_j}, c_1)$. To compute M_9, the adversary needs to correctly decapsulate c_1. Due to the security of the underlying Kyber.KEM, without the secret key sk_{S_j}, K_1' is pseudorandom. As a result, adversary $\mathcal{A}$ cannot compute the correct message M_9.
- **RC root key exposure attack:** In this attack, the RC's root key K is leaked to the adversary $\mathcal{A}$.
  - The adversary $\mathcal{A}$ cannot learn the identity of user U_i. Recall that in the registration phase, the user U_i selects an identity UID_i and a password PW_i and then imprints BIO_i. The user then invokes $(R_i, P_i) \leftarrow Gen(BIO_i)$, generates a random nonce a, and computes $PWU_i = h(UID_i||PW_i||a)$, $AID_i = h(UID_i||R_i)$ and $A_i = AID_i \oplus a$.
    Note that $AID_i = h(UID_i||R_i)$, where R_i is generated by the fuzzy extractor $Gen(BIO_i) = (R_i, P_i)$ and a is a random nonce in the computation of

$PWU_i = h(UID_i||PW_i||a)$. As long as BIO_i has enough entropy, the output R_i is almost uniformly distributed due to the security of the fuzzy extractor. Consequently, the user's identity UID_i remains concealed from the adversary.

– The adversary $\mathcal{A}$ cannot impersonate a legitimate server S_j. Recall that in the registration phase, the server S_j selects SID_j, computes $(pk_{S_j}, sk_{S_j}) \leftarrow$ Kyber.KeyGen(1^λ) and sends $\{SID_j, pk_{S_j}\}$ over a secure channel to the RC. Then, the RC computes $PSR_j = h(SID_j||K)$, publishes $\{SID_j, pk_{S_j}\}$, and stores PSR_j. We can see that even if the root key K is exposed to $\mathcal{A}$, $\mathcal{A}$ obtains no information about the secret key sk_{S_j}. Without sk_{S_j}, $\mathcal{A}$ cannot impersonate a legitimate server S_j.

5. Performance

This section compares our protocol with other related protocols [8,9,15–18,20–22,28,29,32,33] in terms of security performance and efficiency.

Table 5 shows the analysis of the security functionalities and attack resistance capabilities of the related protocols. Protocols [8,16,17,20,28,29] do not protect user anonymity. Protocols [8,16–18,20] do not ensure user untraceability. These protocols in [18] fail to achieve mutual authentication in the authentication phase. The secure smart card revocation phase does not exist in protocols [8,15,20,28,29,32,33]. The protocol in [18] cannot guarantee *SK* security. Protocols [8,15,21,28] cannot defend against replay attacks. Protocols [8,29,33] are prone to privileged insider attacks, protocols [16,29,33] are prone to smart card stolen attacks, protocols [17] are vulnerable to user impersonation attacks, and protocols [8,15,21,28] are susceptible to Dos. [8,15–18,20–22,32,33] cannot ensure post-quantum security. Compared to these protocols, our protocol can provide all the security functionalities and attack resistance capabilities mentioned previously.

Table 5. Security functionalities and attacks.

	[9]	[28]	[32]	[20]	[8]	[15]	[16]	[17]	[18]	[21]	[33]	[22]	[29]	Our
User anonymity	✓	✗	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓	✗	✓
User untraceability	✓	✓	✓	✗	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓
Mutual authentication	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
SK security	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
Secure smart card revocation	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓	✗	✓	✗	✓
Resist replay attack	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗	✓	✓	✓	✓
Resist privileged insider	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✗	✓
Resists offline password guessing attack	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Resist smart card stolen	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓
Resists user impersonation	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓
Resists DoS	✓	✗	✓	✓	✗	✗	✓	✓	✓	✗	✓	✓	✓	✓
Forward secrecy	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Post-quantum security	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓

To compare the computational and communication costs with other relevant protocols, we denote T_H as the time cost of the SHA-1 hash function, $T_{SE/D}$ for AES-128 symmetric encryption/decryption, T_{MExp} for the 2048-bit prime p modular exponential, and T_{EccM} for ECC-256 scalar multiplication. Let T_{Ke}, T_{En}, T_{De} denote the operation time for Kyber.Keygen,

Kyber.Encap and Kyber.Decaps, respectively. T_{Ec} and T_{Dc} represent the encryption and decryption times of Kyber's public-key encryption scheme used in [29]. These operations are implemented on a desktop configuration comprising a 64-bit operating system, Intel core i5-12400F @ 2.50 GHz, 18GB RAM, and using the Python 3.9.1 programming language. The execution time for T_H, $T_{SE/D}$, T_{MExp}, T_{EccM}, T_{Ke}, T_{En}, T_{De} is 0.0008 s, 0.0143 s, 0.1191 s, 0.0515 s, 0.0045 s, 0.0056 s, 0.007 s, respectively. The times for T_{Ec} and T_{Dc} are 0.0055 s and 0.0016 s. Additionally, we denote the execution time for fuzzy extractor, fuzzy commitment, and biohashing as T_{Fe}, T_{Fc}, and T_{Bh}. According to the article [34], the time of T_{Fe}, T_{Fc}, T_{Bh} is the same as T_{EccM}. The experimental computational times for polynomial multiplication, polynomial addition/subtraction, and PUF in [29] are $T_{PM} = 0.5115$ s, $T_{PAS} = 0.0006$ s, and $T_{PUF} = 0.0005$ s, respectively. The sizes of the hash value, identity, random number, timestamp, c and pk are 160 bits, 40 bits, 40 bits, 32 bits, 1184 bits, and 1088 bits, respectively.

Table 6 shows the comparison of our protocol with other relevant protocols in terms of the computational and communication costs. The table shows that the computational cost of our protocol is lower than those of most other relevant protocols, except protocols in [16,17,32]. The reason for the lower computational cost is that fewer hash functions and a faster KEM are used in our protocol. However, when it comes to communication costs, our protocol fails to demonstrate a clear advantage. The increased communication cost mainly stems from the use of KyberKEM, a lattice-based key encapsulation mechanism that provides post-quantum security. Compared to traditional cryptographic primitives, such as ECC or RSA, lattice-based operations generally involve larger data payloads, especially during key exchange and authentication phases. While this leads to higher communication overhead, it is a necessary trade-off to ensure long-term security in the face of future quantum threats.

Table 6. Computation and communication costs.

Protocol	Computation Cost (s)	Communication Cost (bit)
Auth1 in [29]	$17T_H + T_{Fe} + 2T_{Ec} + 2T_{Dc} \approx 0.0935$	7136
[28]	$19T_H + 2T_{PAS} + 4T_{PM} \approx 2.0624$	8448
[32]	$19T_H \approx 0.0152$	1800
[20]	$9T_H + 4T_{MExp} + T_{Bh} \approx 0.5351$	4520
[35]	$11T_H + 2T_{MExp} + T_{Bh} \approx 0.2985$	2688
[8]	$16T_H + 8T_{EccM} + 2T_{Bh} \approx 0.5278$	4864
[15]	$24T_H + 8T_{EccM} + 2T_{Bh} \approx 0.5342$	4288
[16]	$17T_H + T_{Fc} \approx 0.0651$	864
[17]	$14T_H + T_{Fc} \approx 0.0627$	1056
[18]	$16T_H + 3T_{SE/D} + T_{Fe} \approx 0.1072$	2096
[21]	$20T_H + 8T_{EccM} + T_{Bh} \approx 0.4795$	3408
[33]	$16T_H + 6T_{EccM} + 2T_{Bh} \approx 0.4248$	1568
[22]	$13T_H + 6T_{EccM} + 2T_{SE/D} + T_{Fe} \approx 0.3852$	1856
[29]	$19T_H + 2T_{SE/D} + 13T_{PM} + 11T_{PAS} + T_{PUF} + 2T_{Fe} \approx 6.7519$	4800
Our	$12T_H + T_{Ke} + 2T_{En} + 2T_{De} + T_{Fe} \approx 0.0908$	4192

Our approach is particularly well suited for medium- to high-security environments where post-quantum resilience is a priority, even at the cost of slightly increased communication overhead. Example application scenarios include the following: Government and military communication systems, where long-term data confidentiality is critical. Healthcare information systems, especially those handling sensitive patient records that must remain secure for decades. Critical infrastructure control systems, where future-proof security is essential despite moderate performance costs. Enterprise-level multi-service access platforms, where users need to access multiple internal services after a single registration, without relying on an online RC.

In summary, our protocol is designed for high-security environments where the benefits of post-quantum security, multi-server support, and offline RC functionality outweigh the need for minimal communication overhead.

6. Conclusions

This paper presents a post-quantum secure multi-factor authentication protocol designed for multi-server architecture and offline RC scenarios. The proposed protocol uses Kyber KEM and an information-theoretically secure fuzzy extractor as building blocks. A formal security analysis shows the semantic security of the proposed protocol in the real or random model. The informal security analysis shows that our protocol can resist various known attacks, including user anonymity, replay attacks, privileged-insider attacks and offline password guessing attacks. The performance analysis reveals that our protocol outperforms most existing multi-factor authentication protocols in computational efficiency and security. Despite a minor increase in communication overhead, this trade-off is offset by improvements in both security and computational performance.

Author Contributions: Conceptualization, Y.W., Y.S. and W.L.; methodology, Y.W., Y.S. and W.L.; formal analysis, Y.S., Y.W. and W.L.; investigation, Y.W., Y.S. and W.L.; writing—original draft preparation, Y.S.; writing—review and editing, Y.S.; visualization, Y.S.; supervision, Y.W. and W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No. 62102077) and the Shanghai Natural Science Foundation (Grant No. 24ZR1401300)

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lamport, L. Password authentication with insecure communication. *Commun. ACM* **1981**, *24*, 770–772. [CrossRef]
2. Ali, R.; Pal, A.K.; Kumari, S.; Karuppiah, M.; Conti, M. A secure user authentication and key-agreement scheme using wireless sensor networks for agriculture monitoring. *Future Gener. Comput. Syst.* **2018**, *84*, 200–215. [CrossRef]
3. Messerges, T.; Dabbish, E.; Sloan, R. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Comput.* **2002**, *51*, 541–552. [CrossRef]
4. Yoon, E.; Yoo, K. Robust biometrics-based multi-server authentication with key agreement scheme for smart cards on elliptic curve cryptosystem. *J. Supercomput.* **2013**, *63*, 235–255. [CrossRef]
5. Khan, A.S.; Javed, Y.; Saqib, R.M.; Ahmad, Z.; Abdullah, J.; Zen, K.; Abbasi, I.A.; Khan, N.A. Lightweight Multifactor Authentication Scheme for NextGen Cellular Networks. *IEEE Access* **2022**, *10*, 31273–31288. [CrossRef]
6. Tyagi, G.; Kumar, R. An Improved Multifactor User Authentication Scheme for Wireless Sensor Networks. *Wirel. Pers. Commun.* **2022**, *123*, 1311–1343. [CrossRef]
7. Pradhan, M.; Mohanty, S. A Blockchain-Assisted Multifactor Authentication Protocol for Enhancing IoMT Security. *IEEE Internet Things J.* **2024**, *11*, 39323–39332. [CrossRef]
8. Kumari, S.; Li, X.; Wu, F.; Das, A.K.; Choo, K.K.R.; Shen, J. Design of a provably secure biometrics-based multi-cloud-server authentication scheme. *Future Gener. Comput. Syst.* **2017**, *68*, 320–330. [CrossRef]
9. Chuang, M.C.; Chen, M.C. An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics. *Expert Syst. Appl.* **2014**, *41*, 1411–1418. [CrossRef]
10. Kumari, A.; Jangirala, S.; Abbasi, M.Y.; Kumar, V.; Alam, M. ESEAP: ECC based secure and efficient mutual authentication protocol using smart card. *J. Inf. Secur. Appl.* **2020**, *51*, 102443. [CrossRef]
11. Li, C.T.; Hwang, M.S. An efficient biometrics-based remote user authentication scheme using smart cards. *J. Netw. Comput. Appl.* **2010**, *33*, 1–5. [CrossRef]
12. Arshad, H.; Nikooghadam Nikoughadam, M. Three-Factor Anonymous Authentication and Key Agreement Scheme for Telecare Medicine Information Systems. *J. Med. Syst.* **2014**, *38*, 136. [CrossRef]
13. Shao, Z.; Li, Z.; Wu, P.; Chen, L.; Zhang, X. Multi-factor combination authentication using fuzzy graph domination model. *J. Intell. Fuzzy Syst.* **2019**, *37*, 4979–4985. [CrossRef]

14. Iftikhar, J.; Hussain, S.; Mansoor, K.; Ali, Z.; Chaudhry, S.A. Symmetric-Key Multi-factor Biometric Authentication Scheme. In Proceedings of the 2019 2nd International Conference on Communication, Computing and Digital systems (C-CODE), Islamabad, Pakistan, 6–7 March 2019; pp. 288–292. [CrossRef]
15. Feng, Q.; He, D.; Zeadally, S.; Wang, H. Anonymous biometrics-based authentication scheme with key distribution for mobile multi-server environment. *Future Gener. Comput. Syst.* **2018**, *84*, 239–251. [CrossRef]
16. Barman, S.; Das, A.K.; Samanta, D.; Chattopadhyay, S.; Rodrigues, J.J.P.C.; Park, Y. Provably Secure Multi-Server Authentication Protocol Using Fuzzy Commitment. *IEEE Access* **2018**, *6*, 38578–38594. [CrossRef]
17. Barman, S.; Shum, H.P.H.; Chattopadhyay, S.; Samanta, D. A Secure Authentication Protocol for Multi-Server-Based E-Healthcare Using a Fuzzy Commitment Scheme. *IEEE Access* **2019**, *7*, 12557–12574. [CrossRef]
18. Ali, Z.; Hussain, S.; Rehman, R.H.U.; Munshi, A.; Liaqat, M.; Kumar, N.; Chaudhry, S.A. ITSSAKA-MS: An Improved Three-Factor Symmetric-Key Based Secure AKA Scheme for Multi-Server Environments. *IEEE Access* **2020**, *8*, 107993–108003. [CrossRef]
19. Mishra, D.; Das, A.K.; Mukhopadhyay, S. A secure user anonymity-preserving biometric-based multi-server authenticated key agreement scheme using smart cards. *Expert Syst. Appl.* **2014**, *41*, 8129–8143. [CrossRef]
20. Chaturvedi, A.; Das, A.K.; Mishra, D.; Mukhopadhyay, S. Design of a secure smart card-based multi-server authentication scheme. *J. Inf. Secur. Appl.* **2016**, *30*, 64–80. [CrossRef]
21. Luo, H.; Wang, F.; Xu, G.; Meng, W. Provably Secure ECC-Based Three-Factor Authentication Scheme for Mobile Cloud Computing with Offline Registration Centre. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 8848032. [CrossRef]
22. Shukla, S.; Patel, S.J. A design of provably secure multi-factor ECC-based authentication protocol in multi-server cloud architecture. *Clust. Comput.* **2023**, *27*, 1559–1580. [CrossRef]
23. Ajtai, M.; Dwork, C. A public-key cryptosystem with worst-case/average-case equivalence. In Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC '97, El Paso, TX, USA, 4–6 May 1997; pp. 284–293. [CrossRef]
24. Applebaum, B.; Cash, D.; Peikert, C.; Sahai, A. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In Proceedings of the Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2009; Halevi, S., Ed.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 595–618.
25. Lyubashevsky, V.; Peikert, C.; Regev, O. On Ideal Lattices and Learning with Errors over Rings. In Proceedings of the Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, France, 30 May–3 June 2010; Gilbert, H., Ed.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 1–23.
26. Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehle, D. CRYSTALS-Kyber: A CCA-Secure Module-Lattice-Based KEM. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS P), London, UK, 24–26 April 2018; pp. 353–367. [CrossRef]
27. Bahache, A.N.; Chikouche, N.; Akleylek, S. Securing Cloud-based Healthcare Applications with a Quantum-resistant Authentication and Key Agreement Framework. *Internet Things* **2024**, *26*, 101200. [CrossRef]
28. Ahmad, A.; Jagatheswari, S. Quantum Safe Multi-Factor User Authentication Protocol for Cloud-Assisted Medical IoT. *IEEE Access* **2025**, *13*, 3532–3545. [CrossRef]
29. Palaniswamy, B.; Karati, A. QPTA: Quantum-Safe Privacy-Preserving Multi-Factor Authentication Scheme for Lightweight Devices. In Proceedings of the 21st International Conference on Security and Cryptography—SECRYPT, Dijon, France, 8–10 July 2024; pp. 804–811. [CrossRef]
30. Dodis, Y.; Ostrovsky, R.; Reyzin, L.; Smith, A. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM J. Comput.* **2008**, *38*, 97–139. [CrossRef]
31. Dolev, D.; Yao, A.C. On the Security of Public Key Protocols (Extended Abstract). In Proceedings of the 22nd Annual Symposium on Foundations of Computer Science (sfcs 1981), Nashville, TN, USA, 28–30 October 1981; pp. 350–357. [CrossRef]
32. Amin, R.; Biswas, G.P. A Novel User Authentication and Key Agreement Protocol for Accessing Multi-Medical Server Usable in TMIS. *J. Med. Syst.* **2015**, *39*, 1–17. [CrossRef]
33. Ryu, J.; Oh, J.; Kwon, D.; Son, S.; Lee, J.; Park, Y.; Park, Y. Secure ECC-Based Three-Factor Mutual Authentication Protocol for Telecare Medical Information System. *IEEE Access* **2022**, *10*, 11511–11526. [CrossRef]
34. Rehman, H.U.; Ghani, A.; Chaudhry, S.A.; Alsharif, M.H.; Nabipour, N. A secure and improved multi server authentication protocol using fuzzy commitment. *Multimed. Tools Appl.* **2021**, *80*, 16907–16931. [CrossRef]
35. Ali, R.; Pal, A.K. Three-Factor-Based Confidentiality-Preserving Remote User Authentication Scheme in Multi-server Environment. *Arab. J. Sci. Eng.* **2017**, *42*, 3655–3672. [CrossRef]