A Blockchain-Based Privacy-Awareness Authentication Scheme with Efficient Revocation for Multi-Server Architectures

LING XIONG\textsuperscript{1,2}, FA-GEN LI\textsuperscript{2}, SHENG-KE ZENG\textsuperscript{1,2}, TU PENG\textsuperscript{3} AND ZHI-CAI LIU\textsuperscript{1}

\textsuperscript{1}School of Computer and Software Engineering, Xihua University, Chengdu 610039, P.R. China
\textsuperscript{2}School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
\textsuperscript{3}School of Software, Beijing Institute of Technology, Beijing 100081, P.R. China

Corresponding author: Fa-gen Li (e-mail: fagenli@uestc.edu.cn).

This research was supported by National Science Foundation of China under Grant number (Grant No. 61502034 and 61571375) and the civil aviation administration of China (No. PSDSA201802)

\textbf{ABSTRACT} Multi-server authentication technology has become more and more popular with the extensive applications of networks. Although it has brought great convenience to people’s life, security becomes a critical issue and attracts lots of attentions in both academia and industry. Over the past two decades, a series of multi-server authentication schemes without the help of online registration center using the self-certified public key cryptography have been proposed to enhance security. However, it may cause the single-point failure problem due to the centralized architecture. Besides, user revocation facility is not well resolved in these schemes. To the best of our knowledge, blockchain technology has lots of advantages, bringing a promising solution to the problems of single-point failure and user revocation compared with the traditional cryptography technologies. In this work, we apply the idea and method of blockchain technology to construct a privacy-awareness authentication scheme for the multi-server environment, which can achieve distributed registry and efficient revocation. Moreover, the proposed scheme not only provides multiple security requirements like mutual authentication, user anonymity and perfect forward secrecy, but also resists various kinds of malicious attacks. The security of the proposed scheme is proved by rigorous formal proof using the random oracle model. Compared with recently related schemes, the proposed scheme has better communication performance, which make it be very suitable for real-life applications.

\textbf{INDEX TERMS} blockchain, multi-server, authentication, revocation

\section{I. INTRODUCTION}

With the rapid development of network and information techniques, many applications and services that are based on the Internet platform are emerging one after another. As a result, two-thirds of users tend to reuse the same identities and passwords for multiple applications or services to make memorizing them easier. Although this kind of handing brings much convenience to users, it also comes with a potential security risk. The multi-server authentication mechanism is an effective solution to address this barrier, which only needs users to register once at the registration center. Thus, users can access all registered servers using the same identity and password [1]. As shown in Fig. 1, a multi-server authentication system includes three roles, namely, a registration center (RC), servers and users [2], [3]. Generally, the RC is the system manager who is responsible to provide a trusted credential, which is also called certificate, to both servers and users. During the process of authentication, the server and the user can authenticate each other through this credential or certificate.

Due to the openness Internet network, the adversary can easily eavesdrop, insert, block, and alter the transmitted messages in the multi-server environment. Hence, it is indispensable to design privacy-awareness authentication schemes for multi-server environment [2]. Over the past two decades, a series of remarkable multi-server privacy-awareness authentication schemes (e.g. [2]–[6]) have been proposed. According to whether requires the help of online
RC in the authentication process, these schemes are divided into two types, namely, online RC schemes and no online RC schemes [2]. Obviously, online RC schemes increase the communication costs and complexity. Therefore, in recent years, no online RC schemes have gradually become the research focus. In this article, we concern with no online RC schemes for multi-server architectures. Generally, the self-certified public key cryptography (SCPKC) [2] has been used in multi-server authentication schemes to achieve no online RC. Although these schemes have many advantages, such as low communication overhead. However, there still exist some security and design issues needed to be resolved, like the following.

• **Single-point failure:** In the multi-server environment, when a new user wants to access a server, he/she has to register first. Generally, there is only a single RC in the traditional multi-server architecture. Thus, the only RC has full knowledge of the registered users’ information (such as identity, secret key, etc.) and can trace the actions of users [7]. Additionally, if the single RC is a failure under attacks or natural disasters, the whole stored data will be in danger [8].

• **User revocation:** Several circumstances in the multi-server system require the user revocation mechanism to revoke misbehaving-compromised users from the system within the stipulated expiration dates [9], [10]. To the best of our knowledge, the existing SCPKC multi-server authentication protocols (e.g. [2], [4], [10], [11]) adopts two measures to revoke users for access authorization. The first is the black/white (or revocation/permission) list mechanism [10]. Once the user is revoked, the RC will notify each server to add the revoked user to the black/white list. Thus, the RC and servers may require to manage a backend channel for the black/white list. The second is the expiration time method [11]. The user’s credential is bound by a time period. Before the expiration time, users remain legitimate unless the time has expired. Unfortunately, if the credential is obtained by an adversary within the expiration time, the adversary can access servers in the multi-server system using the old credential.

Certainly, traditional cryptography techniques generally may not be applicable to the above two issues. Blockchain technology, that has several additional technological advantages like decentralized, unforgeability, etc, offers a promising alternative solution. Motivated by this idea, in this work, we design a blockchain-based privacy-awareness authentication scheme for the multi-server system. The proposed scheme avoids the single RC problem and provides an effective user revocation method. Furthermore, our scheme can achieve mutual authentication, user anonymity, perfect forward secrecy, untraceability, and resistance to various attacks.

### A. RELATED WORK

In this section, we first introduce the recent related work of multi-server authentication schemes. Then a rough overview of the blockchain and its application in authentication technique will be described.

1) **Multi-server authentication**

In 2001, Li et al. [12] proposed the first password-based multi-server authentication scheme based on neural networks. While the neural networks are so complex that the scheme cannot be practical. To enhance efficiency, Juang [13] proposed a multi-server authentication scheme using symmetric cryptography. However, Juang’s scheme is vulnerable to insider attack and off-line dictionary attacks. In order to increase security, a series of improved schemes (e.g. [14]–[16]) had been proposed. However, these schemes still suffer from some security problems, such as perfect forward security, impersonation attack, user anonymity, etc. In 2009, Liao et al. [17] designed a dynamic ID-based authentication scheme for the multi-server environment. Unfortunately, this scheme cannot resist impersonation attack, server spoofing attack. Although several schemes [18]–[20] had been proposed to improve after Liao et al.’s scheme, there were still security weaknesses. In 2015, He and Wang [4] presented a robust biometrics-based authentication scheme for the multi-server environment. They claimed that their scheme could support various security requirements and resist a variety of attacks. Later, Odulu et al. [3] pointed out that He and Wang’s scheme was vulnerable to known session-specific temporary information attack, impersonation attack, wrong password login attack. To address these issues, they put forward a secure biometrics-based multi-server authentication protocol using smart cards, which can provide the problem of user revocation and resist various attacks. Most recently, more and more multi-server authentication schemes are be applied in various environments, such as cloud computing (e.g. [21]) and wireless sensor networks (e.g. [22]). Unfortunately, Odulu et al.’s scheme requires a trust third-party to participate in each authentication phase, which may make the trusted third party being a bottleneck of communication. To solve this issue, several multi-server authentication schemes without online third-party participation had been proposed [2], [23]–[26].
Obviously, these schemes using the SCPKC have lower communication cost. So, they have become very popular among researchers and have been applied to mobile cloud computing environment (e.g. [5], [11], [27]). However, to the best of our knowledge, these multi-server authentication schemes using the SCPKC cryptography adopt the black/white list mechanism or expiration time method to revoke users, which may cause communication costs or security problem. Additionally, all of these multi-server authentication schemes share a common problem: users have to register on a single trust third party. Therefore, how to design a multi-server authentication scheme with a distributed registry and efficient revocation is an urgent problem to be solved.

2) Blockchain and its application in authentication

In 2008, the blockchain was originally published in the cryptography mailing group by a scholar named Satoshi Nakamoto [28]. In recent years, with the increasing popularity of Bitcoin, blockchain technology research has been motivated to grow quickly. The blockchain is a distributed peer-to-peer network where transactions are posted and verified by non-trusting network members via a cryptographically verifiable manner [29], [30]. One of a key challenge in maintaining the blockchain data structure is the consensus algorithm, such as proof-of-work (PoW), proof-of-stake (PoS), delegated proof-of-stake (DPoS), etc. PoW utilizes a physical resource (either storage or computational power) to achieve the leader election process, in which miners have to compete to complete some difficult but easily verifiable task. Bitcoin [28], Namecoin [31] and Litecoin [32] are typical PoW-based cryptocurrencies. The disadvantage of this kind of consensus algorithm is that itexpends a lot of energy and causes a serious waste. PoS is an alternative consensus algorithm to resolve the waste of energy. Rather than miners investing computational resources, PoS randomly selects one of the miners proportionally to be the leader [33], [34]. Most recently, Kiayias et al. [35], [36] presented the first blockchain protocol named Ouroboros based on PoS with rigorous security guarantees, which offers qualitative efficiency advantages over blockchains based on PoW. DPoS [37] is a variant of PoS, in which the leader is performed by voting. Due to the better performance in computation and energy efficiency, many cryptocurrencies adopt PoS or DPoS as their consensus algorithm after Bitcoin.

The authentication technology based on blockchain has come to the foreground in recent years and receives more and more attentions [8], [38]. In 2014, Conner et al. [39], [40] proposed the first blockchain public key infrastructure (PKI) system called Certcoin, which provides a solution to some security problems, such as DigiNotar incident [41], in the traditional PKI. However, all network numbers can find the link between the identity and its corresponding public key by viewing the blockchain. Then, they can trace the actions of identities. Thus, privacy cannot be provided by Certcoin. To address this issue, Axon et al. [42], [43] designed a privacy-awareness blockchain PKI, which achieve user anonymity through short term online public keys. Obviously, Axon et al.’s scheme is sacrificing storage and efficiency in exchange for privacy. Different from the above blockchain PKI, Matsumoto and Reischuk [44] presented Instant Karma PKI (IKP), which offers automatic responses to certificate authority (CA) misbehavior using smart contract [45], [46] and incentives for those who help detect misbehavior. Although these existing schemes explore the potential of applying blockchain technology for authentication, there still exist many challenges. The current research on the blockchain for the multi-server system has not been reported. In this paper, we are to address these challenges.

B. CONTRIBUTIONS

In this paper, we present a blockchain-based privacy-awareness authentication scheme with efficient revocation for multi-server architectures. The contributions of the paper are summarized mainly as follows.

1) The proposed scheme focuses on the combination of the blockchain and multi-server authentication. The permission servers as blockchain network miners utilize Ouroboros algorithm to ensure the consistency. Thus the false issuing credential can be avoided.

2) The proposed scheme can solve the problem of a single RC.

3) The proposed scheme increases user revocation mechanism to prevent the misuse of the smart card when it is lost/stolen.

4) The proposed scheme has higher efficiency in communication, which makes it more suitable for real-life applications.

C. ORGANIZATION OF THE PAPER

The rest of this paper is organized as follows. Section II reviews the background for our system. Section III shows the system building blocks in our system. Section IV presents the detailed procedure of the proposed scheme. Section V gives security analysis of our scheme. The computation and communication costs analysis of the proposed scheme are discussed in Section VI. Finally, Section VII concludes this paper. All the notations mentioned in our proposed scheme are defined in Table 1.

II. BACKGROUND

This section will introduce the system model and the security requirements of our scheme.

A. SYSTEM MODEL

Protocol participant. The proposed scheme involves two participants: the user \( U_i \) and the server \( S_j \), \( S_j \) in the blockchain network is the service provider who is assessed by the remote user.

- Users: The remote users with smart card or mobile device, are able to access multiple servers. As shown in Fig. 2, when these users wish to ask for an access request...
TABLE 1. Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_i$</td>
<td>The remote user</td>
</tr>
<tr>
<td>$S_j$</td>
<td>The server</td>
</tr>
<tr>
<td>$ID_i$</td>
<td>Unique identity of $U_i$</td>
</tr>
<tr>
<td>$ID_{Sj}$</td>
<td>Unique identity of $S_j$</td>
</tr>
<tr>
<td>$PW_i$</td>
<td>Password of $U_i$</td>
</tr>
<tr>
<td>$REV_{U_i}$</td>
<td>Revocation status, if $REV_{U_i} = 1$, it specifies that $U_i$ has been revoked, otherwise not.</td>
</tr>
<tr>
<td>$G$</td>
<td>An additive group with order $q$</td>
</tr>
<tr>
<td>$Tx_1, Tx_2, ..., Tx_n$</td>
<td>The number of transaction in blockchain</td>
</tr>
<tr>
<td>$T$</td>
<td>Current time stamp values</td>
</tr>
<tr>
<td>$h_i(.), i = 0, 1, 2, 3, 4$</td>
<td>One-way hash function</td>
</tr>
<tr>
<td>$MAC(k, M)$</td>
<td>Authenticate a message $M$ using the entity’s key $k$</td>
</tr>
<tr>
<td>$X</td>
<td></td>
</tr>
</tbody>
</table>

| $\oplus$ | Concatenate operation |

to the multi-server system, they need to register in the nearest server first, who will post a corresponding transaction to the blockchain network (see Section IV-B).

- **Servers:** In our multi-server system, the permission servers as the role of miners or consensus nodes constitute the blockchain network. We assume that servers in our system are semi-trusted parties, which means that servers may misbehave on their own but will not conspire with either of the other servers [47]. So, the proposed system adopts private or consortium blockchain, which adopts an efficient consensus mechanism like PoS. As shown in Fig. 2, when a server $S_j$ receives registering request from a user $U_i$, he/she need to check the validity of user’s public key and personal information, such as passport, identification card, mobile number or any authorized identities. After successful verification, $S_j$ signs user’s identity and public key using her/his own private key and posts the signature to the blockchain network.

![Blockchain Network](Image)

**FIGURE 2.** The multi-server authentication architecture based on blockchain.

**Protocol execution.** The proposed scheme has five phases: the initialization phase, the user registration phase, the mutual authentication phase, the password update phase, and user revocation and re-registration phase. The initialization phase, the user registration phase and user revocation and re-registration phase are assumed to be executed securely.

**Adversary model.** The adversary $A$ has two goals. One is that $A$ can successfully impersonate $U_i$ authenticating to $S_j$, and the other is that $A$ can successfully impersonate $S_j$ authenticating to $U_i$. Assume that $A$ is a probabilistic polynomial time attacker, and the feasible attacks are summarized as follows:

- $A$ can control the channel between the user and the server. It means that $A$ can eavesdrop, insert, block, and alter the transmitted messages through the communication channel.
- $A$ can obtain one of the two authentication factors: the smart card or the password. If $A$ has obtained the smart card, he/she can extract the secret information in the smart card. Then he/she has the capability of enumerating the password space $|D_{PW}|$.
- $A$ may be another legitimate but malicious user in the multi-server system.

**B. SECURITY REQUIREMENTS**

According to the recent literatures for multi-server authentication (e.g. the literatures [2]–[5], [48]), the blockchain-based multi-server authentication scheme should satisfy the following security properties.

- **Mutual authentication:** It ensures that servers and users can successfully authenticate each other.
- **User anonymity:** It ensures that the adversary cannot obtain users’ identities through the transmitted messages in the public channel.
- **Un-traceability:** It ensures that the adversary cannot trace users’ behaviors from the transmitted messages in the public channel.
- **Efficient and user-friendly password update:** It ensures that users can freely update passwords and should be allowed updating passwords without servers’ assistance.
- **Multi-factor security:** Multi-factor (assuming there are $n$ factors, generally, $n = 2$ or $n = 3$) security implies the protocol is still secure when $n - 1$ of $n$ factors are lost [49], [50]. In our proposed scheme, we adopt $n = 2$, the password and the smart card are two used factors. So, it ensures that the blockchain-based two-factor authentication scheme for multi-server architecture should be able to satisfy the following requirements. 1) If an adversary has obtained the smart card and gets its secret value, he/she should not be able to perform the offline password guessing attack; 2) The adversary who knows the password should not be able to perform impersonation attack without secret value in the smart card [48], [51].
- **Resistance to wrong password login/update attack:** To avoid the waste of computation and communication resources for invalid login, it is necessary to check the
correctness of the password in the user login procedure. Besides, once a mistake occurs in the password update phase, a valid user can no longer log in the server using the same smart card. Therefore, the blockchain based multi-server authentication scheme should consider quick detection mechanism to avoid wasting the server’s resources [3], [48].

- **User revocation and re-registration**: It ensures that the blockchain based multi-server authentication scheme should support user revocation and re-registration. If the user’s smart card is lost or stolen, there must be some measures to prevent the adversary to impersonate the user. In other words, if an adversary has obtained the identity of the user, he/she cannot impersonate the user in the registration phase [3], [48].

- **Secure session key agreement**: It ensures that two participants should be able to agree with a secure session key, which will protect transmitted messages in future communications.

- **Perfect forward secrecy**: It ensures that the adversary is unable to obtain the session key generated in previous sessions even if the long-term private keys of the two participants are leaked.

- **Resistance to various attacks**: It ensures that various attacks should be prevented in the multi-server environment, such as impersonation attack, man-in-the-middle attack, replay attack and stolen-verifier attack.

III. SYSTEM BUILDING BLOCKS

In this section, we will introduce the cryptographic primitives, transaction and consensus mechanism of blockchain network used in the proposed blockchain-based scheme.

A. CRYPTOGRAPHIC PRIMITIVES

The proposed scheme leverages the elliptic curve digital signature algorithm ECDSA [52]. The digital signature consists of three algorithms, which will be reviewed as follows:

- $\text{keygen}(1^k) \rightarrow (SK, PK)$: the function generates a private key $SK$ and a corresponding public key $PK$ with the security parameter $k$.

- $\text{Sig}(SK, m) \rightarrow S_i$: the function computes a digital signature value $S_i$ of message $m$ using the private key $SK$.

- $\text{Ver}(PK, S_i, m) \rightarrow b \in \{0, 1\}$: the function verifies whether the value $S_i$ is correct signature value of message $m$ using the public key $PK$.

The signature algorithm should be unforgeable [39], [53], which means that no probabilistic polynomial-time adversary can forge a legal signature value $S_i$ without the private key $SK$.

B. TRANSACTION

As shown in Table 2, instead of posting transactional information in the transaction of the bitcoin system, the transactions (Tx) in our system include identity, public key, revocation status and signature. The detail of the transaction structure is described below.

- **from**: represents the identity of the user.
- **UPK**: represents the public key linked with the user.
- **to**: represents the identity of the server who handle registration information from the user.
- **SPK**: represents the public key of the server who handle registration information from the user.
- **REV**: represents the revocation status of the user. If the value of REV is one, it represents the user is revoked. Otherwise not.
- **T**: represents the current timestamp.
- **USIG**: represents the signature value of the user’s information (the identity of the user, the revocation status, the current timestamp) with the user’s private key.
- **SSIG**: represents the signature value of the user’s information (the identity of user, the public key of the user, the identity of the server, the revocation status, the current timestamp) with the server’s private key.

<table>
<thead>
<tr>
<th>from</th>
<th>UPK</th>
<th>to</th>
<th>SPK</th>
<th>REV</th>
<th>T</th>
<th>USIG</th>
<th>SSIG</th>
</tr>
</thead>
</table>

C. BLOCKCHAIN

The blockchain is made of a chronologically ordered chain of blocks. Every block includes a certain amount of transactions and each block links to its predecessor by a hash value [54]. As shown in Fig. 3, the structure of our blockchain system is similar to the Bitcoin [39], which includes the number of block, the hash value of the previous block, the timestamp and Merkle tree root. In our blockchain-based multi-server system, the permission servers are miners in blockchain network, who will participate in issuing the next block. Generally, miners have to compete to complete some PoW to create a new block (e.g. Bitcoin [39], Namecoin [31]). However, these systems have always relied on large computing power to verify transactions and write them into a new block, which costs a lot of money and energy. Another alternative to PoW is the concept of PoS [35], [36] or DPoS [37], which randomly selects one of the miners to complete a new block. Obviously, the two later consensus mechanisms PoS and DPoS are more effective. In our design, a provable secure PoS protocol name Ouroboros [35], [36] is selected as our consensus mechanism to write a new block. Ouroboros can process hundreds or a couple of thousand transactions within seconds and eliminate the needs for an energy-hungry PoW. When a user initiates a registration requirement to the server the blockchain node (the server), the blockchain node will verify his/her information. After successful checking, the blockchain node will post the transaction into the blockchain network and the whole nodes will generate a new block through Ouroboros algorithm.
The number of block
Hash of previous block
header
Time stamp
Random
Hash of current block header

Head of block

Previous block

FIGURE 3. The bitcoin blockchain structure.

IV. THE PROPOSED SCHEME

This section will describe the details of the proposed privacy-awareness authentication scheme. Our proposed scheme consists of five phases: initialization phase, user registration phase, mutual authentication phase, password update phase, and revocation and re-registration phase. Each phase in detail will be introduced as follows.

A. INITIALIZATION PHASE

Assuming that there are $n$ permission servers. In the initialization phase, all servers $S_j$ agrees upon an additive group of point $G$ with order $q$, $P$ is a generator of $G$, and five hash functions $h_i : \{0, 1\}^* \rightarrow \{0, 1\}^{l_i}$, $h_2 : \{0, 1\}^* \rightarrow \{0, 1, 2, ..., 1023\}$, $h_4 : \{0, 1\}^* \rightarrow Z_q^*$, where $l$ is the bit length of output and $i = 0, 1, 3$. Every server $S_j$ generates its private key $SK_{sj} \in Z_q^*$ and calculates the public key $PK_{sj} = SK_{sj} \cdot P$. We assume without loss of generality that each public key $PK_{sj}$ is knowing by all servers and users. Then $S_j$ stores $SK_{sj}$ into its memory as secret and publishes the parameters $\{G, P, PK_{sj}, h_0, h_1, h_2, h_3, h_4\}$.

B. USER REGISTRATION PHASE

When a user $U_i$ wants to access a multi-server system, he/she must register with any one of the servers in multi-server architectures. As shown in Fig. 4, the procedure of user registration is described as follows.

1. A user $U_i$ chooses a nearest server $S_j$ to him/her in the blockchain network, selects identity $ID_i$ and a random number $K_{ui} \in Z_q^*$, sets $REV_{ui} = 0$, and calculates the public key $P_{ui} = K_{ui} \cdot P, S_{ui} = Sig(K_{ui}, ID_i||T1||REV_{ui})$, where $T1$ is the timestamp. $U_i$ submits the messages $\{T1, ID_i, reg, P_{ui}, S_{ui}\}$ and his personal information (e.g. passport, identification card and mobile number) to $S_j$ through a secure channel. (The signature $S_{ui}$ demonstrates that the user is able to sign with $K_{ui}$).

2. Upon receipt of the message, $S_j$ first checks the correctness of personal information and timestamp. Then, $S_j$ sets $REV_{ui} = 0$ and verifies whether the equation $Ver(P_{ui}, S_{ui}, ID_i||T1||REV_{ui}) = 1$ holds. If it does not hold, $S_j$ rejects the registration request.

Otherwise, $S_j$ checks whether $ID_i$ has been previously registered through lookup the blockchain. If it has been registered and $REV_{ui} = 0$, $S_j$ rejects the registration request. Otherwise, $S_j$ computes $S_{jui} = Sig(SK_{sjj}, ID_i||ID_{sjj}||P_{ui}||REV_{ui}) = 0 || T2 \}$ and broadcasts the transaction $\{ID_i, P_{ui}, ID_{sjj}, PK_{sjj}, REV_{ui}, T2, S_{ui}, S_{jui}\}$ to blockchain network, where $T2$ is the timestamp. After that, the block miners generate a new candidate block $N_{ui}i$ by Ouroboros [35], [36] algorithm, where $N_{ui}i$ is the number of block in the blockchain. Finally, $S_j$ transmits $\{N_{ui}i\}$ to $U_i$ through a secure channel. (The signature $S_{jui}$ demonstrates that the server $S_j$ has verified that $P_{ui}$ is the corresponding public key of the identity owner. $S_j$ has to take responsibility for this claim.).

3. After received the message, $U_i$ selects password $PW_i$ and a random number $b_i$. Then, $U_i$ computes $C_i = h_0(ID_i||PW_i||b_i), F_i = (K_{ui}||N_{ui}) \oplus C_i, V = h_3(h_2(K_{ui}||N_{ui}||C_i))$. Finally, $U_i$ stores $P_{ui}, F_i, V$ and $b_i$ into the secret memory of smart card.

C. MUTUAL AUTHENTICATION PHASE

When the user $U_i$ wants to log in a server $S_j$, $U_i$ needs to achieve mutual authentication with $S_j$. As shown in Fig. 5, the process of mutual authentication is as follows.

1. $U_i$ inputs $ID_i$ and $PW_i$ into the smart card. The smart card computes $C_i = h_0(ID_i||PW_i||b_i), K_{ui}||N_{ui} = F_i \oplus C_i, V' = h_3(h_2(K_{ui}||N_{ui}||C_i))$ and checks whether $V'$ and $V$ are equal. If not, the smart card terminates this session. Otherwise, it generates a random number $x \in Z_q^*$, and computes $X = x \cdot P, HID_i = h_4(X||ID_i||ID_{sjj}||N_{ui}||T), st = x+HID_i K_{ui} mod q, CT = (ID_i||ID_{sjj}||N_{ui}||st) \oplus h_1(x \cdot PK_{sj})$, where $T$ is the current timestamp. Then $U_i$ sends the messages $\{T, X, CT\}$ to $S_j$ by a public channel.

2. After received the messages $\{T, X, CT\}$, $S_j$ first checks the timestamp $T$, and computes $ID_i||ID_{sjj}||N_{ui}||st = CT \oplus h_1(SK_{sj} \cdot X)$. Then, $S_j$ checks whether the following three conditions are satisfied.

• whether $ID_i$ exists in the block $N_{ui}$. 

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.
• whether IDj’s revocation status REVui = 0 holds in the block Nui.
• the blocks Nui > N0ui do not include the tuple
\{IDi, Pui, IDsj, PKsj, REVui = 1, Sui, Sjui\}.

If one of the above conditions does not hold, Sj terminates the session. Otherwise, Sj gets the public key Pui of Uj, and verifies whether the equation st · P = X + h4(X)|IDi||IDsj||Y||X||st||x·Y holds. If it does not hold, Sj terminates the session. Otherwise, Sj generates a random number y ∈ Z∗
2, and computes
V = MAC(Key, IDsj||IDi||Y||X). If V ≠ V aborts; otherwise, accepts ...

x, computes X = x · P, HIDi = h4(X||IDi||IDsj||Nui)||T), st = x + HIDiKui mod q, CT = (IDi||IDsj||Nui||st) ⊕ h1(x · PKsj)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2939368, IEEE Access

E. USER REVOCAITION AND RE-REGISTRATION PHASE

Once the smart card is lost or stolen, the user must revoke his/her account and re-register with any one of the servers in multi-server architectures using the same identity. The process of user revocation and re-registration is described as follows.

1. Uj chooses the nearest server Sj in the multi-server system, selects a new random number K′uj ∈ Z∗
3, set REVui = 0, and calculates the public key P′uj = K′uj · P, S′ui = Sig(K′uj, IDi||T3||REVui), where T3 is the timestamp. Uj submits the messages \{T3, IDi, rev, P′uj, S′ui\} and some personal information to Sj through a secure channel, where rev is the revocation requirement.

2. Upon receipt of the messages, Sj at first checks the correctness of personal information and timestamp. Then Sj set REVui = 0, and verifies whether the equation V er(P′ui, S′ui, IDi||T3||REVui) = 1 holds. If they do not hold, Sj rejects the revocation and re-registration request. Otherwise, Sj gets the corresponding old public key of IDi by looking up blockchain, computes SJ1ui = Sig(SKsj, IDi||IDsj||Pui||REVui = 1||T3), S2ui = Sig(SKsj, IDi||IDsj||Pui||REVui = 0||T3), broadcasts two transactions \{IDi, Pui, IDsj, PKsj, REVui = 1, T4, Sui, Sj1ui\}, \{IDi, P′ui, IDsj, PKsj, REVui = 0, T5, S′ui, Sj2ui\} to blockchain network, where T4 and T5 are current timestamp. The block miners generate a new candidate block N′ui by Ouroboros algorithm, where N′ui is the number of block in the blockchain. Then, Sj transmits \{N′ui\} to Uj through a secure channel.

3. After received the message, Uj selects password PW′i and a random number b′. Then, Uj computes C′ = h0(IDi||PW′i||b′), F′ui = (K′ui||N′ui||C′) ⊕ h3(h2(K′ui||N′ui||C′)). Finally, Uj stores F′ui and b′ into the secret memory of smart card.

V. SECURITY ANALYSIS OF THE PROPOSED SCHEME

In this section, we will show our proposed scheme meets all the security requirements in Section II. Because the initialization phase, user registration phase, and user revocation and re-registration phase are executed in the secure channel. The proposed scheme may suffer security and privacy threats in the authentication phase. Therefore, in this section, we demonstrate the authentication phase is secure.

Security Model. Based on literature [50], [53], [55]–[57], we proposed a security model for our scheme. The security model of our scheme is defined by a game played by a probabilistic polynomial time (PPT) adversary A and a PPT Turing machine C. Let instance \[ \prod_{s}^{r} \] be the user oracle in session s, \[ \prod_{s}^{r} \] be the server oracle in session s. A can make oracle queries as follows.

1. ExtractUi − Oracle: This query simulates A registration as a legitimate user Ui. A issues this inquiry...
with \( U_i \)'s identity \( ID_i \), \( \zeta \) generates the number of block \( N_{ui} \), \( U_i \)'s private key and public key, stores them in the list \( L_U \) and returns \( N_{ui} \) and \( ID_i \) to \( A \).

(2) \( \text{ExtractSj} - \text{Oracle} \): This query simulates \( A \) registration as a legitimate server \( S_j \). \( A \) issues this inquiry with \( S_j \)'s identity \( ID_{sj} \). \( \zeta \) generates \( S_j \)'s private key and public key, stores them in the list \( L_S \).

(3) \( \text{Send} - \text{Oracle}(t, s, t', M) \): This query simulates the participate \( s \) sends message \( M \) to the oracle \( \prod_S^\zeta \). \( A \) issues inquiry and receives a response which is specified by the protocol.

(4) \( \text{Reveal} - \text{Oracle}(U_1, S_j, s) \): This query simulates the leakage of session key attack and will output the session key \( \text{Key} \).

(5) There are three corruption queries:
   a) \( \text{Corrupt}(ID_i, PW_i) \): This query simulates password leakage attack, and will output the user password \( PW_i \).
   b) \( \text{Corrupt}(ID_i, SC_i) \): This query simulates the smart card loss attack, and will output the secret information stored in the smart card \( SC_i \).
   c) \( \text{Corrupt}(S_j) \): This query simulates the server compromise attack.

Definition 1: Matching sessions: The session in instance \( \prod_U^s \) and the session in instance \( \prod_S^s \) are said to be matching if \( s = s', pid_U = S, pid_S = U \) and both have accepted, where \( pid_U \) and \( pid_S \) denote a peer identity.

Definition 2: Security authentication protocol: A authentication scheme is secure if the following properties hold:
   - \( \prod_U^s \) and \( \prod_S^s \) are matching session, and they accept each other.
   - \( \prod_U^s \) and \( \prod_S^s \) derive the same key.
   - The probability of \( \prod_U^s \) accepted \( A \) as \( \prod_S^s \) is negligible.
   - The probability of \( \prod_S^s \) accepted \( A \) as \( \prod_U^s \) is negligible.

A. FORMAL SECURITY ANALYSIS

To prove the security of our proposed scheme, we assume that our scheme is defined by a game played an adversary \( A \) and a Turing machine \( \zeta \). At first, we give two mathematical problems used for our security analysis as follows.

Definition 3: Discrete Logarithm (DL) Problem: Given \( X = x \cdot P \), where \( x \in \mathbb{Z}_q^* \), \( X \in G \), it is infeasible to compute \( x \).

Our concrete protocol is as below.

1) \( U_i \rightarrow S_j: M_1 = \{T, X, CT\} \).
2) \( S_j \rightarrow U_i: M_2 = \{Y, V\} \).

Lemma 1: (Secure User Authentication): In the proposed scheme, if \( h_0, h_1, h_2, h_3, h_4 \) are ideal random functions, the DL problem is hard and \( \prod_S^s \) has been accepted, then there is no polynomial adversary against our proposed scheme who can forge a legal user authentication message with a non-negligible probability.

Proof. We assume that the adversary \( A \) can forge a legitimate user authentication message with a non-negligible probability. Then there is a PPT Turing machine \( \zeta \) who can win the DL problem with a non-negligible probability by employing \( A \). We assume that the probability of the advantage of DL problem is \( Pr_{adv}[DL] \).

Given an instance \((P, P_{uc} = K_{uc} \cdot P)\) of DL problem, the task of \( \zeta \) is to compute \((K_{ui} \in \mathbb{Z}_q^*)\). To win the game, \( \zeta \) must simulate an environment of our proposed scheme which is indistinguishable from the real proposed scheme to the adversary \( A \). Hence, \( \zeta \) should answer all oracle queries issued by \( A \). To achieve this goal, \( \zeta \) needs to generate all initialization parameters \( \{G, P, PK_{sj}, h_0, h_1, h_2, h_3, h_4\} \) and public them. Besides, \( \zeta \) needs to generate all users’ private key \( SK_i \in \mathbb{Z}_q^* \) except for the challenger \( ID_i \)'s private key \( K_{uc} \) and calculates their public key \( P_{ui} = K_{ui} \cdot P \). \( \prod_U^s \) denotes the user oracle. \( \prod_S^s \) denotes the server oracle. Then, \( \zeta \) answers \( A \)'s queries as follows:

1) \( Hi(m_i): \) The hash query \( HI(m_i), i = 0, 1, 2, 3, 4 \) maintains a list \( L_{hi} \) with initialized empty. \( \zeta \) checks whether the message \( m_i \) exists in \( L_{hi} \). If it exists, \( \zeta \) returns its value \( h_i \) to \( A \). Otherwise, \( \zeta \) generates a random number \( h_i \), stores the tuple \((m_i, h_i)\) into \( L_{hi} \) and returns \( h_i \) to \( A \).
2) \( \text{ExtractUi} - \text{Oracle} \): In this query \( \zeta \) maintains a list \( L_U \) with initialized empty. \( \zeta \) checks if a tuple \((ID_i, P_{ui}, K_{ui}, N_{ui})\) exists in \( L_U \). If it exists, \( \zeta \) returns \( ID_i \) and \( N_{ui} \) to \( A \). Otherwise, \( \zeta \) operates as follows:
   - If \( ID_i = ID_{jc} \), \( \zeta \) generates a random number as the number of block \( N_{ui} \), selects \( K_{ui} = 1 \), and asks the user oracle \( \prod_U^s \) to get \( ID_i \)'s public key \( P_{ui} \). \( \zeta \) stores the tuple \((ID_i, P_{ui}, K_{ui}, N_{ui})\) into \( L_U \) and returns \( ID_i \) and \( N_{ui} \) to \( A \).
   - If \( ID_i \neq ID_{jc} \), \( \zeta \) generates a random number as the number of block \( N_{ui} \), selects a random number \( K_{ui} \in \mathbb{Z}_q^* \), and calculates the public key \( P_{ui} = K_{ui} \cdot P \). \( \zeta \) stores the tuple \((ID_i, P_{ui}, K_{ui}, N_{ui})\) into \( L_U \) and returns \( ID_i \) and \( N_{ui} \) to \( A \).
3) \( \text{ExtractSj} - \text{Oracle} \): In this query, \( \zeta \) maintains a list \( L_S \) with initialized empty. \( \zeta \) checks if a tuple \((ID_{sj}, PK_{sj}, SK_{sj})\) exists in \( L_S \). If it exists, \( \zeta \) returns \( ID_{sj} \) to \( A \). Otherwise, \( \zeta \) generates a random number \( SK_{sj} \), calculates \( PK_{sj} = SK_{sj} \cdot P \), stores the tuple \((ID_{sj}, PK_{sj}, SK_{sj})\) into \( L_S \) and returns \( ID_{sj} \) to \( A \).
4) \( \text{Send} - \text{Oracle}(U_i, s, S_j, M) \): In this query, \( A \) sends the first message \( M_1 \) to \( \zeta \). \( \zeta \) decrypts CT and obtains \( ID_i \) and \( P_{ui} \). \( \zeta \) operates according to the specification of the proposed scheme and returns \( M_2 \) to \( A \).
5) \( \text{Send} - \text{Oracle}(S_j, s, U_i, M) \): After receiving this query, \( \zeta \) checks whether the equation \( ID_i = ID_C \) holds. If not, \( \zeta \) operates according to the specification of the proposed scheme and returns the first message \( M_1 \) to \( A \). Otherwise, \( \zeta \) asks the user oracle \( \prod_U^s \) to get \( M_1 \) and returns it to \( A \).
6) \( \text{Reveal} - \text{Oracle}(U_i, S_j, s) \): In this query, \( \zeta \) retrans the session key \( \text{Key} \) between \( U_i \) and \( S_j \) in session \( s \).
7) \( \text{Corrupt}(ID_i, one factor) \): After receiving this query, \( \zeta \) asks \( \prod_U^s \) to send the corresponding password \( PW_i \).
or the secret parameters in smart card \( SC_i \). If \( ID_i = \overline{IDC} \), \( \zeta \) aborts the game.

8) \( \text{Corrupt}(S_j) \): After receiving this query, \( \zeta \) returns the private key of the server \( S_j \).

According to above queries, if \( A \) can successfully pass user authentication, it means that \( A \) has successfully forged a authentication message \( \{ T, X, CT \} \) and sends it to \( \zeta \), where \( CT = (ID_i)[|ID_j|[N_{ui}][st]) \oplus h_1(x \cdot PK_{sj}), st = x + HID_iK_{ui} \). Based on the forking lemma [58], \( A \) has successfully forged another authentication message \( \{ T, X, CT' \} \) via repeat the simulation with a difficult value of \( h_4 \). Thus, we gets the below two equations.

\[
\begin{align*}
st &= x + HID_iK_{ui} \quad (1) \\
st' &= x + HID_iK_{ui} \quad (2)
\end{align*}
\]

Based on equations \( (1) \) and \( (2) \), we get the following equations

\[
\begin{align*}
st - st' &= (HID_i - HID_i')K_{ui} \quad (3)
\end{align*}
\]

\( \zeta \) computes \( (st - st')(HID_i - HID_i')^{-1} \) as the answer of DL problem. The probability of it is analyzed below.

We assume that \( \epsilon \) is the non-negligible probability of \( A \) forges a legal authentication message and \( \rho \) is the probability of \( \zeta \) winning the DL problem when \( A \) has failed to forge the user authentication message. Thus, the probability of \( \zeta \) winning the DL problem may be reduced to the following value similar to that of reference [50].

\[
Pr_{\text{win}}[DL] = \frac{1}{q_s} \cdot (\epsilon + (1 - \epsilon) \cdot \rho) = \frac{\epsilon + (q_s - \epsilon) \cdot \rho}{q_s} \quad (4)
\]

Where \( q_s \) denote the number of Send query. Based on the above analysis, \( Pr_{\text{win}}[DL] \) is non-negligible and \( \zeta \) can win the DL problem with non-negligible. Obviously, it is a contradictory assumption. Therefore, there is no polynomial adversary can forge a legitimate user’s authentication message with a non-negligible probability.

**Lemma 2: (Secure Server Authentication):** In our proposed scheme, if \( h_0, h_1, h_2, h_3, h_4 \) and the message authentication code (MAC) are ideal random functions, and \( \prod_i \) has been accepted, then there is no polynomial adversary against the proposed scheme who can forge a legal server authentication message with a non-negligible probability.

**Proof.** We assume that the adversary \( A \) can forge a legal server authentication message with a non-negligible probability. Then there is a PPT Turing machine \( \zeta \) who can win the underlying game of MAC (Game-MAC) without knowing the secret session key \( Key \) with a non-negligible probability by employing \( A \).

The Game-MAC has two participants: a challenger and a MAC oracle \( \prod_{MAC} \), which has the secret key \( Key \). The challenger can ask \( \prod_{MAC} \) for the MAC value of any message as many times as he/she wants. Let \( Pr_{\text{win}}[MAC] \) is the probability that the challenger won the game. The game is described as the following three steps:

- The challenger sends two difficult messages \( m_0 \) and \( m_1 \) to the MAC oracle \( \prod_{MAC} \).
- The oracle chooses a random bit \( b \in \{0, 1\} \). If \( b = 1 \), the oracle returns \( MAC(Key, m_0) \) to \( A \), otherwise \( MAC(Key, m_1) \) is returned.
- The challenger guesses the value of \( b' \). If \( b' = b \), it means that the challenger wins the game.

To win the Game-MAC, \( \zeta \) must simulate an environment of our proposed scheme which is indistinguishable from the real proposed scheme to the adversary \( A \). Hence, \( \zeta \) should answer all oracle queries issued by \( A \). Firstly, \( \zeta \) setups all system parameters except challenger \( ID_sc \)'s private key \( SK_{sc} \). \( \zeta \) answers the Hi query, Execute-Oracle query and Reveal-Oracle query as he does in the proof of Lemma 1. Then, \( \zeta \) answers \( A \)'s queries as follows:

1) \( \text{Extract}ui − \text{Oracle} \): In this query \( \zeta \) maintains a list \( L_U \) with initialized empty. \( \zeta \) checks if a tuple \( (ID_i, P_{ui}, K_{ui}, N_{ui}) \) exists in \( L_U \). If it exists, \( \zeta \) returns \( ID_i \) and \( N_{ui} \) to \( A \). Otherwise, \( \zeta \) generates a random number as the revocation status value \( N_{ui} \), selects a random number \( K_{ui} \in Z_q \) and calculates the public key \( P_{ui} = K_{ui} \cdot P \). \( \zeta \) stores the tuple \( (ID_i, P_{ui}, K_{ui}, N_{ui}) \) into \( L_U \) and returns \( ID_i \) and \( N_{ui} \) to \( A \).

2) \( \text{Extract}sj − \text{Oracle} \): In this query, \( \zeta \) maintains a list \( L_S \) with initialized empty. \( \zeta \) checks if a tuple \( (ID_{sj}, PK_{sj}, SK_{sj}) \) exists in \( L_S \). If it exists, \( \zeta \) returns \( ID_{sj} \) to \( A \). Otherwise, \( \zeta \) operates as follows:
- If \( ID_{sj} = ID_{sc} \), \( \zeta \) sets \( SK_{sj} = \bot \), and asks the server oracle \( \prod_{MAC} \) to get \( ID_{sj} \)’s public key \( PK_{sj} \), stores the tuple \( (ID_{sj}, PK_{sj}, SK_{sj}) \) in \( L_S \) and returns \( ID_{sj} \) to \( A \).
- If \( ID_{sj} \neq ID_{sc} \), \( \zeta \) generates a random number \( SK_{sj} \), calculates \( PK_{sj} = SK_{sj} \cdot P \), stores the tuple \( (ID_{sj}, PK_{sj}, SK_{sj}) \) in \( L_S \) and returns \( ID_{sj} \) to \( A \).

3) \( \text{Send} − \text{Oracle}(ui, s, sj, M) \): In this query, \( A \) sends the first message \( M_1 \) to \( \zeta \), \( \zeta \) operates according to the specification of the proposed scheme and returns \( M_2 \) to \( A \). After receiving \( M_2 \) from \( A \), \( \zeta \) sends the result of user authentication messages according to \( M_1 \) and \( M_2 \) and asking \( \prod_{MAC} \) in order to verify the MAC value.

4) \( \text{Send} − \text{Oracle}(sj, s, Ui, M) \): After receiving this query, \( \zeta \) sends the first message \( M_2 \) as the protocol specified using the user’s private key to \( A \). If \( ID_{sj} = ID_0 \), \( \zeta \) aborts the game.

5) \( \text{Corrupt}(ID_i, one factor) \): After receiving this query, \( \zeta \) asks \( \prod_i \) to send the corresponding password \( PW_i \) or the secret parameters in smart card \( SC_i \).

6) \( \text{Corrupt}(S_j) \): After receiving this query, \( \zeta \) checks whether the equation \( ID_{sj} = ID_0 \) holds. If not, \( \zeta \) returns the private key of the server \( S_j \). Otherwise, \( \zeta \) aborts the game.

According to above queries, if \( A \) can successfully pass server authentication, it means that \( A \) has successful forged a
Our proposed scheme can provide user anonymity and untraceability. Thus, the probability of a probability of winning the Game-MAC may be reduced to the following value similar to that of reference [50].

\[
Pr_{\text{win}}[\text{MAC}] = \frac{1}{2} q_s \cdot (\epsilon + (1 - \epsilon) \cdot \frac{1}{2} + \frac{q_s - 1}{2} \cdot \frac{1}{2} - \frac{1}{2}) = \frac{2}{2q_s} 
\]

Based on the above analysis, \(Pr_{\text{win}}[\text{MAC}]\) is non-negligible and \(\zeta\) can win the Game-MAC with non-negligible probability. Obviously, it is a contradictory assumption. Therefore, there is no polynomial adversary can forge a legitimate server’s authentication message with a non-negligible probability.

**Theorem 1:** Our proposed scheme is secure protocol, if: (A) \(\prod_U^m\) and \(\prod_S^s\) have been accepted; (B) \(h_0, h_1, h_2, h_3, h_4\), MAC are ideal random functions; (C) the DL problem is hard.

**Proof.** Based on Lemma 1 and Lemma 2, we can know that there is no polynomial adversary can forge a legal user or server if MAC is ideal random function and the DL problem is hard. Besides, since \(\prod_U^m\) has been accepted, it can ensure that there is a peer (\(\prod_S^s\)) session of the scheme that has derived precisely the same key. According to Definition 2, the proposed scheme is a secure protocol.

**B. FURTHER SECURITY ANALYSIS OF THE PROPOSED SCHEME**

1) **Mutual authentication**

According to Theorem 1, we can conclude that there is no polynomial adversary can forge a legal user or server if DL problem is hard and MAC is an ideal random function. Therefore, the user and the server can successfully authenticate each other.

2) **User anonymity and untraceability**

To protect user’s real identity, our proposed scheme encrypt the identity \(ID_i\) using the \(h_1(x \cdot PK_{s,i})\). Besides, the value of \(h_1(x \cdot PK_{s,i})\) changes at every session due to the fresh of \(x\). Anyone who does not know \(x\) or the server’s private key \(SK_{s,i}\) can not know the value of \(h_1(x \cdot PK_{s,i})\). Therefore, our proposed scheme can provide user anonymity and untraceability.

3) **Two-factor security**

Obviously, the adversary cannot forge a legitimate user when he only knows the user’s password. On the other hand, when the smart card is lost or stolen by the adversary \(A\). We assume that \(A\) can obtain the secret parameters in the smart card. \(A\) still cannot guess the correct password, because there exist \(|DPW_{\text{ID}}|/1024\) candidates of the password, where \(|DPW_{\text{ID}}|\) is the space of password. This method is called ‘fuzzy verifier’ [48], [51], which prevents the adversary from obtaining the exact correct password. Therefore, our proposed scheme can provide two-factor security.

4) **Resistance to wrong password login/update attack**

In the proposed scheme, the password verification information \(V = h_3(h_2(K_{s,i}||N_{s,i}||C_i))\) is stored in the smart card, which is designed to check the correctness of password. If the user inputs password \(PW_i\), the verification data \(V\) and \(V’ = h_3(h_2(F_i \oplus h_0(ID_i||PW_i||b_i)||h_0(ID_i||PW_i||b_i)))\) will not be equal. Therefore, our proposed scheme can quickly detect unauthorized login and password update.

5) **User revocation re-registration**

In the proposed scheme, the identities and public keys of users are maintained in the blockchain. Once the smart card is lost or stolen, the user can revoke his/her account, update the revocation status and re-register with a new public key. Due to the revocation status value is recorded in the blockchain, a malicious adversary cannot access the multi-server system using the old public key. In addition, if an adversary wants re-register with the same identity of \(U_i\), he/she must forge a signature \(Sig(K’_{s,i}||ID_i||T_{Rev_i}||REV_{\text{ID}})\) or some personal information. However, we assume that the signature function is unforgeable against adaptive chosen message attack. Similarly, if a semi-trusted server wants to add a fake revocation transaction into blockchain, he/she also must forge a signature. But he/she cannot. Therefore, the revocation and invalid re-registration will be checked.

6) **Known session key security**

In our proposed scheme, the value \(X = x \cdot P\) and \(Y = y \cdot P\) are fresh and different at every session. If the adversary got the session keys in previous sessions, he/she could not compute the current session key without knowing the value of \(x\) or \(y\). Therefore, our scheme can provide known session key security.

7) **Perfect forward security**

In our scheme, the value \(X = x \cdot P\) and \(Y = y \cdot P\) are fresh and different at every session. If the adversary has obtained the private keys of the user and the server, he/she still cannot compute the session key \(Key_i = h_3(ID_i||ID_{s,i}||X||Y||s|||st||x \cdot Y)\) without the value \(x\) or \(y\) in previous sessions. Therefore, our scheme can provide perfect forward security.

8) **Resistance to user impersonation attack**

In our scheme, in order to impersonate \(U_i\), the adversary has to forge a valid message \(T, X, CT\). However, Lemma...
1 shows that it is infeasible due to the DL problem is hard. Therefore, our proposed scheme can resist against user impersonation attack.

9) Resistance to server spoofing attack

Theorem 1 shows that no polynomial adversary can forge a legitimate user’s or a server’s authentication message without the private key of them. In our scheme, the server only has his own private key and does not know other servers’ or users’ private key. Therefore, he cannot spoof any users to other servers.

10) Resistance to reply attack

Our scheme uses the challenge-response mechanism and timestamp mechanism to prevent the replay attack. The random number x and y is fresh and different at every session and the timestamp is used in the first message. Therefore, when the user and the server accept each other, it must be the current session, not the previous session. So, our proposed scheme can avoid the replay attack.

11) Resistance to man-in-the-middle attack

In our scheme, the message transmitted is protected by \( h_1(x \cdot PK_{sj}) \), anyone without \( x \) or \( SK_{sj} \) can not forge legal authentication message. Therefore, our scheme can resist the man-in-the-middle attack.

C. SECURITY COMPARISONS

In this section, we compare security features of our proposed scheme with the prior related schemes [2]–[5]. The results of the comparison are listed in Table 3. From Table 3, we can see that Odelu et al.’s scheme and our proposed scheme are only two schemes who can provide user revocation and re-registration. However, Odelu et al. scheme requires RC to participate in each user authentication phase, which may make RC being a bottleneck of security. Furthermore, our scheme is the only one which is able to resist against various known attacks and fulfill the desirable security features. Therefore, our proposed scheme has better security than previously related schemes.

VI. COMPARISONS

This section first compares the computational costs and communication overheads of our proposed scheme with other related schemes such as He et al.’s scheme [2], [4], [5] and Odelu et al.’s scheme [3]. Because the initialization phase, registration phase, password update phase and user revocation and re-registration phase are not used frequently, we only compare the mutual authentication phase. Then, we will compare the qualitative property of our blockchain-based approach with the traditional registration center-based approach. In order to measure the effectiveness of our proposed scheme, we present the comparison results in different tables.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutual authentication</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>User anonymity and un-traceability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multi-factor security</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resistance to wrong password login/update attack</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>User revocation re-registration</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Known session key security</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Perfect forward security</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resistance to user impersonation attack</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resistance to server spoofing attack</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resistance to reply attack</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resistance to man-in-the-middle attack</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

A. COMPUTATION ANALYSIS

For efficiency analysis, we compare the computation cost of our proposed scheme with the prior related schemes [2]–[5]. Almost all of the operations in our scheme and prior related schemes have appeared in He et al.’s scheme [5]. According to [59], one MAC operation is about as fast as two hash operations in software implementation. In addition, we assume that the running time in RC is as fast as one in the server. As shown in Table 4, we continue to follow the running time of all operations in their scheme. To facilitate analysis, we use the following notations and their running time to measure the computation cost.

<table>
<thead>
<tr>
<th>Running time of operations(millisecond)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The user</td>
</tr>
<tr>
<td>The server</td>
</tr>
<tr>
<td>( T_{tmp} )</td>
</tr>
<tr>
<td>33.582</td>
</tr>
<tr>
<td>5.495</td>
</tr>
<tr>
<td>( T_{bp} )</td>
</tr>
<tr>
<td>32.713</td>
</tr>
<tr>
<td>5.427</td>
</tr>
<tr>
<td>( T_{pm} )</td>
</tr>
<tr>
<td>13.405</td>
</tr>
<tr>
<td>2.165</td>
</tr>
<tr>
<td>( T_{ps} )</td>
</tr>
<tr>
<td>0.081</td>
</tr>
<tr>
<td>0.013</td>
</tr>
<tr>
<td>( T_{sig} )</td>
</tr>
<tr>
<td>13.405</td>
</tr>
<tr>
<td>2.165</td>
</tr>
<tr>
<td>( T_{ver} )</td>
</tr>
<tr>
<td>26.81</td>
</tr>
<tr>
<td>4.33</td>
</tr>
<tr>
<td>( T_{MAC} )</td>
</tr>
<tr>
<td>2.249</td>
</tr>
<tr>
<td>0.339</td>
</tr>
<tr>
<td>( T_h )</td>
</tr>
<tr>
<td>0.056</td>
</tr>
<tr>
<td>0.007</td>
</tr>
<tr>
<td>( T_{MAC} )</td>
</tr>
<tr>
<td>0.112</td>
</tr>
<tr>
<td>0.014</td>
</tr>
</tbody>
</table>

(1) \( T_{tmp} \): The execution time of map-to-point hash function;
(2) \( T_{bp} \): The execution time of bilinear paring operation;
(3) \( T_{pm} \): The execution time of point multiplication operation in \( G \);
(4) \( T_{ps} \): The execution time of point addition operation in \( G \);
(5) \( T_{sig} \): The execution time of signature operation in \( G \);
(6) \( T_{ver} \): The execution time of verification operation in \( G \);
(7) \( T_{exp} \): The execution time of exponentiation operation;
(8) \( T_h \): The execution time of general hash function.
(9) $T_{MAC}$: The execution time of MAC function.

The results of computation cost comparisons are summarized in Table 5. From Table 5, we can see that the computational efficiency of He et al.’s scheme [2] is the most efficient, while they use the heavy bilinear pairings operations and the security of this scheme is based on exponentiation operation. Although the computational efficiency of Odelu et al.’s scheme [3] and He et al.’s scheme [4] are more efficient than our scheme, they achieve at the price of frequent authentication interaction with an online trusted third party. The computational efficiency of our scheme is not the most efficient. But, our scheme provides more security functions.

**TABLE 5. Computation comparisons between our proposed scheme and other related schemes**

<table>
<thead>
<tr>
<th>Scheme</th>
<th>User</th>
<th>Server</th>
<th>RC</th>
<th>Total cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>He [2]</td>
<td>$2T_{pm}$ + $T_{hp}$ + $4T_{exp}$ + $5T_{h}$</td>
<td>31.837ms</td>
<td>\</td>
<td>38.641ms</td>
</tr>
<tr>
<td>Odelu [3]</td>
<td>$3T_{pm}$ + $T_{h}$</td>
<td>40.607ms</td>
<td>\</td>
<td>47.221ms</td>
</tr>
<tr>
<td>He [4]</td>
<td>$3T_{pm}$ + $T_{h}$</td>
<td>40.607ms</td>
<td>\</td>
<td>51.53ms</td>
</tr>
<tr>
<td>He [5]</td>
<td>$T_{msg}$ + $3T_{pm}$ + $2T_{exp}$ + $4T_{h}$</td>
<td>78.519ms</td>
<td>\</td>
<td>90.285ms</td>
</tr>
<tr>
<td>Ours</td>
<td>$T_{msg}$ + $6T_{h} + T_{MAC}$</td>
<td>54.068ms</td>
<td>\</td>
<td>64.928ms</td>
</tr>
</tbody>
</table>

**FIGURE 6. Communication comparisons.**

In this section, we compare communication cost of our proposed scheme with the recent related schemes [2]–[5]. To achieve convincing comparisons, we assume that the bit length of the hash output, the number of block, the identity, the random number, the block size of symmetric encryption/decryption and the timestamp $T$ are 160, 32, 32, 128, 128 and 32 bits, the bit length of the elliptic curve point and exponentiation are 160 and 1024 bits, respectively. Furthermore, we assume that the bit length of signature messages is 320 bits [59]. The results of communication efficiency comparisons are summarized in Table 6.

In the proposed scheme, the first messages $\{T, X, CT\}$ require $(32+320+(32+32+32+160))=608$ bits, and the second messages $\{Y, V\}$ require $320 +160=480$ bits. Adding the two values, the total communication cost in the authentication phase of our scheme is 1568 bits. Similarly, the total communication cost of the other related schemes can be computed in Table 6.

From comparison in Table 6 and Fig. 6, we conclude that our proposed scheme requires the least rounds of message exchange. Furthermore, the proposed scheme is the most efficient in communication overhead.

**C. QUALITATIVE COMPARISONS**

The analysis of qualitative property includes single registration, using online RC, resistance to single-point failure, search times and storage. In Table 7, we compare the qualitative property of our blockchain-based approach with the traditional registration center-based (RC-based) approach. Here, we divide RC-based approach into two categories, namely no online RC-Based, and online RC-based, according to whether with the help of online RC in traditional RC-based approach.

**TABLE 6. Communication comparisons between our proposed scheme and other related schemes**

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Rounds of message exchange</th>
<th>Number of bits required</th>
</tr>
</thead>
<tbody>
<tr>
<td>He [2]</td>
<td>3</td>
<td>3072 bits</td>
</tr>
<tr>
<td>Odelu [3]</td>
<td>5</td>
<td>2944 bits</td>
</tr>
<tr>
<td>He [4]</td>
<td>5</td>
<td>3520 bits</td>
</tr>
<tr>
<td>He [5]</td>
<td>4</td>
<td>5296 bits</td>
</tr>
<tr>
<td>Ours</td>
<td>2</td>
<td>1088 bits</td>
</tr>
</tbody>
</table>

**TABLE 7. Qualitative comparisons between our blockchain-based approach and RC-based approach**

<table>
<thead>
<tr>
<th>Qualitative property</th>
<th>No online RC-Based</th>
<th>Online RC-based</th>
<th>Blockchain-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single registration</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Using online RC</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Resistance to single-point failure</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Search times</td>
<td>$L_{req}$</td>
<td>$L_{req}$</td>
<td>$L_{bc}$</td>
</tr>
<tr>
<td>Storage</td>
<td>The blacklist</td>
<td>Nothing</td>
<td>The blockchain</td>
</tr>
</tbody>
</table>

The qualitative property of single registration represents whether users register only once. Obviously, the traditional RC-based approach enables users to register once. In our blockchain-based approach, if a user wants to access a server, he/she requires registration only once with any one of the servers in the multi-server system. After the user’s information like the identity and public key have been recorded in the blockchain, the user can access the multi-server system.
Besides, the traditional RC-based approach belongs to the centralized administration. All new users have to register with the only RC. Our blockchain-based approach can avoid it. A new user can select the closest server in the multi-server system to register, which may be more suitable for practical application.

The using online RC denotes whether the authentication phase between the user and the server needs the help of online RC. According to the above definition, no online RC-Based approach has not online RC in the authentication phase, while online RC-Based approach needs. For our blockchain-based approach, when the server authenticates a user, he/she only verifies the user’s signature through searching for the public key in the blockchain. Generally, the blockchain is stored in the own side of the server, there’s no need for a trust third party to take part in.

As already stated earlier in this document, the traditional RC-based approach has the problem of single-point failure. All users’ data, including users’ identities, public keys, possible secret parameters, blacklist, etc., are stored in the single RC. If the single RC attacked or suffered from natural disasters, the whole data will be in danger. To address this issue, we introduce blockchain technology into the multi-server authentication scheme. In our proposed blockchain-based scheme, users’ data are recorded in the blockchain, which is decentralized stored in every server in the multi-server system. Once registered on the blockchain, users’ data can not be unforgeability.

In practice, according to previous no online RC schemes, like [10], the server has to search for the blacklist to check whether the corresponding user is revoked in the authentication phase. Meanwhile, in previous online RC schemes, like [3], the server has to search for the user information table to check the revocation status. Similarly, in our proposed scheme, the server must search for the blockchain to check user’s revocation status. It is obvious that all of multi-server authentication schemes which have considered user revocation have to search for the revocation status. The efficiency of search operations is determined by the length of blacklist, table or blockchain. In general, the blacklist includes all the revoked users, the user information table contains all registered users, and our blockchain involves all registered and revoked users. We let $L_{rev}$, $L_{reg}$ and $L_{bc}$ denote the length of blacklist, user information table and blockchain, respectively. Since the same user can revoke multiple times, $L_{bc} \geq L_{rev} + L_{reg}$. Obviously, the search efficiency of our blockchain-based is the lowest approach.

The qualitative property of storage means what the server-side stores to achieve user revocation. As analyzed above, the server stores blacklist to check whether the corresponding user is revoked for no online RC schemes. For online RC schemes, since the online RC participates in every authentication phase, the server can query revocation status from RC. So the server does not need to store any user information or revoke information. In our proposed scheme, the server must store the whole blockchain to check user’s revocation status. Obviously, the storage cost of our blockchain-based approach is the highest.

From comparison in Table 7, it can be concluded that our blockchain-based scheme solves the problem of single-point failure at the price of storage and search efficiency. In practice, there may be some applications suitable for our blockchain-based approach. For example, in mobile cloud computing environment [5], [11], the service provider has the capability in powerful computing and massive data storage. Thus, it pay more attention to security and privacy. Our blockchain-based multi-server authentication scheme may be more suit for such environment.

VII. CONCLUSION

In this paper, we propose a blockchain-based privacy-aware authentication scheme with efficient revocation for the multi-server system, which provides various security requirements like mutual authentication, user anonymity, perfect forward security. Besides, in comparison with recently related schemes, the proposed scheme solve the problem of a single registration center. The security analysis demonstrates that our scheme is secure the random oracle model. Performance analysis shows that the proposed scheme has higher communication efficient, which may be suitable to deploy in practice for the multi-server system.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and the Associate Editor for providing constructive and generous feedback.

REFERENCES