electronics

MDPI

Article

Distributed and Federated Authentication Schemes Based on Updatable Smart Contracts

Keunok Kim [1], Jihyeon Ryu [2], Hakjun Lee [3], Youngsook Lee [4] and Dongho Won [5,*]

[1] Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon-si 16419, Republic of Korea
[2] School of Computer and Information Engineering, Kwangwoon University, Seoul-si 01897, Republic of Korea
[3] Department of Computer Engineering, Kyungnam University, Changwon-si 51767, Republic of Korea
[4] Department of Computer Information Security, Howon University, 64 Impi-myeon, Howondae 3-gil, Gunsan-si 54058, Republic of Korea
[5] Department of Computer Engineering, Sungkyunkwan University, Suwon-si 16419, Republic of Korea
* Correspondence: dhwon@security.re.kr

Abstract: Federated authentication, such as Google ID, enables users to conveniently access multiple websites using a single login credential. Despite this convenience, securing federated authentication services requires addressing a single point of failure, which can result from using a centralized authentication server. In addition, because the same login credentials are used, anonymity and protection against user impersonation attacks must be ensured. Recently, researchers introduced distributed authentication schemes based on blockchains and smart contracts (SCs) for systems that require high availability and reliability. Data on a blockchain are immutable, and deployed SCs cannot be changed or tampered with. Nonetheless, updates may be necessary to fix programming bugs or modify business logic. Recently, methods for updating SCs to address these issues have been investigated. Therefore, this study proposes a distributed and federated authentication scheme that uses SCs to overcome a single point of failure. Additionally, an updatable SC is designed to fix programming bugs, add to the function of an SC, or modify business logic. ProVerif, which is a widely known cryptographic protocol verification tool, confirms that the proposed scheme can provide protection against various security threats, such as single point of failure, user impersonation attacks, and user anonymity, which is vital in federated authentication services. In addition, the proposed scheme exhibits a performance improvement of 71% compared with other related schemes.

Keywords: federated authentication; smart contracts; updatable smart contracts

Citation: Kim, K.; Ryu, J.; Lee, H.; Lee, Y.; Won, D. Distributed and Federated Authentication Schemes Based on Updatable Smart Contracts. *Electronics* **2023**, *12*, 1217. https://doi.org/10.3390/electronics12051217

Academic Editors: Huy Kang Kim and Younho Lee

Received: 1 February 2023
Revised: 18 February 2023
Accepted: 21 February 2023
Published: 3 March 2023

1. Introduction

The rapid development of electronic transactions has significantly improved user privacy and transaction safety. To secure electronic transactions, a user must remember a long password or supply a digital certificate or a one-time password. Most internet sites employ a user authentication scheme based on an isolated identity architecture (isolated authentication scheme), in which a service provider (SP) (providing services) and an identity provider (IdP) (managing login credentials) are combined into a single server [1]. For example, in an isolated authentication scheme based on a password, a user requests a login from an SP by submitting the user's identity (ID) and password (PW), which are pre-registered with an SP serving as an IdP. Upon receiving the user ID and PW, the SP verifies whether they are in the password table and responds to the user for the authentication result, as shown in Figure 1. Isolated authentication is one of the simplest authentication schemes and is widely used in numerous internet services.

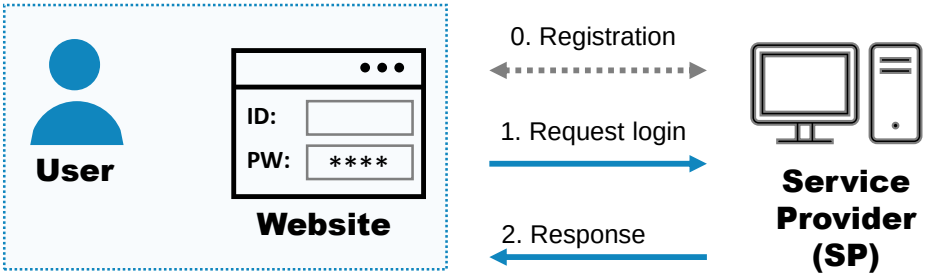

Figure 1. Isolated authentication scheme.

However, this scheme is inconvenient because it requires users to recall and manage their login credentials (for example, user ID, PW, and certificate) for each website. In addition, security threats may occur when managing login credentials for multiple websites [2].

In contrast, a user authentication scheme based on a federated identity architecture (federated authentication scheme), such as Google ID, can authenticate users on multiple websites using a single login credential that is pre-registered at an IdP, such as Google. In this scheme, an SP and IdP can be separated and login credentials can be stored at an IdP rather than with an SP. Furthermore, an SP and an IdP can be classified under the same group [1,3,4].

If a user requests a login from an SP, then the SP redirects the user to an IdP website with which the user has registered. The user enters their login credentials on the IdP website. Upon receipt of the user's login credentials, the IdP verifies whether the login credentials are in the login table and then responds to the SP with the authentication result, as illustrated in Figure 2.

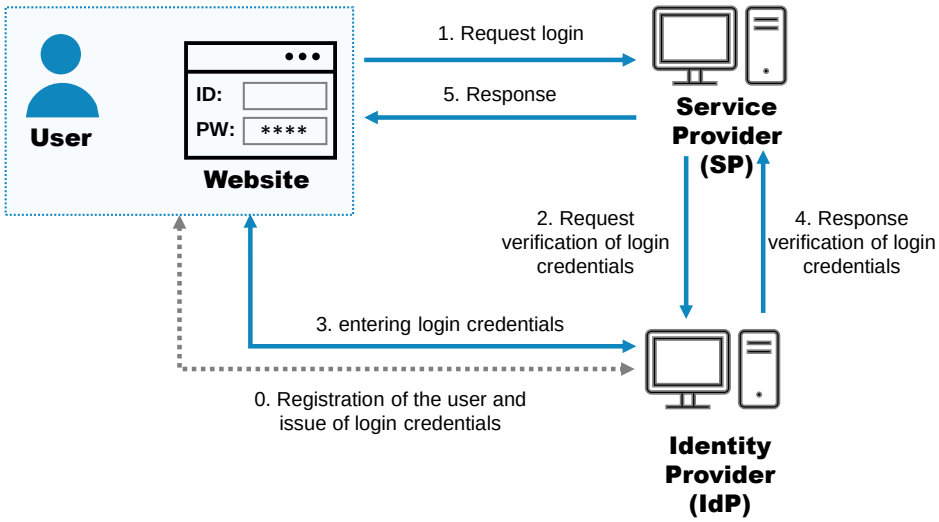

Figure 2. Federated authentication scheme.

A federated authentication scheme improves user convenience because it does not require managing login credentials for each website. However, to maintain the security of a federated authentication scheme, the following security principles are required [5,6]:

1. Avoiding a single point-of-failure: even if the IdP server fails, the SP using the corresponding IdP should be available.
2. Ensure anonymity: because the same login credentials are used on multiple websites, anonymity must be guaranteed.
3. Protection against impersonation attacks: login credentials must be protected in federated systems, similar to all other access-protected systems. This is particularly

important in federated authentication because if the login credentials are exposed, then user impersonation attacks can occur on multiple websites.

Research [7–12] has focused on the high availability and reliability of such systems. In 2022, Xue et al. [13] proposed a distributed authentication scheme that could resist a single point-of-failure using smart contracts (SCs) and investigated the roaming services. In this study, to avoid a single point-of-failure in federated authentication, Xue et al.'s scheme was applied to the federated authentication concept. The vulnerabilities of the scheme of Xue et al. [13] were analyzed and an improved scheme addressing these vulnerabilities was proposed. Moreover, the SCs proposed by Xue et al. [13] cannot maintain previously registered data if they are updated for programming bug fixes or authentication functions that update SCs. Therefore, an updatable SC was designed to address these issues and maintain registered data after updating the SCs. Finally, security and performance analyses were conducted on the proposed scheme, and its safety and performance were compared with those of other state-of-the-art schemes.

1.1. Our Contribution

This study proposed a distributed and federated authentication scheme based on updatable smart contracts by improving Xue et al.'s [13] scheme. The contributions of this study are summarized as follows.

1. A scheme was proposed in which each node has a copy of the chain owing to the property of the blockchain to resist a single point of failure.
2. The scheme of Xue et al. [13], which allows impersonation attacks and does not guarantee anonymity, was improved. To counter impersonation attacks, verification logic was added to determine whether the owner of the public key is legitimate during the authentication process. In addition to ensuring user anonymity, the proposed scheme does not directly transmit user ID during authentication.
3. An updatable SC was designed to support the programming of bug fixes or authentication functions to update an SC and maintain registered data after updating the smart contract.

1.2. Paper Structure

The remainder of this paper is organized as follows. Section 2 presents the related studies. Section 3 provides a preliminary overview of the basic elements used in this study. Section 4 provides a review of Xue et al.'s scheme and Section 5 presents an analysis of the security vulnerabilities of Xue et al.'s scheme. Section 6 presents the system model, assumptions, and security requirements, and Section 7 presents the proposed scheme. Sections 8 and 9 present security and performance analyses, respectively. Section 10 provides a discussion and limitations , and Section 11 concludes the study.

2. Related Studies

Blockchain-based decentralized authentication schemes have been proposed to satisfy the high availability and reliability requirements of various systems.

Blockchain is a crucial emerging technology in e-health systems for the management of sensitive medical data. The application of blockchain technology provides stable real-time services and enables the distributed storage of medical data. Xiang et al. [7] proposed a decentralized authentication and access control protocol for blockchain-based e-health systems; this protocol uses BAN logic to validate the reliability of the system security protection. Cheng et al. [8] proposed a medical data sharing scheme based on blockchain that realizes medical treatment data sharing and satisfies various security requirements during the authentication phase. Cheng et al. [8] claimed that their scheme provides mutual authentication, anonymity, untraceability, session key security, and perfect forward secrecy. However, Xiang et al. [7] reported that their scheme does not provide mutual authentication without RA. Zhang et al. [9] proposed security and privacy requirements for healthcare blockchains.

For mobile vehicular networks, researchers [10–12] have proposed several technologies combined with blockchain technology to ensure secure and real-time communication. Recently, Xue et al. [13] proposed a decentralized fraudproof roaming authentication framework based on the blockchain using SCs. In contrast, the present study focused on using SCs to overcome the single point-of-failure of a centralized system. Notably, various security threats are encountered when applying Xue et al.'s [13] scheme to federated authentication schemes. Although Xue et al.'s [13] scheme addresses the important single point-of-failure in federated authentication, it presents vulnerabilities to anonymity and user impersonation attacks.

Data on a blockchain are immutable, and deployed SCs cannot be changed or tampered with. However, in practice, updates are performed to fix programming bugs, add to the function of an SC, or modify the business logic. Recently, studies have been conducted to identify methods for updating SCs to address the abovementioned functions [14–18]. Zheng et al. [14] introduced various smart contract development platforms in 2020. Representatively, they mentioned Ethereum, Hyperledger fabric, Corda, Stellar, Rootstock, and EOS and compared these platforms based on the execution environment, supported language, data model, consensus algorithms, permissions, and applications of SCs. In 2022, a contract upgrade was implemented using a proxy, delegator, and dispatcher patterns [15]. Additionally, design patterns for smart contracts in Ethereum have been studied [16].

In addition, the applications of SCs have been studied [17]. Shao et al. [17] developed an LSC, which is an anomaly detection mechanism, using SCs. The LSC was developed using Solidity to enable log anomaly detection for log systems running on Ethereum [17]. Górski et al. [18] introduced an SC design and implementation patterns using Java; implemented patterns increased source code reusability and foster testing. This method could reduce the time taken for SC validation [18]. EL PASSO [19] was proposed based on a zero-knowledge algorithm for prohibiting both IdP and RP to determine a user trace. However, this method relies on a single ID provider and can forge IDs if a malicious provider exists.

In this study, a distributed and federated authentication scheme was proposed by improving the scheme of Xue et al. [13] by overcoming anonymity and user-impersonation attacks. Additionally, an updatable SC was designed to support the programming of bug fixes and the authentication function for updating an SC, and to maintain the user revocation status after updating the SC.

3. Preliminaries

This section reviews the background information on updatable SCs and elliptic curve digital signature algorithms.

3.1. Updatable Smart Contracts

Szabo first proposed SCs in 1994 before the introduction of blockchain [20]. However, with the advent of the Ethereum blockchain, alongside the development and application of SCs accelerated [21]. SCs based on blockchain are self-executed programs executed on a decentralized blockchain network. They enable contractual terms of an agreement to be enforced automatically without the intervention of a central authority [14].

The building and execution processes for general SCs are illustrated in Figure 3. The user sends a bytecode that compiles SCs to the blockchain and saves the address of the SC. A person wishing to create an SC can do so using the address of the SC.

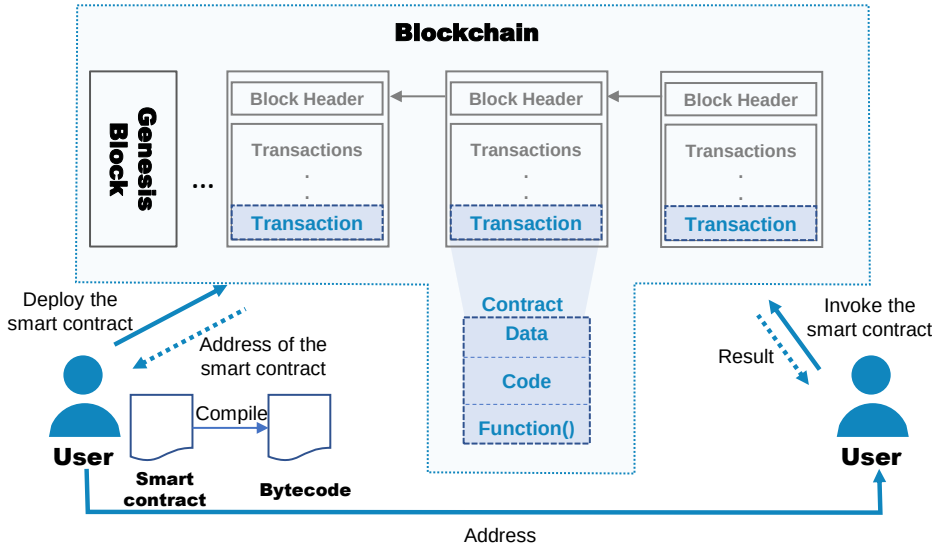

Figure 3. Building and executing processes of general smart contracts.

These processes are performed in an immutable, transparent, and completely secure manner because an SC is deployed in the blockchain network. Therefore, deployed SCs cannot be changed or tampered with.

However, updates may be performed by fixing programming bugs, adding the function of an SC, or changing business logic. If a new SC is created, deployed, and used over the old one, the previously stored data may be unusable. In addition, distributing the address of the new SC to the user is a convenient approach.

For example, if the saved data and business logic are configured as an SC, as shown in Figure 4, the saved data cannot be maintained when the SC is updated.

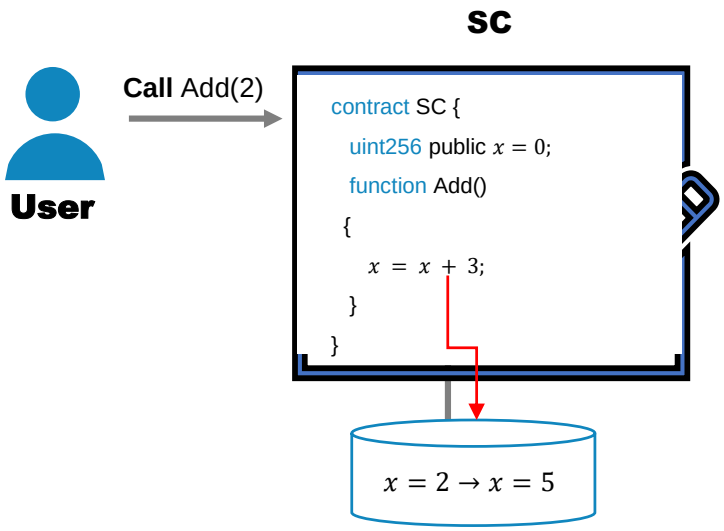

Figure 4. SC that configures data and business logic into the same SC.

To address this, as shown in Figure 5, if SC1 stores data and SC2 contains business logic, and SC1 calls SC2, then the operation result of SC2 is not reflected in SC1.

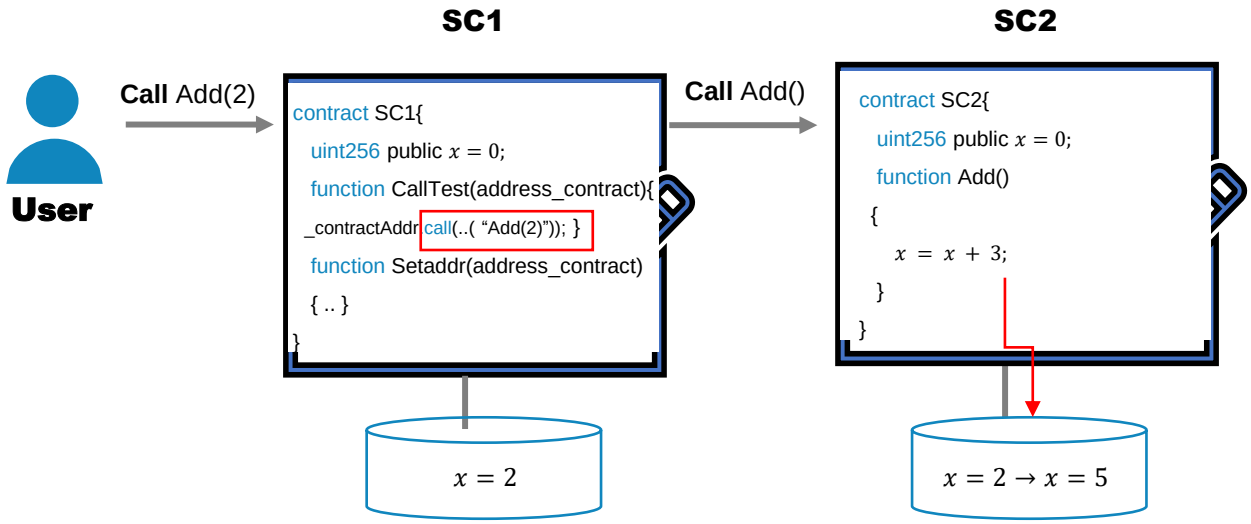

Figure 5. Concept for SC1 calling SC2.

The updatable smart contract comprises SCs such as SC1 and SC2, as shown in Figure 6, wherein SC1 stores data and SC2 contains business logic. If SC1 calls SC2 (a delegate call), the operational result of SC2 is reflected in SC1.

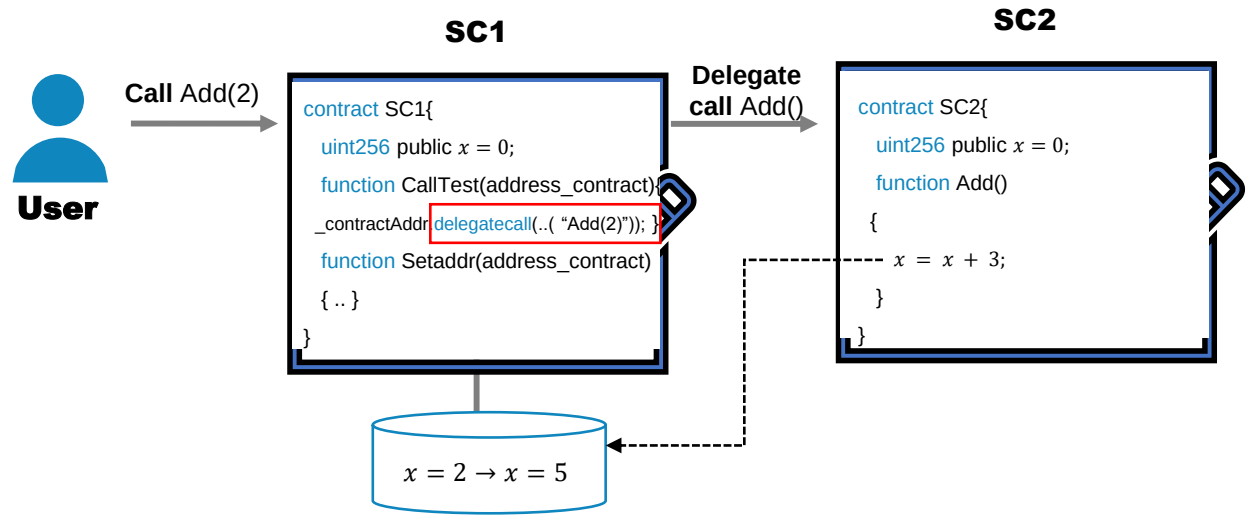

Figure 6. Concept for SC1 delegate calling SC2.

When an SC update is required for a program change, we change the program and deploy a new SC2. After that, in SC1, we can call a function to set the new address of SC2. Finally, the saved data are maintained even if the new SC2 is updated. The details are shown in Figure 7.

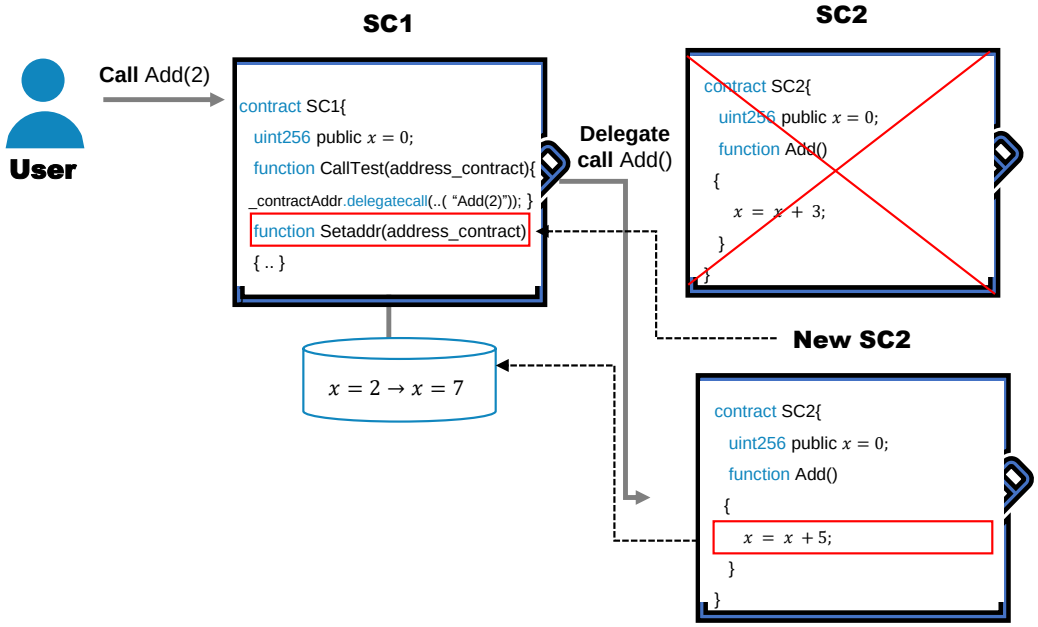

Figure 7. Concept of updatable SCs.

3.2. Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA is a standardized digital signature algorithm [22,23] that uses elliptic curve cryptography (ECC). This algorithm comprises three processes [24,25]:

- Key Generation: ECC comprises two types of keys: public and private. Let E be an elliptic curve over a finite field, F_q, of characteristic p and base point $G \in E(F_q)$.
  1. Select a random integer d in the interval $[1, n-1]$, where n is a sufficiently large prime.
  2. Compute $Q = dG$.

  Therefore, d is the private key and Q is the public key.
- Signature Generation: the signer performs the following steps to generate a digital signature.
  1. Select a random number k in the interval $[1, n-1]$.
  2. Compute $(x, y) = kG$ and $r = x \ mod \ n$. If $r = 0$, return to step 1.
  3. Compute $e = h(M)$ and $s = k^{-1}(e + rd) \ mod \ n$, where M represents the data to be signed. If $s = 0$, return to step 1.

  Thus, pair (r, s) is a signature for message M.
- Signature Verification: the verifier performs the following steps to verify the digital signature.
  1. Verify that r and s are integers in the interval $[1, n-1]$. Otherwise, the signature is considered invalid.
  2. Compute $e = h(M)$, $u_1 = es^{-1} \ mod \ n$, and $u_2 = rs^{-1} \ mod \ n$.
  3. Compute $(x', y') = u_1 G + u_2 Q$ and $v = x' \ mod \ n$.
  4. Verify that $v = r$.

  Thus, if $v = r$, then the signature is valid; otherwise, it is invalid.

4. Review of Xue et al.'s Scheme

This section describes Xue et al.'s distributed authentication scheme [13]. In this study, blockchain-based SCs were used for distributed authentication. The system comprises multiple domains, each of which contain access points (APs) and a network control center (NCC). For example, a home network registering a user comprises home APs and a home NCC (HNCC), whereas a foreign network visited by a user comprises foreign APs (FAPs) and a foreign NCC. Each AP and NCC is connected to a blockchain. An NCC creates and

maintains SCs for authentication and deploys them to a blockchain. Additionally, it issues and manages login credentials for users/APs. In addition, APs authenticate users through an SC issued by the NCC. Before authentication, the user must register with the HNCC and issue login credentials. If the user wishes to access a foreign network, the user requests authentication from the FAP. Upon receiving the user authentication request, the FAP authenticates the user through the SCs and provides network access services. The system model of Xue et al.'s scheme is shown in Figure 8.

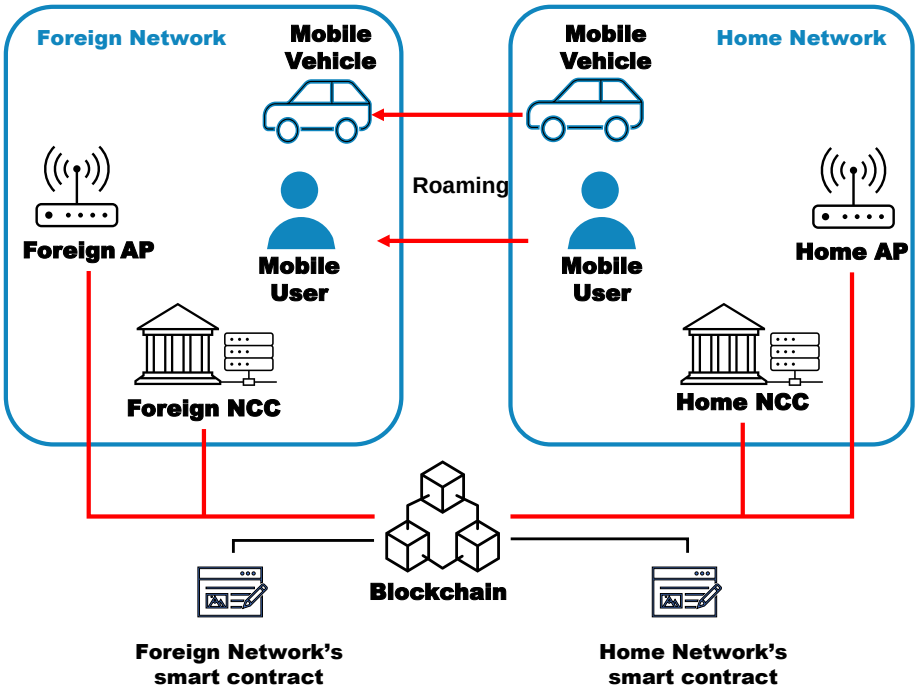

Figure 8. System model of Xue et al.'s scheme.

Xue et al.'s scheme comprises three types of smart contracts: the main contract (MC), the authentication contract (AC), and the revocation contract (RC), as shown in Figure 9. The MC has the address of its own AC and a table that maps the other NCCs to their MC addresses. The AC has an actual authentication function and the address of its own RC, where the RC manages the user's revocation status.

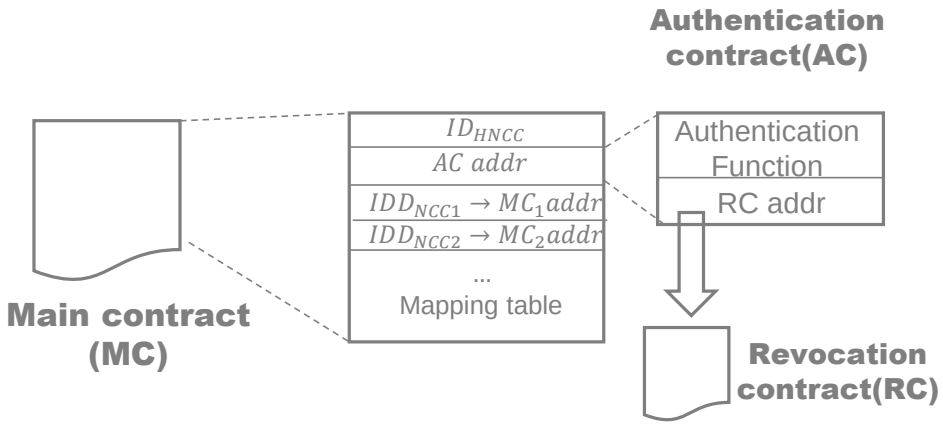

Figure 9. Xue et al.'s SCs.

This scheme comprises four phases: user and AP registration, user authentication, dynamic user enrollment and revocation, and the establishment of a roaming partner-

ship. The billing phase in this study was excluded because it is not directly related to user authentication.

4.1. Registration Phase

In the registration step, the user and APs register with an NCC. Before a user accesses a foreign network, a user must register with an HNCC. Before the registration phase, the NCC generates its own ECDSA key pair, i.e., SK_{NCC} and PK_{NCC}, and deploys smart contracts. The details of the user registration phase are as follows:

1. The user sends their identity, ID_U, to the HNCC through a secure channel.
2. Upon receipt of the user's identity, ID_U, the HNCC generates an ECDSA private key, SK_U, an ECDSA public key, PK_U, and a login credential, $CR_U = EC.Sign(SK_{NCC}, ID_U||PK_U)$, for the user. Subsequently, the HNCC sends ID_U, ID_{NCC}, SK_U, PK_U, CR_U, and $MADDR$ to the user through a secure channel. ID_{NCC} is the HNCC identity and $MADDR$ is the MC address of the HNCC.

The APs must be registered with their HNCC similarly to the user registration. The details of the user and AP registration phases are presented in Figure 10.

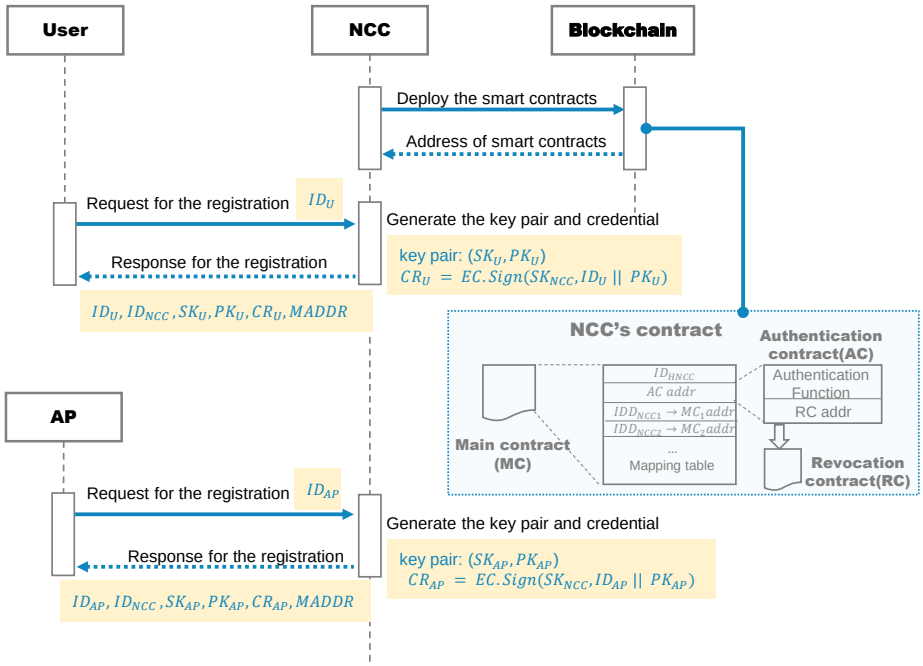

Figure 10. Registration phase of Xue et al.'s scheme.

4.2. User Authentication Phase

In this authentication step, a user requests access to a foreign network by submitting a request message, M_U, to a FAP. Upon receipt of the user's request message, M_U, the FAP verifies the user's request message. If the request message is valid, then the FAP sends the response message, M_{AP}, to the user. The details of the user authentication phase are as follows:

1. The user selects a random number r_U and generates timestamps ts_U, $R_U = r_U \cdot G$, $T_U = h(R_U||ts_U)$, and $V_U = EC.sign(SK_U, T_U)$. Finally, the user generates a request message as $M_U = ID_U||\ PK_U||\ CR_U||\ R_U||\ ID_{NCC}||\ V_U||\ ts_U$ and sends M_U to the FAP.
2. Upon receiving the user's request message, M_U, the FAP verifies whether ts_U is within an allowable range. If ts_U is valid, then the AP calculates $T'_U = h(R_U||ts_U)$ and verifies whether T_U is equal to T'_U. If ts_U or T_U is invalid, the FAP stops the authentication phase. If the FAP does not generate the main contract's input as $TX_{in} = ID_{NCC}||\ T'_U||\ ID_U||\ PK_U||CR_U||\ V_U$, it invokes the MC and sends TX_{in} for user authentication.

3. If ID_{NCC} points to its own identity, the MC calls its AC to verify whether CR_U and V_U are valid. The AC verifies CR_U as $EC.Verify(PK_{NCC}, CR_U)$ and V_U as $EC.Verify(PK_U, V_U)$, and performs verification through the RC to confirm whether the user is revoked. If CR_U, V_U, or the response of the RC is invalid, the AC stops this authentication phase and sends a failure response. Otherwise, the AC sends a true response to the FAP. If ID_{NCC} points to another NCC, then the MC searches for ID_{NCC} in the address mapping table. If ID_{NCC} exists in the mapping table, then the MC calls the corresponding MC to verify TX_{in} and respond to the resulting receipt by the corresponding MC to the FAP.
4. Upon receiving the true response of the MC, the FAP selects a random number r_{AP} and generates ts_{AP}, $R_{AP} = r_{AP} \cdot G$, $T_{AP} = h(R_{AP}||ts_{AP})$, and $V_{AP} = EC.sign(SK_{AP}, T_{AP})$, where SK_{AP} is the ECDSA private key of the AP. Finally, the FAP generates a response message as $M_{AP} = ID_{AP}||\ PK_{AP}||\ CR_{AP}||\ R_{AP}||\ ID_{NCC}||\ V_{AP}||\ ts_{AP}$ and sends it to the user. Subsequently, the FAP computes the session key $SK = r_{AP} \cdot R_U$.
5. Upon receiving the FAP response message, M_{AP}, the user process is identical to that of the FAP. Details regarding the user authentication phase are presented in Figure 11.

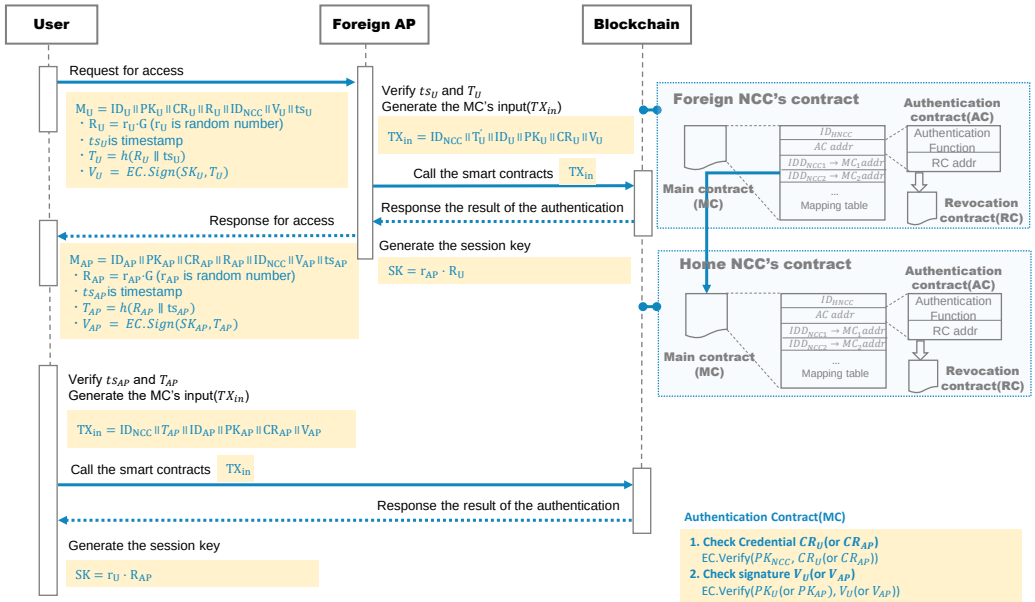

Figure 11. Authentication phase of Xue et al.'s scheme.

4.3. Dynamic User Enrollment and Revocation

In Xue et al.'s scheme, if a user wishes to enroll in the system, then the user must register with the HNCC. The user can revocate joining this system because of issues such as key loss and illegal usage. If the user requests the HNCC to revocate, then the HNCC updates the user's identity, ID_U, in the RC variable.

4.4. Establishment of Roaming Partnership

Different network operators can enroll in or cancel roaming partnerships. If an NCC wishes to enroll or cancel different network operators, it must update or erase the corresponding address mapping table in its MC.

5. Analysis of Xue et al.'s Scheme

This section analyzes the security vulnerabilities of Xue et al.'s scheme, which allows impersonation attacks (i.e., attacks that do not guarantee anonymity). In addition, the disadvantages of smart contracts are also discussed.

5.1. User Impersonation Attacks

Assume that the attacker can register with the same HNCC as the user and steal the user's request message, M_U, during the user authentication phase. The attacker possesses their own data, ID_A, CU_A, PK_A, SK_A, and stolen user data, $M_U = ID_U || PK_U || CR_U || R_U || ID_{NCC} || V_U || ts_U$. Details regarding user impersonation attacks are described as follows:

1. An attacker generates a request message as $M_A = ID_U || PK_A || CR_A || R_A || ID_{NCC} || V_A || ts_A$ and sends it to the FAP pretending to be the user.
2. Upon receiving the attacker's request message, M_A, the FAP verifies ts_A and T_A and generates the main contract input TX_in. Subsequently, the FAP sends TX_in to the MC, which calls an AC for user authentication.
3. The AC verifies CR_A as $EC.Verify(PK_{NCC}, CR_A)$ and V_A as $EC.Verify(PK_A, V_A)$. The verification results for CR_A and V_A are valid because the AC does not verify whether PK_A corresponds to the user.

Consequently, an attacker can impersonate the user. In addition, the attacker who receives the response result can obtain the session key between the FAPs by pretending to be the user.

5.2. Anonymity

Xue et al.'s scheme does not guarantee user anonymity because every transaction includes the user's identity, ID_U.

5.3. Disadvantages of Smart Contracts

Xue et al.'s [13] scheme does not define how to maintain registered data when updating the AC for error correction or the authentication function. Consequently, if the AC is updated in the scheme proposed by Xue et al., all users must be re-registered.

6. System Model and Security Requirements

This section introduces the system model and security requirements. In particular, a system model for federated authentication schemes is introduced, along with the security requirements that the scheme satisfies.

6.1. System Model

The system model of the federated authentication scheme comprises IdPs, SPs, and users. The system model of the proposed scheme is shown in Figure 12.

1. SCs are of two types, namely, the MC and the AC. The MC and AC correspond to proxy contracts and logic contracts, respectively, in [26]. The MC possesses registered data, i.e., identity, certificate, and revocation status; however, it does not possess authentication logic, whereas an AC possesses only authentication functions. IdPs create, manage, and deploy SCs. Additionally, an SP can serve as an IdP by creating, managing, and deploying SCs. Each user, SP, and IdP interfaces with the MC, and the MC authenticates through the AC. To support AC updating, the IdP must implement the SC such that the MC calls the AC using a delegate call. The design of the SCs in the proposed scheme is illustrated in Figure 13.
2. Each user and SP can register with an IdP. If the user or SP requests registration, the IdP registers and stores the registration data in their MC. Subsequently, the IdP issues the login credentials and the address of the MC to the requester.
3. Users can log into the SP website using their login credentials. If the user requests a login, the SP requests authentication from the IdP's MC. The MC authenticates the user through its own AC and then responds to the SP with the result; the SP responds to the user. Subsequently, the user requests authentication from the IdP's MC for mutual authentication. After authentication is completed through the AC in the aforementioned manner, the user and SP share a session key and communicate securely using the session key.

4. Each user and the SP can revocate their registration with the IdP. If the user or SP requests revocation, the IdP requests revocation to its own MC. Subsequently, the MC revocates the registration after completing authentication through its own AC.

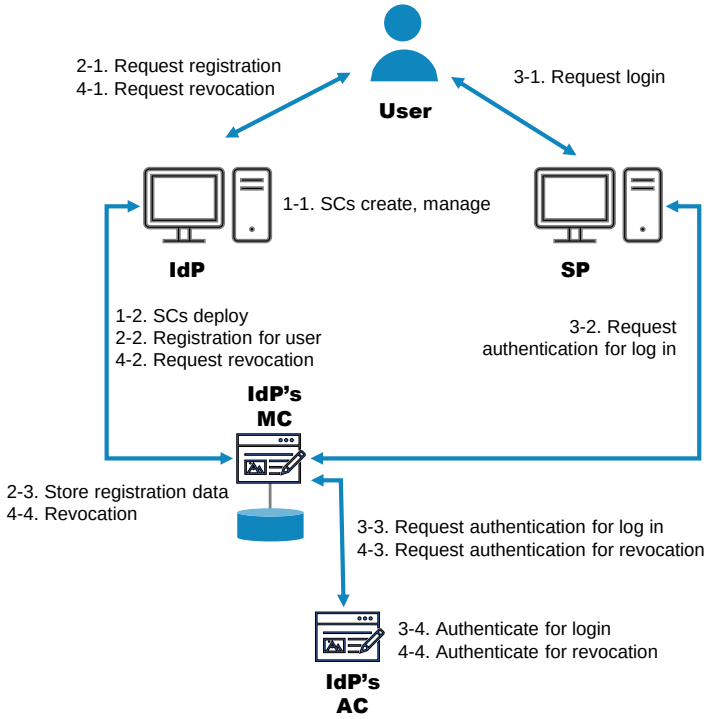

Figure 12. System model of the proposed scheme.

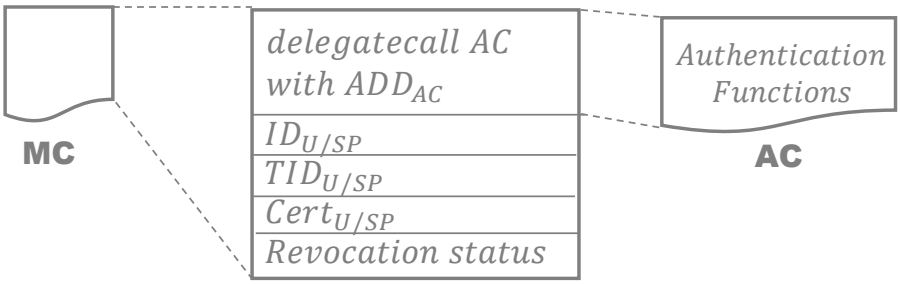

Figure 13. Design of SCs for the proposed scheme.

6.2. Security Requirements

The proposed scheme should satisfy the following security requirements [5,6]:

1. Single point-of-failure: if the IdP system fails, then user authentication should be provided continually.
2. User anonymity: user anonymity must be guaranteed in the authentication phase via a public channel.
3. User impersonation attack: even if an attacker eavesdrops on messages between the user and the SP transmitted via a public channel, the user impersonation attack must be protected.

6.3. Assumption

This paper assumes that the private key is stored securely in a tamper-proof smart card and protected by the user's password. In the event that the password is entered incorrectly more than a set number of times, for example, five times, access to the private key will

be blocked. To regain access to the service, the user must complete the re-registration process, thereby ensuring that only authorized users can access the private key. This security measure effectively mitigates the risk of unauthorized access and provides a robust solution for secure private key storage.

7. Proposed Scheme

This section proposes a distributed and federated authentication scheme based on updatable SCs that comprise six phases: initialization, user and SP registration, mutual authentication, user revocation, service provider enrollment revocation, and SC updating. The notation used in the proposed scheme are listed in Table 1.

Table 1. Notation

Notations and Abbreviations	Description
ID_U, ID_{SP}, ID_{IdP}	Identity of the user, SP, and IdP, respectively
TID_U, TID_{SP}	Temporary identity of the user and SP, respectively
PW_U, PW_{SP}	Password of the user and the SP, respectively
SK_U, SK_{SP}, SK_{IdP}	ECDSA private key of the user, SP, and IdP, respectively
ESK_U, ESK_{SP}	Encrypted private key using a password of the user and the SP, respectively
PK_U, PK_{SP}, PK_{IdP}	ECDSA public key of the user, SP, and IdP, respectively
$Cert_U$, $Cert_{SP}$	Certification of the user and SP, respectively
ADD_{IdP}	Address for an IdP's MC
ADD_{AC}	Address for an AC
$Sig(.)$	ECDSA signature generation function
$Verify(.)$	ECDSA signature verification function
$h(.)$	Hash function
$\|\|$	Concatenation operator
$\oplus$	Bit wise XOR
SK	Session key
SP	Service provider
IdP	Identity provider
MC, AC, SC	Main contract, authentication contract, and smart contract, respectively

7.1. Initialization

During system initialization, each IdP creates and deploys SCs in the blockchain and stores the addresses of the SCs. The detailed content of the SCs is shown in Figure 13. The IdP chooses an elliptic curve, E, defined over a finite field, F_q, of characteristic p and base point $G \in E(F_q)$, and securely generates the ECDSA's key pair, SK_{IdP} and PK_{IdP}, and the IdP's identity ID_{IdP}.

7.2. Registration Phase

In the registration phase, the user and SP register with the IdP. First, the user registration process is as follows:

1. The user generates a PW_U and an ID_U.
2. The user sends the ID_U to the IdP through a secure channel for registration.
3. Upon receiving the ID_U, the IdP generates the ECDSA key pair, SK_U and PK_U, and computes certificate $Cert_U = PK_U||ID_U||ID_{IdP}||Sig(SK_{IdP}, PK_U||ID_U||ID_{IdP})$.
4. The IdP stores ID_U and $Cert_U$ in its own MC, as presented in Table 2.
5. The IdP sends $SK_U||ADD_{IdP}$ via a secure channel to the user.
6. Upon receiving $SK_U||ADD_{IdP}$, the user stores ID_U, $ESK_U = SK_U \oplus h(PW_U)$, and ADD_{IdP}, as listed in Table 3.

Table 2. Data stored in the IdP's MC.

Data	Description
$ID_{U/SP}$	Identity of user or SP
$TID_{U/SP}$	Temporary identity of user or SP Initial value = $ID_{U/SP}$
$Cert_{U/SP}$	User or SP certificate.
Revocation status	Revocation status of user or SP Initial value = false

Table 3. Data stored in the user or SP.

Data	Description
$TID_{U/SP}$	Temporary identity of user or SP Initial value = $ID_{U/SP}$
$ESK_{U/SP}$	Encrypted private key of user or SP
ADD_{IdP}	Address of the IdP

Second, the registration phase of the SP is identical to that of the users. The SP registration process is as follows:

1. The SP generates PW_{SP} and ID_{SP}.
2. The SP sends ID_{SP} to the IdP through a secure channel for registration.
3. Upon receiving ID_{SP}, the IdP generates the ECDSA key pair, SK_{SP} and PK_{SP}, and computes the certificate $Cert_{SP} = PK_{SP}||ID_{SP}||ID_{IdP}||Sig(SK_{IdP}, PK_{SP}||ID_{SP}||ID_{IdP})$.
4. The IdP stores ID_{SP} and $Cert_{SP}$ in its MC, as presented in Table 2.
5. The IdP sends $SK_{SP}||ADD_{IdP}$ via a secure channel to the SP.
6. Upon receiving $SK_{SP}||ADD_{IdP}$, the SP stores ID_{SP}, $ESK_{SP} = SK_{SP} \oplus h(PW_{SP})$, and ADD_{IdP}, as listed in Table 3.

Details regarding the registration phase are shown in Figure 14.

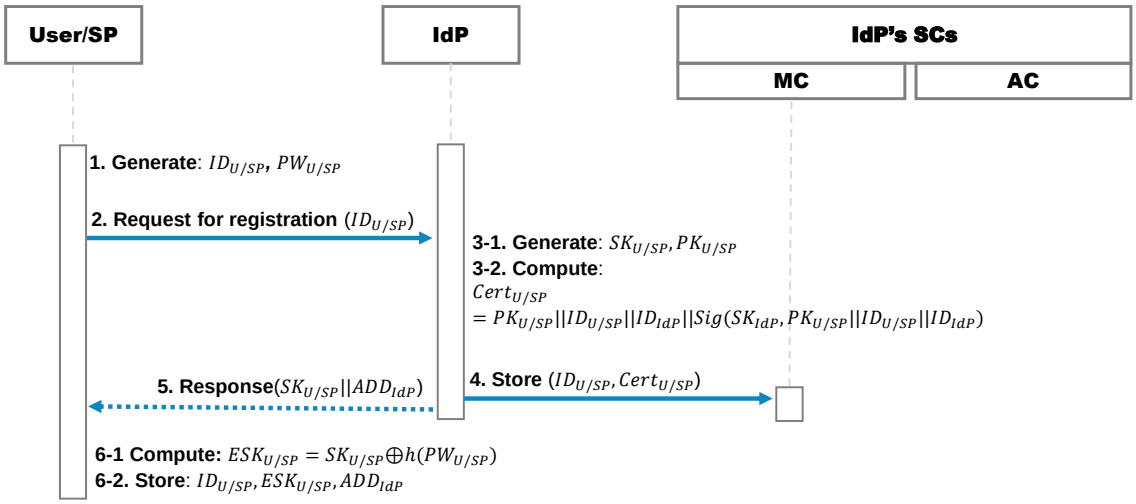

Figure 14. Registration phase of the proposed scheme.

7.3. Mutual Authentication Phase

In this authentication step, a user and an SP perform mutual authentication using the IdP SCs for the user to access the SP. After this phase, the user and the SP can generate session keys for secure communication. The mutual authentication phase process is as follows:

1. The user enters PW_U and computes $SK_U = ESK_U \oplus h(PW_U)$. Subsequently, the user generates timestamp ts_U and a random number, r_U, and computes $K_U = r_U \cdot G$, $T_U = h(K_U||ts_U)$, and $V_U = EC.sign(SK_U, T_U)$. Subsequently, the user generates the request message $M_U = TID_U||ADD_{IdP}||K_U||V_U||ts_U$.
2. The user requests to log in by sending M_U to the SP via a public channel.
3. Upon receiving M_U, the SP verifies whether ts_U is in a valid range. The SP terminates the authentication process if ts_U is invalid. Otherwise, the SP generates the MC's input as $TX_{in} = Type_{entity}||TID_U|||V_U||ts_U$, where $Type_{entity}$ represents the "user".
4. The SP invokes the IdP's MC using ADD_{IdP} in M_U by sending TX_{in}.
5. The MC searches for the ID_U that matches TID_U in TX_{in}, and then verifies the revocation status. If the revocation status of the ID_U is *true*, then the MC returns the fail signal to the SP; otherwise, the MC searches for a certificate, $Cert_U$, that matches ID_U.
6. The MC calls its own AC as a delegate call by sending TX_{in} and $Cert_U$.
7. Upon receiving TX_{in} and $Cert_U$, the AC verifies V_U in TX_{in} as $EC.Verify(PK_U, V_U)$. Subsequently, if $Type_{entity}$ equals "user", the AC changes the temporary identity of the user to $TID_U = h(TID_U||V_U)$ and stores it in the MC's storage.
8. If V_U is invalid, the AC returns a failed signal to the MC. Otherwise, the AC returns a successful signal to the MC.
9. Upon receiving the results of the AC, the MC returns ID_U to the SP.
10. Upon receiving the results of the MC, the SP enters PW_{SP} and computes $SK_{SP} = ESK_{SP} \oplus h(PW_{SP})$. Next, the SP generates the timestamp ts_{SP} and a random number r_{SP} and computes $K_{SP} = r_{SP} \cdot G$, $T_{SP} = h(K_{SP}||ts_{SP})$, and $V_{SP} = EC.sign(SK_{SP}, T_{SP})$. Subsequently, the SP generates a response message, $M_{SP} = TID_{SP}||ADD_{IdP}||K_{SP}||V_{SP}||ts_{SP}$, and generates a session key, $SK = r_{SP} \cdot K_U$.
11. The SP allows login for the user corresponding to the identity ID_U by sending M_{SP} to the user.
12. Upon receiving M_{SP}, the user verifies whether ts_{SP} is within a valid range. If ts_{SP} is invalid, then the user terminates the authentication process. Subsequently, the user changes TID_U to $TID_U = h(TID_U||V_U)$ and generates a session key as $SK = r_U \cdot K_{SP}$. Next, the user generates the MC's input as $TX_{in} = Type_{entity}||TID_{SP}||V_{SP}||ts_{SP}$, where $Type_{entity}$ represents "SP".
13. The user invokes the IdP's MC using ADD_{IdP} in M_{SP} by sending TX_{in}.
14. The MC searches for the ID_{SP} that matches TID_{SP} in TX_{in} and then verifies the revocation status. If the revocation status of ID_{SP} is true, then the MC returns the fail signal to the user; otherwise, the MC searches for a certificate, $Cert_{SP}$, that matches ID_{SP}.
15. The MC calls its AC as a delegate call by sending TX_{in} and $Cert_{SP}$.
16. Upon receiving TX_{in} and $Cert_{SP}$, the AC verifies V_{SP} in TX_{in} as $EC.Verify(PK_{SP}, V_{SP})$.
17. If V_{SP} is invalid, the AC returns a failed signal to the MC. Otherwise, the AC returns a successful signal to the MC.
18. Upon receiving the AC's result indicating a failed signal, the MC returns it to the user. Otherwise, the MC returns the user ID_{SP} to the user.

Details regarding the mutual authentication phase are shown in Figure 15.

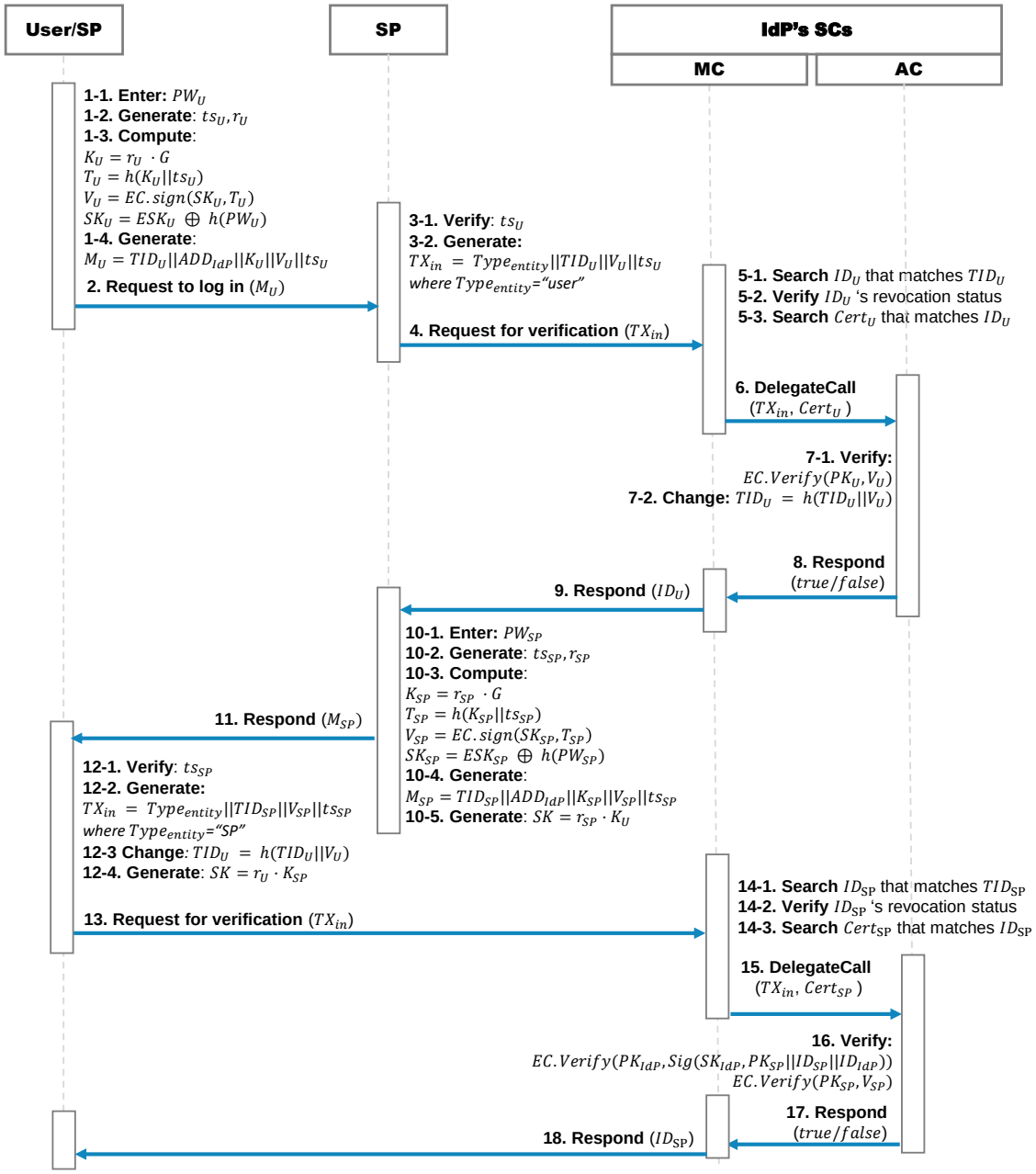

Figure 15. Mutual authentication phase of proposed scheme.

7.4. Revocation Phase

In this registration phase, the user and SP revoke their registration because of the loss of the ECDSA private key. The user can perform revocation using the SP or IdP, whereas the SP can be revoked using the IdP. The revocation process is as follows:

1. The user enters PW_U and computes $SK_U = ESK_U \oplus h(PW_U)$. Next, the user generates timestamp ts_U, selects the revocation list $textReason$, and computes $T_U = h(ts_U||textReason)$ and $V_U = EC.sign(SK_U, T_U)$. Subsequently, the user generates a request message $M_U = TID_U||ADD_{IdP}||V_U||ts_U||textReason$.
2. The user requests the user revocation by sending M_U to the SP or the IdP via a public channel.
3. Upon receiving M_U, the SP or the IdP verifies whether ts_U is within a valid range. If ts_U is invalid, the SP or IdP terminates the revocation process. Otherwise, the SP or IdP generates the MC input as $TX_{in} = TID_U||V_U||ts_U||textReason$.

4. The SP or IdP invokes the IdP's MC using ADD_{IdP} in M_U by sending TX_{in}.
5. The MC searches for an ID_U that matches TID_U in TX_{in}; subsequently, the MC verifies the revocation status. If the revocation status of ID_U is *true*, the MC returns a fail signal to the requester. Otherwise, the MC searches for a certificate, $Cert_U$, that matches ID_U.
6. The MC calls its own AC as a delegate call by sending TX_{in} and $Cert_U$.
7. Upon receiving TX_{in} and $Cert_U$, the AC verifies V_U in TX_{in} as $EC.Verify(PK_U, V_U)$. Subsequently, the AC changes the revocation status of ID_U to true and stores it in the MC storage.
8. If V_U is invalid, the AC returns a failed signal to the MC. Otherwise, the AC returns a successful signal to the MC.
9. Upon receiving the AC's result, the MC returns the result to the SP or the IdP.
10. Upon receiving the SP or IdP's result, the SP or IdP returns the result to the user.

The revocation phase of the SP is identical to that of the users. The SP revocation process is as follows:

1. The SP enters PW_{SP} and computes $SK_{SP} = ESK_{SP} \oplus h(PW_{SP})$. Next, the SP generates timestamp ts_{SP}, selects the revocation list $textReason$, and computes $T_U = h(ts_{SP}||textReason)$ and $V_{SP} = EC.sign(SK_{SP}, T_{SP})$. Subsequently, the user generates a request message, $M_{SP} = TID_{SP}||ADD_{IdP}||V_{SP}||textReason$.
2. The SP requests revocation by sending M_{SP} to the IdP via a public channel.
3. Upon receiving M_{SP}, the IdP verifies whether ts_{SP} is within a valid range. If ts_{SP} is invalid, the IdP terminates the revocation. Otherwise, the IdP generates the MC input as $TX_{in} = TID_{SP}||V_{SP}||ts_{SP}||textReason$.
4. The IdP invokes the IdP's MC using ADD_{IdP} in M_{SP} by sending TX_{in}.
5. The MC searches for the ID_{SP} that matches TID_{SP} in TX_{in}. Subsequently, the MC verifies the revocation status. If the revocation status of ID_{SP} is true, the MC returns a fail signal to the requester. Otherwise, the MC searches for a certificate, $Cert_{SP}$, that matches ID_{SP}.
6. The MC calls its own AC as a delegate call by sending TX_{in} and $Cert_{SP}$.
7. Upon receiving TX_{in} and $Cert_{SP}$, the AC verifies V_{SP} in TX_{in} as $EC.Verify(PK_{SP}, V_{SP})$. Subsequently, the AC changes the revocation status of ID_{SP} to *true* and stores it in the MC storage.
8. If V_{SP} is invalid, the AC returns a failed signal to the MC. Otherwise, the AC returns a successful signal to the MC.
9. Upon receiving the AC's result, the MC returns the result to the IdP.
10. Upon receiving the IdP's result, the IdP returns the result to the SP.

Details regarding the revocation phase are shown in Figure 16.

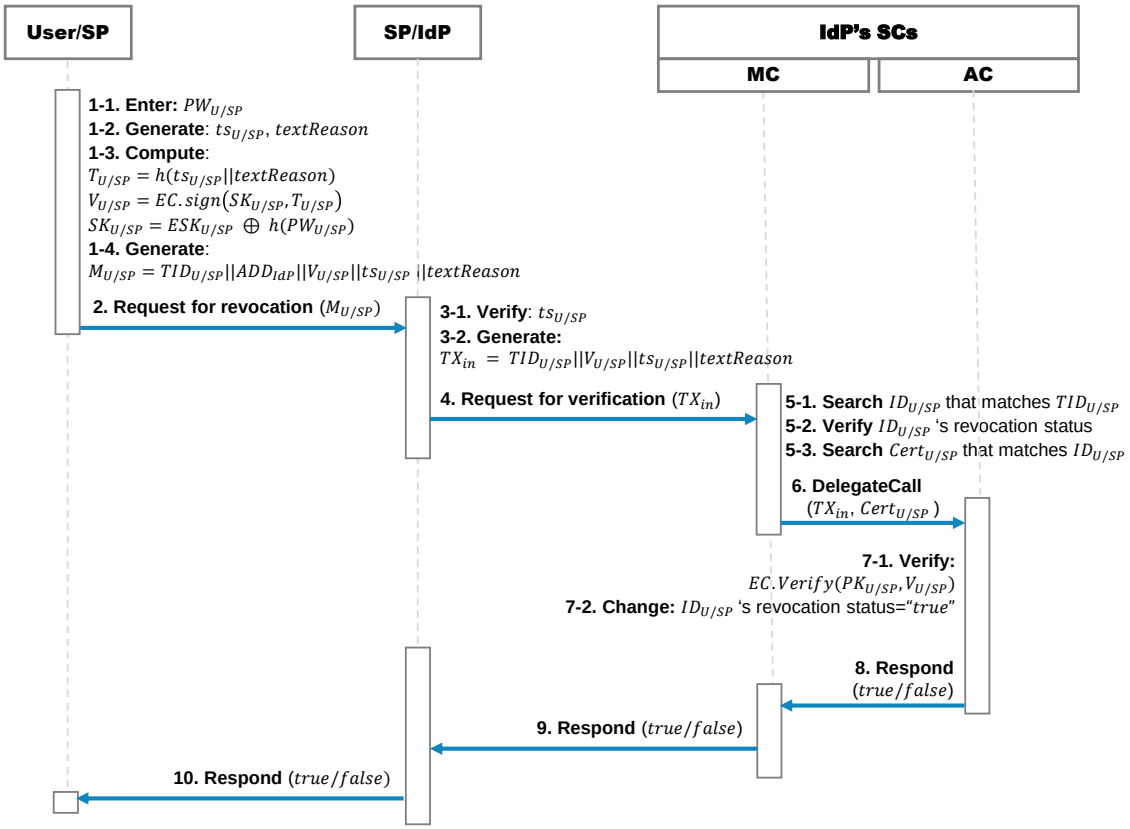

Figure 16. Revocation phase of the proposed scheme.

7.5. Smart Contract Updating Phase

In the SC updating phase, an IdP can update its AC. The IdP creates and deploys a new AC and updates ADD_{AC} in its MC.

8. Security Analysis of Proposed Scheme

This section analyzes the security of the proposed scheme using two methods. For a formal security analysis, the ProVerif [27] protocol verification tool is used to show that the proposed scheme can satisfy security and authentication features. In addition, an informal security analysis is presented to demonstrate that the proposed scheme satisfies the security requirements for federated authentication.

8.1. Formal Security Analysis

Herein, ProVerif [27] was used to analyze the security of the proposed scheme. Recently, Ryu et al. [28], Kang et al. [29], Zhang et al. [30], and Edris et al. [31] have adopted ProVerif [27], which is an automatic cryptographic protocol verifier, for protocol security validation. Based on the formal security analysis in this study, the session key was confirmed to not be exposed. Three channels were used for security analysis. "Private Channel 1" transmits sensitive data between the user and the SP or the IdP. "Public Channel 1" transmits general data between the user and SP. Finally, "Public Channel 2" transmits general data between the user or the SP and SCs. Table 4 lists the channels, variables, and related parameters. Tables 5–7 list the processes of the user, SP, and SCs, respectively, during the registration and mutual authentication phases. The SC process was analyzed for an AC that performs an actual authentication operation. The queries and the main processes are listed in Table 8. The results of the proposed scheme are presented in Table 9. The query, not attackers (SK[]), is true in Table 9, thus indicating that the proposed protocol protects the session key, SK, from attackers.

Table 4. Definitions of channels, variables, and other related parameters.

```
(*—-channels—-*)
free privateChannel1:channel [private].
free publicChannel1:channel.
free publicChannel2:channel.
(*—-constants—-*)
free IDU:bitstring [private].
free IDSP:bitstring [private].
free ADDAC:bitstring.
(*—-shared key—-*)
free G:bitstring [private].
free SK:bitstring [private].
(*—-functions—-*)
fun concat(bitstring, bitstring):bitstring.
fun h(bitstring):bitstring.
fun mult(bitstring, bitstring):bitstring.
fun sign(bitstring, bitstring):bitstring.
fun verify(bitstring, bitstring):bitstring.
(*—-events—-*)
event startUi(bitstring).
event endUi(bitstring).
event startSP(bitstring).
event endSP(bitstring).
event startAC(bitstring).
event endAC(bitstring).
```

Table 5. User process.

```
(*—-Ui process—-*)
let Ui =
out(privateChannel1,(IDU));
in(privateChannel1, (XSKu:bitstring, ADDIdP:bitstring));
let TIDU=IDU in
event startUi(IDU);
new tsU:bitstring;
new rU:bitstring;
let KU = mult(rU, G) in
let TU = h(concat(KU, tsU)) in
let VU = sign(XSKu, TU) in
out(publicChannel1, (TIDU, ADDIdP, KU, VU, tsU));
in(publicChannel1, (TIDSP:bitstring, ADDIdP2:bitstring, KSP:bitstring, VSP:bitstring, tsSP:bitstring));
let TID=h(concat(TIDU, VU)) in
let SK = mult(rU, KSP) in
new Typeentity:bitstring;
out(publicChannel2, (Typeentity, TIDSP, VSP, tsSP));
in(publicChannel2, (xIDSP:bitstring));
event endUi(IDU).
```

Table 6. SP process.

```
(*——SP process——*)
let SP =
out(privateChannel1,(IDSP));
in(privateChannel1, (XSKsp:bitstring, ADDIdP:bitstring));
let TIDSP=IDSP in
event startSP(IDSP);
in(publicChannel1, (TIDU:bitstring, ADDIdP2:bitstring, KU:bitstring, VU:bitstring, tsU:bitstring));
new Typeentity:bitstring;
out(publicChannel2, (Typeentity, TIDU, VU, tsU));
in(publicChannel2, (xIDU:bitstring));
new tsSP:bitstring;
new rSP:bitstring;
let KSP = mult(rSP, G) in
let TSP = h(concat(KSP, tsSP)) in
let VSP = sign(XSKsp, TSP) in
let SK = mult(rSP, KU) in
out(publicChannel1, (TIDSP, ADDIdP2, KSP, VSP, tsSP));
event endUi(IDSP).
```

Table 7. AC process.

```
(*——AC process——*)
let AC =
in(privateChannel1,(XIDentity:bitstring));
new SKIdP:bitstring;
new PKIdP:bitstring;
new SKentity:bitstring;
new PKentity:bitstring;
new IDIdP:bitstring;
let Certsign = sign(SKIdP, concat(concat(PKentity,XIDentity),IDIdP)) in
event startAC(ADDAC);
in(publicChannel2, (XPKentity:bitstring, Ventity:bitstring));
new true:bitstring;
if (verify(XPKentity, Ventity)=true) then
out(publicChannel2, true);
event endAC(ADDAC).
```

Table 8. Queries and main processes.

```
(*——queries——*)
query idu:bitstring; inj-event(endUi(idu)) ==> inj-event(startUi(idu)).
query idsp:bitstring; inj-event(endSP(idsp)) ==> inj-event(startSP(idsp)).
query addac:bitstring; inj-event(endAC(addac)) ==> inj-event(startAC(addac)).
query attacker(SK).
(*——process——*)
process
((!Ui) | (!SP) | (!AC))
```

Table 9. Results.

```
Verification summary:
Query inj-event(endUi(idu)) ==> inj-event(startUi(idu)) is true.
Query inj-event(endSP(idsp)) ==> inj-event(startSP(idsp)) is true.
Query inj-event(endAC(addac)) ==> inj-event(startAC(addac)) is true.
Query not attacker(SK[]) is true.
```

8.2. Informal Security Analysis

1. ID Guessing Attack: In the proposed scheme, $ID_{U/SP}$ is not directly used. As this study employed $TID_{U/SP}$, which changes every authentication, rather than $ID_{U/SP}$, an attacker cannot obtain $ID_{U/SP}$ through a public channel. Therefore, the proposed scheme can protect against ID guessing attacks.
2. Replay Attack: Even if the attacker steals M_U in the mutual authentication phase and presents it to the SP, the SP can determine whether the message M_U is reused because the SP verifies that the timestamp ts_U is within a valid range. Moreover, because V_U is a digital signature that can only be created using the user's private key, ts_U cannot be changed and a signature for the changed $ts_U^{'}$ cannot be generated. Therefore, the proposed scheme can provide protection against replay attacks.
3. User Impersonation Attack: Even if the attacker steals M_U and replaces the user's identity, TID_U, with the attacker's identity, TID_A, the SP can identify an invalid user. If the attacker is not registered in the IdP registered by the user, then the MC returns a fail signal to the SP because it can not identify ID_A that matches TID_U. In addition, if the attacker is registered in the IdP registered by the user, the MC returns a fail signal to the SP because the AC verifies V_U with $Cert_A$ and fails the verification. Therefore, the proposed scheme can provide protection against user impersonation attacks.
4. Server Impersonation Attack: For the same reason as user impersonation attacks, the user can determine an invalid SP. Even if the attacker steals M_{SP} and replaces the SP's identity, TID_{SP}, with the attacker's identity, TID_{SP}, the user can determine an invalid SP. If the attacker is unregistered in the IdP registered with the SP, the MC returns a fail signal to the user because the MC cannot find the ID_{SP} that matches TID_{SP}. In addition, if the attacker is registered in the IdP registered with the SP, the MC returns a fail signal to the user because the AC verifies V_{SP} with $Cert_A$ and fails the verification. Therefore, the proposed scheme can protect against server impersonation attacks.
5. Privileged Insider Attack: In the proposed scheme, the SP and IdP are separated and the SP does not possess information related to user authentication; therefore, an attack by an insider of the SP is impossible. In addition, without the random number, r_U, possessed only by the user, the session key cannot be obtained with only ID_U, TID_U, and $Cert_U$ stored in the IdP's SC. Therefore, the proposed scheme can provide protection against privileged insider attacks.
6. Session Key Disclosure: Computing the session keys $SK = r_U \cdot r_{SP} \cdot G$, r_U and r_{SP} is required. However, because r_U and r_{SP} contain only the user and SP information, only the user's session keys and the SP can be generated. Therefore, the proposed scheme can protect against session key disclosure attacks.
7. Forward Secrecy and Backward Secrecy: Even if the attacker obtains the session key, they will be unable to access the previous or subsequent keys, as each one is generated randomly with no correlation to the others. Therefore, the proposed scheme can preserve forward and backward secrecy.
8. Mutual Authentication: The user and the SP authenticate each other by verifying M_U and M_{SP} because M_U and M_{SP} can utilize their private keys. Therefore, the proposed scheme provides mutual authentication.
9. User Anonymity: In the proposed scheme, $TID_{U/SP}$, which changes at every authentication step, was used. Therefore, the proposed scheme ensures user anonymity.
10. Single Point-of-Failure: Even if the IdP system fails, user authentication can be performed normally because the proposed scheme uses SCs for authentication. Therefore, the proposed scheme can protect against single-point failures.

The results of the security analysis, with comparisons to related studies, are presented in Table 10.

Table 10. Comparison of security features.

Security Features	Cheng et al. [8]	Xiang et al. [7]	Xue et al. [13]	Ours
1. ID Guessing Attack	O	O	X	O
2. Replay Attack	O	O	O	O
3. User Impersonation Attack	O	O	X	O
4. Server Impersonation Attack	O	O	X	O
5. Privileged Insider Attack	O	O	X	O
6. Session Key Disclosure	O	O	O	O
7. Forward Secrecy and Backward Secrecy	O	O	O	O
8. Mutual Authentication	O	O	O	O
9. User Anonymity	O	O	X	O
10. Single Point-of-Failure	X	X	O	O

9. Performance Analysis

This section analyzes the performance of our system in terms of the computational cost of cryptographic calculations using the same method as that used in numerous authentication-related studies [32–37]. The development environment used to analyze the performance is presented in Table 11. The five symbols listed in Table 12 were used for the performance analysis of the proposed scheme. A comparison of the computational overhead in the authentication phase was performed between the proposed and other related schemes [7,8,13] (see Table 13).

Table 11. Development environment.

Item	Value
CPU	Intel(R) Core(TM) i7-8565U CPU @ 1.80 GHz 1.99 GHz
RAM	16.0 GB
OS	Windows 10 Home
Software	JDK 17
Security level	secp521r1 ECC

Table 12. Computational cost of cryptographic calculations (ms).

Symbol	Meaning	Time (ms)
T_h	The computational cost of a one-way hash function operation.	0.007
T_m	The computational cost of scalar multiplication in the field.	34.32
T_e	The computational cost of symmetric encryption or decryption.	0.23
T_s	The computational cost of the ECDSA sign.	22.93
T_v	The computational cost of the ECDSA verification.	44.03

Table 13. Comparison of computational costs.

Schemes	Cheng et al. [8]	Xiang et al. [7]	Xue et al. [13]	Ours
User	$4T_m + 4T_h + 2T_e$ $= 138.02$	$8T_m + 4T_h$ $= 274.84$	$2T_m + T_h + T_s$ $= 91.64$	$2T_m + 3T_h + T_s$ $= 91.78$
Server	$4T_m + 2T_h + T_e$ $= 137.65$	$9T_m + 4T_h$ $= 309.16$	$2T_m + T_h + T_s$ $= 91.64$	$2T_m + 2T_h + T_s$ $= 91.71$
Smart Contracts	- -	- -	$T_h + 2T_v$ $= 88.13$	$2T_h + T_v$ $= 44.17$
Total	$8T_m + 6T_h + 3T_e$ $= 275.67$	$18T_m + 8T_h$ $= 618.32$	$4T_m + 3T_h + 2T_s + 2T_v$ $= 271.41$	$4T_m + 7T_h + 2T_s + T_v$ $= 227.66$

10. Discussion and Limitations

This study provides a distributed and federated authentication scheme that can resist various security threats, including a single point-of-failure. That is, even if the IdP server

fails, the authentication service using the corresponding IdP is not affected. Moreover, this scheme using updatable SCs can support the fixing of programming bugs, adding the function of a smart contract, or changing business logic. For example, the previously registered user and SP data can be easily maintained, even if the cryptographic algorithms must be changed for security. In Section 9, the computational overhead during the authentication phase of the three schemes was compared [7,8,13].

$$(t_1 - t_2)/t_2 \tag{1}$$

In Equation (1), t_1 represents the average computational cost of the two schemes [7,8] and t_1 represents the cost of the proposed method. The results of the computational calculations indicate that the proposed method outperformed the other methods by 71%. Therefore, the proposed scheme is expected to provide federated authentication services efficiently using SCs. However, the proposed scheme assumes that the private key is stored on a tamper-proof smart card. Therefore, the user may have the inconvenience of having to carry a smart card. In future studies, we will focus on improving usability, and we plan to conduct additional research and apply private key protection technology, such as behavior-based biometric authentication, without the need for a smart card.

11. Conclusions

In this study, a distributed and federated authentication scheme was proposed based on updatable SCs. Federated authentication, such as Google ID, enables users to conveniently access multiple websites using a single login credential. In the event of an IdP server failure, the entire service may become inaccessible owing to the IdP server's general management of all login credentials. Thus, in this study, we focused on using SCs to overcome a single point-of-failure in federated authentication. A security analysis was performed and confirmed that the proposed scheme provides protection against not only single point failure, but also common attacks, including ID guessing attacks, replay attacks, user impersonation attacks, server impersonation attacks, privileged insider attacks, session key disclosure, and forward and backward security. Furthermore, the proposed scheme can maintain previously registered user and SP data even if the SC is updated for bug fixes or to update the authentication function of an SC. In addition, the proposed scheme outperformed other methods by 71%. Thus, it is expected to be widely used as one login credential in various fields such as internet banking and e-commerce without concerns regarding single points-of-failure. However, the user may have the inconvenience of carrying a smart card, because our scheme assumes that the private key is stored in a tamper-proof smart card. Therefore, further studies are required on technologies that protect private keys, such as behavior-based biometric authentication, to improve usability.

Author Contributions: Conceptualization, K.K.; methodology, K.K. and J.R.; software, K.K.; validation, J.R. and H.L.; formal analysis, K.K.; investigation, D.W.; writing—original draft preparation, K.K. and J.R.; writing—review and editing, H.L. and Y.L.; visualization, K.K. and H.L.; supervision, Y.L. and D.W.; project administration, Y.L. and D.W.; funding acquisition, D.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: This work was supported by an Institute of Information & Communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00558, Development of National Statistical Analysis System using Homomorphic Encryption Technology).

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

References

1. Carretero, J.; Izquierdo-Moreno, G.; Vasile-Cabezas, M.; Garcia-Blas, J. Federated identity architecture of the European eID system. *IEEE Access.* **2018**, *6*, 75302–75326. [CrossRef]
2. Chen, Y.; Dong, G.; Hao, Y.; Zhang, Z.; Peng, H.; Yu, S. An Open Identity Authentication Scheme Based on Blockchain. In *Algorithms and Architectures for Parallel Processing, Proceedings of the 19th International Conference, ICA3PP 2019, Melbourne, VIC, Australia, 9–11 December 2019*; Springer International Publishing: Cham, Switzerland, 2020; pp. 421–438.
3. Dey, A.; Weis, S. PseudoID: Enhancing privacy in federated login. In *Hot Topics in Privacy Enhancing Technologies*; 2010; pp. 95–107. Available online: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/36553.pdf (accessed on 9 February 2023).
4. Chadwick, D. Federated identity management. *Found. Secur. Anal. Des. V* **2009**, 96–120. [CrossRef]
5. Isaakidis, M.; Halpin, H.; Danezis, G. UnlimitID: Privacy-preserving federated identity management using algebraic MACs. In Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society, Vienna, Austria, 24–28 October 2016; pp. 139–142.
6. Jensen, J. Federated identity management challenges. In Proceedings of the Seventh International Conference on Availability, Reliability and Security, Prague, Czech Republic, 20–24 August 2012; pp. 230–235.
7. Xiang, X.; Cao, J.; Fan, W. Decentralized authentication and access control protocol for blockchain-based e-health systems. *J. Netw. Comput. Appl.* **2022**, *207*, 103512. [CrossRef]
8. Cheng, X.; Chen, F.; Xie, D.; Sun, H.; Huang, C. Design of a secure medical data sharing scheme based on blockchain. *J. Med. Syst.* **2020**, *44*, 52. [CrossRef]
9. Zhang, R.; Xue, R.; Liu, L. Security and privacy for healthcare blockchains. *IEEE Trans. Serv. Comput.* **2021**, *15*, 3668–3686. [CrossRef]
10. Feng, X.; Cui, K.; Jiang, H.; Li, Z. EBAS: An Efficient Blockchain-Based Authentication Scheme for Secure Communication in Vehicular Ad Hoc Network. *Symmetry* **2022**, *14*, 1230. [CrossRef]
11. Tomar, A.; Tripathi, S. BCAV: Blockchain-based certificateless authentication system for vehicular network. *Peer-Peer Netw. Appl.* **2022**, *15*, 1733–1756. [CrossRef]
12. Gong, C.; Xiong, L.; He, X.; Niu, X. Blockchain-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. *J. Ambient. Intell. Humaniz. Comput.* **2022**, 1–14. [CrossRef]
13. Xue, K.; Luo, X.; Ma, Y.; Li, J.; Liu, J.; Wei, D.S.L. A Distributed Authentication Scheme Based on Smart Contract for Roaming Service in Mobile Vehicular Networks. *IEEE Trans. Veh. Technol.* **2022**, *71*, 5284–5297. [CrossRef]
14. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, W.; Chen, X.; Weng, J.; Imran, M. An overview on smart contracts: Challenges, advances and platforms. *Future Gener. Comput. Syst.* **2020**, *105*, 475–491. [CrossRef]
15. Zheng, G.; Gao, L.; Huang, L.; Guan, J.; Zheng, G.; Gao, L.; Huang, L.; Guan, J. Upgradable contract. *Ethereum Smart Contract Dev. Solidity* **2021**, 197–213. [CrossRef]
16. Wöhrer, M.; Zdun, U. Design patterns for smart contracts in the ethereum ecosystem. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018; pp. 1513–1520.
17. Shao, W.; Wang, Z.; Wang, X.; Qiu, K.; Jia, C.; Jiang, C. LSC: Online auto-update smart contracts for fortifying blockchain-based log systems. *Inf. Sci.* **2020**, *512*, 506–517. [CrossRef]
18. Górski, T. Reconfigurable Smart Contracts for Renewable Energy Exchange with Re-Use of Verification Rules. *Appl. Sci.* **2022**, *12*, 5339. [CrossRef]
19. Zhang, Z.; Krol, M.; Sonnino, A.; Zhang, L.; Riviére, E. EL PASSO: Efficient and lightweight privacy-preserving single sign on. *Proc. Priv. Enhancing Technol.* **2021**, *2021*, 70–87. [CrossRef]
20. Szabo, N. Smart contracts: Building blocks for digital markets. *EXTROPY J. Transhumanist Thought* **1996**, *18*, 28.
21. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2017**, *151*, 1–32.
22. American National Standards Institute. *X9. 62, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*; American National Standards Institute, X9-Financial Services: Annapolis, MD, USA, 2005.
23. Qu, M. *Section 2: Recommended Elliptic Curve Domain Parameters*; Tech. Rep. SEC2-Ver-0.6; Certicom Res.: Mississauga, ON, Canada, 1999.
24. Andi, A.; Juliandy, C.; Robet, R.; Pribadi, O. Securing Medical Records of COVID-19 Patients Using Elliptic Curve Digital Signature Algorithm (ECDSA) in Blockchain. *CommIT (Commun. Inf. Technol.) J.* **2022**, *16*, 87–96. [CrossRef]
25. Al-Zubaidie, M.; Zhang, Z.; Zhang, J. Efficient and secure ECDSA algorithm and its applications: A survey. *Int. J. Com-Munication Netw. Inf. Secur. (IJCNIS)* **2019**, *11*, 7–35. [CrossRef]
26. Ethereum.org. Upgrading Smart Contracts. Available online: https://ethereum.org/en/developers/docs/smart-contracts/upgrading/ (accessed on 9 February 2023).
27. Blanchet, B.; Smyth, B.; Cheval, V.; Sylvestre, M. ProVerif 2.04: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial. Available online: https://proverif.inria.fr/manual.pdf (accessed on 9 February 2023).
28. Ryu, J.; Lee, H.; Lee, Y.; Won, D. SMASG: Secure Mobile Authentication Scheme for Global Mobility Network. *IEEE Access* **2022**, *10*, 26907–26919. [CrossRef]

29. Kang, D.; Lee, H.; Lee, Y.; Won, D. Lightweight user authentication scheme for roaming service in GLOMONET with privacy preserving. *PLoS ONE* **2021**, *16*, e0247441. [CrossRef]
30. Zhang, J.; Yang, L.; Cao, W.; Wang, Q. Formal analysis of 5G EAP-TLS authentication protocol using proverif. *IEEE Access* **2020**, *8*, 23674–23688. [CrossRef]
31. Edris, E.K.K.; Aiash, M.; Loo, J. Formal verification of authentication and service authorization protocols in 5G-enabled device-to-device communications using ProVerif. *Electronics* **2021**, *10*, 1608. [CrossRef]
32. Wu, Z.Y.; Lee, Y.C.; Lai, F.; Lee, H.C.; Chung, Y. A secure authentication scheme for telecare medicine information systems. *J. Med. Syst.* **2012**, *36*, 1529–1535. [CrossRef]
33. Wu, F.; Xu, L.; Kumari, S.; Li, X. An improved and provably secure three-factor user authentication scheme for wireless sensor networks. *Peer-Peer Netw. Appl.* **2018**, *11*, 1–20. [CrossRef]
34. Ryu, J.; Lee, H.; Kim, H.; Won, D. Secure and efficient three-factor protocol for wireless sensor networks. *Sensors* **2018**, *18*, 4481. [CrossRef]
35. Ryu, J.; Kang, D.; Lee, H.; Kim, H.; Won, D. A secure and lightweight three-factor-based authentication scheme for smart healthcare systems. *Sensors* **2020**, *20*, 7136. [CrossRef] [PubMed]
36. Kim, K.; Ryu, J.; Lee, Y.; Won, D. An Improved Lightweight User Authentication Scheme for the Internet of Medical Things. *Sensors* **2023**, *23*, 1122. [CrossRef] [PubMed]
37. Xu, L.; Wu, F. Cryptanalysis and improvement of a user authentication scheme preserving uniqueness and anonymity for connected health care. *J. Med. Syst.* **2015**, *39*, 1–9. [CrossRef] [PubMed]