Article

Key Backup and Recovery for Resilient DID Environment

Jihwan Kim [1,†], Pyung Kim [2,†], Younho Lee [3,*] and Daeseon Choi [4,*]

1 Graduate School of IT and Policy, SeoulTech, Seoul 01811, Republic of Korea; jihwan0724@seoultech.ac.kr
2 Department of Computer Science, Inje University, Gimhae 50834, Republic of Korea; pkim@inje.ac.kr
3 Department of Industrial Engineering, SeoulTech, Seoul 01811, Republic of Korea
4 Department of Software, Soong-Sil University, Seoul 07027, Republic of Korea
* Correspondence: younholee@seoultech.ac.kr (Y.L.); sunchoi@ssu.ac.kr (D.C.); Tel.: +82-2-820-0943 (Y.L.); +82-2-970-7283 (D.C.)
† These authors contributed equally to this work.

Abstract: This paper delves into the advantages of authentication algorithms employing self-sovereign identity, highlighting a reduced communication overhead and the elimination of single points of failure. However, it acknowledges the vulnerability of digital wallets to real-world issues like loss or theft. To address these challenges, we propose an efficient key backup and recovery protocol based on the FROST threshold signature algorithm. This protocol involves trusted third parties and backup devices, ensuring secure secret key sharing and rapid key recovery. Performance evaluations, including key recovery time, demonstrate the protocol's efficiency and reliability, bolstering the overall robustness of self-sovereign identity systems.

Keywords: FROST (flexible round-optimized Schnorr threshold signatures); interoperability technology of digital identification; multi-party distributed signature; universal digital wallet; key backup; key recovery

MSC: 94A60; 68M25

check for updates

Citation: Kim, J.; Kim, P.; Lee, Y.; Choi, D. Key Backup and Recovery for Resilient DID Environment. *Mathematics* **2024**, *12*, 830. https://doi.org/10.3390/math12060830

Academic Editor: Antanas Cenys

Received: 31 January 2024
Revised: 24 February 2024
Accepted: 4 March 2024
Published: 12 March 2024

1. Introduction

In the field of digital identity management, self-sovereign identity (SSI) is a technical concept that allows users to directly own and manage their identity information, unlike centralized identity authentication models. For example, SSI refers to allowing the user, not the server, to manage the identity information required during the registration/login process for websites, services, and applications on the web. It is a new paradigm that provides users with the ability to store and control their own identity information, completely [1].

Decentralized identifiers (DID) [2] and verifiable credentials (VC) [3], which are W3C standards, are essential elements for independently managing and proving personal identity information in a self-sovereign identity environment. As with digital or physical identity management, there are three stakeholders in a self-sovereign identity: the issuer, the identity holder, and the verifier. H. Yildiz et al. [4] explains the interrelationship between the three main roles as an identity trust triangle. First, the identity holder receives and stores a credential from the issuer, and the identity holder signs the presentation derived from the credential with a secret key and submits it to the verifier. The verifier verifies the presentation's signature through the public key. The trust relationship between the verifier and the issuer is established indirectly through the public key storage of the credential. Lastly, a digital wallet is also essential for the individual elements mentioned above to operate as components of commercialized services [5]. A digital wallet plays a central role in implementing and managing self-sovereign identity, and enables the creation and verification of secure signatures required for authentication with private and public keys [6,7].

Authentication algorithms using self-sovereign identity have several performance advantages in the process of participants performing and verifying signatures. First of all, it reduces the communication overhead in that there is no need to communicate with a central management server, which is a trusted third party. Another advantage is that it avoids single points of failure on the central management server. However, digital wallets, which are essential applications for managing self-sovereign identities, suffer from the problems of loss, theft, and damage of the secret information, as they do in the real world. In network environments where self-sovereign identities are used, the risk may increase especially significantly given that the network is widely connected and communication between participants occurs frequently.

To solve these problems, research is being conducted on distributing and managing some keys to other trusted parties [8,9] rather than single key-based signatures like the existing BBS+ signature [10]. In order to manage encryption keys in a self-sovereign identity environment, an efficient distributed key management protocol is required, and for this purpose, MPC (multi-party computation)-based signature protocols that use multiple keys rather than one key are being studied [9,11–17]. In particular, the threshold signature system [11] proposed by Desmedt and Frankel uses an approach that can dramatically reduce the security risk for a single point by distributing signature authority to multiple participants. However, the traditional threshold signature system has the potential threat of an attacker being able to create a valid signature if a certain number, or threshold, of keys are collected by the attacker [18]. Therefore, problems regarding the theft, loss, or damage of the user's key may still occur.

In this paper, to overcome the problems stated above, we propose an efficient key backup and recovery protocol based on the threshold signature protocol. The research of Kim, J. et al. [19] is a prior study that applied the Universal Wallet standard [20] and the WACI [21] protocol to strengthen the interoperability of digital wallets, and we extend this. We employ two entities, a trusted third party and a backup device, which can be accessed all times. By using the FROST threshold signature algorithm [22] based on Schnorr signatures, which provides a minimum number of communication rounds for secret key generation and enables secure secret key sharing, we share each user's secret key with the backup device and the trusted third party. The FROST threshold signature algorithm is an algorithm that can generate a signature when there are more than t participants among n people. We apply this mechanism to the recovery of secret information. Because of the sharing, a user can easily recover his secret key share even he loses his secret key. The key recovery algorithm, which is the core of this study, is implemented based on the FROST signature algorithm [22].

The performance is verified for each session of the overall key management framework, including the key recovery stage. In particular, the time taken to recover the key was measured as very short at 0.02 ms, assuming that it was not affected by network speed. This is because participants created the recovery key based on secret information created at the time of initial creation and distribution of the distributed key. In other words, it may take some time to generate and distribute the distributed key through secret sharing, but the time required to recover the key is short.

The organization of this paper is as follows. Section 2 briefly explains the component technologies to help you understand the method proposed in the paper, and Section 3 introduces research applied in the threshold signature and self-sovereign identity environment related to this paper. Section 4 describes the proposed key recovery protocol proposed in the paper. Section 5 provides a step-by-step explanation of how the proposed key management framework operates in a self-sovereign identity environment. Next, we measure and verify the performance required for each step of the key recovery protocol proposed in Section 6. Finally, in the conclusion, the contents of the paper are summarized and concluded, and the direction of future development is discussed.

2. Background

This section briefly explains related technologies to help explain the proposed method. The proposed method can be divided into technology for interoperability of credentials in a self-sovereign identity environment and key management technology for the verification of credentials. First, in order to understand the interoperability technology of credentials in a self-sovereign identity environment, we explain the DID [2], which is decentralized identity information in a self-sovereign identity, and the VC model [3], which is a credential that becomes a digital object of identity information. Next, we briefly describe the Schnorr signature-based threshold signature algorithm FROST [22], which is a signature algorithm for verifying credentials.

2.1. Digital Identity [2]

Decentralized identifiers (DIDs) represent an innovative approach to identity verification in the digital world. These are unique global identifiers that are not managed by a central authority, but instead are distributed over a network. Blockchain, which uses a distributed network model to ensure the integrity of data transactions, is a representative example, and DID for identity management can be managed through this distributed network model. This decentralization enhances privacy and security by ensuring that control of identifiers belongs only to the user, the DID entity.

The structure of a DID is a string consisting of three essential parts: a URI scheme identifier (e.g., "did"), an identifier for the DID method (e.g., "example"), and a unique identifier specific to the DID method (e.g., "123abc") (Figure 1). This configuration (e.g., "did:example:123abc") is consistent with web standards and makes it easy to use on a variety of platforms. DID methods define how DIDs and their corresponding DID documents are created, interpreted, updated, and deactivated.

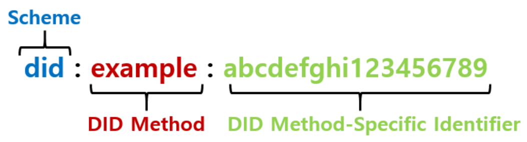
Scheme
did : example : abcdefghi123456789
DID Method
DID Method-Specific Identifier

Figure 1. A simple example of a decentralized identifier (DID) [2].

DID documentation is an important part of this system and contains the elements necessary to demonstrate control over the DID. These documents allow DID owners to assert their identity and, when combined with verifiable credentials (VC), establish and verify their credentials. The entire DID concept is overseen and standardized by W3C, the consortium responsible for global web standards, ensuring a consistent and interoperable approach across a variety of platforms and applications.

2.2. Verifiable Credentials Data Model [3]

Verifiable credential (VC) is a data object that contains the user's identity information, such as a driver's license or graduation certificate, and is a trustworthy identity information object that can prove one's qualifications digitally. As shown in Figure 2, VC is largely composed of three items: Credential Meta, which contains metadata about the VC such as issuing agency and expiration period, Claims, which is the information item that the user wants to prove, and Proof, which is used to verify that the VC is trustworthy.

Verifiable presentation (VP) is one or more verifiable credential objects that selectively extract and provide information items from VC to authenticate identity information digitally. For example, if you need to verify that you are an adult to purchase alcoholic beverages, you only need to provide your date of birth on your resident registration card VC. Using VP, user privacy can be preserved because information such as address and name is not disclosed.

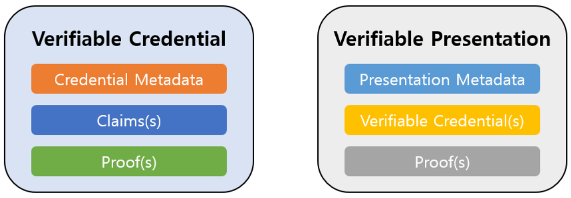

Figure 2. Verifiable Credential Data Model [3].

2.3. Frost Threshold Signature [22]

FROST is a DKG (distributed key generation) protocol that provides minimal communication rounds and provides secure threshold signatures in the Schnorr signature scheme while running in parallel. FROST contains two important components: first, n-participants run the DKG protocol to generate a common verification key, and then share the secret key to obtain the distributed key. Afterwards, all t-out-of-n participants can jointly generate a valid Schnorr signature by running the threshold signature protocol. In the creation process, each participant creates a partial signature and the representative combines the partial signatures to create a final signature. This final signature can be verified through a verification algorithm. Most threshold signature algorithms [13,23,24] require at least three rounds of transactions. However, the FROST signature algorithm can generate distributed keys through only two rounds of transactions [22]. Additionally, the reliability and security of the distributed key generated by the FROST signature algorithm has been verified in numerous studies [22,25].

In this study, it is assumed that threshold signing is performed using FROST. Therefore, it is assumed that users discuss their shares using the key generation method provided by FROST. The result of the key generation process is $s_{ID_u} = f(ID_u)$ for the implicitly shared polynomial, $f(x) = a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \cdots + a_0$, assuming a (t,n) threshold signing environment when the user's ID is ID_u. Here, the coefficient a_i $(i = 0, 1, \cdots, t-1)$ of the polynomial $f(x)$ is an undisclosed value.

The FROST signature algorithm combines the Schnorr signature [26] and Shamir's secret sharing algorithm [27] to implement distributed key generation and a threshold signature. This is achieved through the following process.

1. Distributed key generation:
   Multiple participants jointly run the distributed key generation (DKG) protocol to generate private and public keys. Using Shamir's secret sharing algorithm, the private key is divided into several pieces and distributed to each participant.
2. Threshold signature generation:
   At least t participants collaborate to generate a threshold signature. Each participant generates partial signatures and combines them to generate the final signature. In this case, Shamir's secret sharing algorithm is used to securely share the secret required for the threshold signature.

The Schnorr Signature Algorithm [26]

The Schnorr signature algorithm, which forms the basis of the FROST signature algorithm [22], is defined as follows. Let $\mathbb{G}$ be an elliptic curve group of order q with base point (generator) G. The private key (secret key) is a random value $s \in \mathbb{Z}_q$ and the public key is $Y = g^s \in \mathbb{G}$. The Schnorr signing operation on a message $m \in \{0,1\}^*$ is defined as follows:

1. Sample a random nonce: $k \leftarrow \mathbb{Z}_q$;
2. Compute the commitment: $R = g^k \in \mathbb{G}$;
3. Compute the challenge: $c = H(R, Y, m)$;
4. Compute the response: $z = k + s \cdot c \in \mathbb{Z}_q$;
5. Define the signature over m to be $\sigma = (R, z)$;
6. Compute result: $R' = g^z \cdot Y^{-c}$.

3. Related Work

This section describes related work. The first subsection describes the research related to the threshold signature system used in this study. The second subsection describes research related to sovereign identity environments. The final subsection describes the problems and limitations of existing research.

3.1. Research on Threshold Signature Systems

Related studies on an efficient threshold signature system based on ECDSA are as follows. Doerner, Jack, et al. proposed a multi-party threshold signature algorithm with two thresholds based on the ECDSA encryption algorithm [12]. Research by R Gennaro and S Goldfeder [17] proposes a threshold signature that can support multiple participants through an optimal threshold protocol that improves the signature efficiency and flexibility of multi-party computation in the existing ECDSA threshold signature system.

Recently, in the blockchain and cryptocurrency fields, interest in Schnorr signatures and their derived algorithms for creating multiple signatures has increased significantly. Crites, Elizabeth, and Mary Maller's research [23] developed sparkle, a round three-stage Schnorr critical signature system, and demonstrated multiple levels of security and adaptive security according to various corruption models and assumptions. Dufka, A., and Sedlacek's research [14] developed a signature system using nonce to make Schnorr signature-based systems compatible and interoperable to defend against security threats that may occur in various situations. Boneh, Dan et al.'s research [15] implements a new aggregate signature scheme derived from Schnorr signatures and BLS signatures, which can compress n signatures into a single short signature using a multi-signature scheme. Y Lindell's research [24] proposed a simple three-round multi-party signature protocol that can be fully simulated in terms of efficiency, security, and standard definition.

In the study by Ostrovsky and Yung [18], the secret keys of the threshold system are distributed and stored to prevent a single point of failure, but the security occurs the moment t secret information at the threshold level of (t, n) is exposed. To avoid such attacks, various critical signature systems with key renewal or key recovery are being studied [13,16,17,28–32]. The work of Canetti, R., Gennaro, R., and Goldfeder, S. [30] proposed a newly designed refresh protocol using Paillier encryption and zero-knowledge proofs, but currently this refresh protocol uses an (n, n) threshold. It is provided only for ECDSA and does not provide a refresh function for (t, n) threshold ECDSA schemes where $t < n$. In addition, K. Bae, J. Park, and J. Ryou's research [13] proposed a DID system applied to proof of verifiable presentation using the two-party threshold ECDSA signature algorithm proposed by Doerner. This showed that the ECDSA-based threshold signature algorithm is applicable to the self-sovereign identity environment, but the (t, n) threshold setting is fixed to (2, n), so the threshold setting is not flexible.

3.2. Research on Self-Sovereign Identity

Recently, a number of studies have been conducted showing that the threshold signature algorithm can be applied to blockchain or self-sovereign identity [15,29–34]. A study by S. Ricci et al. [31] showed that it is possible to operate on a blockchain by dividing distributed keys among multiple devices using a Schnorr-based threshold signature system. Soltani, Reza et al.'s study [32] mentions a key backup and recovery approach utilizing self-sovereign identity-based trusted third-party storage.

In self-sovereign identity, credentials are an essential part [35], and user privacy can be strengthened by applying a threshold signature system at the issuance and submission stage of credentials [31–33]. Doerner, J., Kondi et al.'s [33] research proves that user anonymity can be protected in a self-sovereign identity environment by applying threshold signatures based on the BBS+ signature algorithm to credentials. J. Kim et al. [34] proposed an ECDSA-based threshold signature method to verify the signature of a presentation derived through credentials. It was shown that through this method, the identity of the signer cannot be identified through the credentials requested for verification, and privacy can be protected. Research by Zhiji Li [36] proposed a credential framework based on BLS aggregate signatures that can selectively disclose information by signing detailed claims of the credential instead of signing the entire credential.

3.3. Shortcomings of Existing Architecture

The self-sovereign identity environment is a new technology field, and research in combination with signature technology is being actively conducted, but existing architectures have some limitations. In Table 1, the technical scope provided by existing studies is compared with this study. Threshold signature algorithms are largely divided into Schnorr, ECDSA, and BBS+. Each signature system has the number of communication rounds for generating and distributing distributed keys (DKG Round), the existence of key recovery and renewal functions (Proactive Security), the ability to change threshold settings (t-Config Flexibility), and self-sovereign identity. We compare each study by classifying it by applicability to the environment (SSI Applicability).

Table 1. Comparison of Distributed Signature Scheme's function.

	Sign Algorithm	DKG Round	t-Config Flexibility	Proactive Security	SSI Applicability
Ours	Schnorr	2	O	O	O
[23,24]	Schnorr	3	O	X	X
[29,31]	Schnorr	3	O	X	O
[13]	ECDSA	3	X	O	O
[16,30]	ECDSA	3	O	O	X
[34]	ECDSA	3	X	X	O
[33]	BBS+	3	O	X	O

First, the ECDSA series algorithm is very fast for a single operation, but many studies have been conducted on three or more communication rounds [13,16,30]. A communication round is important because it is greatly influenced by the network environment in order to exchange confidential information between participants. This study provides two communication rounds through a communication round optimization study of the Schnorr-based signature algorithm.

Second, most threshold signatures are implemented to allow changes in the threshold (t) and maximum number of participants (n), but algorithms with fixed threshold settings, such as [13], lack flexibility. While some research has been conducted with the threshold setting of the signature system fixed, such as (2, n) or (3, n), this study provides flexibility in threshold setting to respond to various conditions and situations.

Third, key recovery or renewal is essential when the distributed key is stolen at a threshold level [18], and research supporting proactive functionality has been conducted in this study. Some research [16,30] has been done, but most do not consider the security threat.

Lastly, although most of the current research focuses on the theoretical application and algorithm development of threshold signatures, there is a relative lack of research on practical interoperability methodologies needed to integrate them into real systems or user environments.

This study proposes a more flexible and efficient distributed key management method to strengthen the self-sovereign identity environment. To this end, we increase the flexibility of the threshold setting and improve the efficiency based on FROST, a Schnorr-based threshold signature algorithm. In addition, we propose an implementation method that is scalable, including a distributed key recovery protocol, and takes interoperability into consideration in the use environment.

4. Key Recovery Protocol

This section describes the proposed key recovery protocol. We propose a protocol that allows users to recover their own keys when they are lost for keys shared between the User, TTP, and Backup Device using FROST. When the proposed method shares a prime number q with a very long bit length as a parameter, the proposed method assumes that the user has a User's ID ID_u, TTP's ID ID_{ttp}, and Backup Device's ID ID_{bd} (ID_u, ID_{ttp}, $ID_{bd} \in \mathbb{Z}_q$) and shares the (3,2) threshold signing key through FROST. In this case, the three participants effectively have a random polynomial $f(x) = a_1 x + a_0$ where a_i is selected uniformly from $\mathbb{Z}_q$ $(i = 0, 1)$. We let $s_{ID_u} = f(ID_u)$, $s_{ID_{ttp}} = f(ID_{ttp})$, and $s_{ID_{bd}} = f(ID_{bd})$, respectively. As an additional assumption, TTP has $\phi_i = g^{a_i} \mod p$ where p is a prime and g is a generator in $\mathbb{Z}_p^*$ $(i = 0, 1)$, whose order is q. This ϕ_i can be shared when the FROST protocol ends.

Next, in the user's key loss scenario, in this case, the user is losing his share $f(ID_u)$. Therefore, the proposed key recovery protocol operates through the following process. Assume that the integrity and confidentiality of communication between each user is guaranteed through individual encryption and a digital signature:

1. User generates his/her new ID $ID'_u \in \mathbb{Z}_q$ and transmits it to TTP.
2. TTP selects α from uniform distribution over $\mathbb{Z}_q$ and delivers α and ID'_u to the Backup Device.
3. TTP computes $a_{ttp} \leftarrow L_{ID_{ttp}}(ID'_u) \times s_{ID_{ttp}} + \alpha (\mod q)$ and delivers it to User, where $L_{ID_{ttp}}(x) = (ID_{bd} - x)/(ID_{ttp} - ID_{bd}))(\mod q)$.
4. Backup Device computes $a_{bd} \leftarrow L_{ID_{bd}}(ID'_u) \times s_{ID_{bd}} - \alpha (\mod q)$ and provides it to the user, where $L_{ID_{bd}}(x) = (x - ID_{ttp})/(ID_{ttp} - ID_{bd}))(\mod q)$.
5. User creates a new share for his/her ID'_u using $s_{ID'_u} \leftarrow a_{ttp} + a_{bd} (\mod q)$.
6. TTP delivers ϕ_1, ϕ_0 to User.
7. User computes $(\phi_1)^{s_{ID'_u}} * \phi_0 (\mod p)$ and $g^{s_{ID'_u}} \mod p$. He/she verifies both values are the same.

Correctness Check: The share that user receives through the above protocol is $L_{ID_{ttp}}(ID'_u) \times s_{ID_{ttp}} + L_{ID_{bd}}(ID'_u) \times s_{ID_{bd}} (\mod q)$, and since this is Lagrange Interpolation, it becomes $f(ID'_u)$. ($\because L_{ID_{ttp}}(ID'_u) \times s_{ID_{ttp}} + L_{ID_{bd}}(ID'_u) \times s_{ID_{bd}} \mod q = (ID_{bd} - ID'_u)/(ID_{ttp} - ID_{bd})(a_1 ID_{ttp} + a_0) + (ID'_u - ID_{ttp})/(ID_{ttp} - ID_{bd})(a_1 ID_{bd} + a_0) = a_1 ID'_u + a_0 = f(ID'_u)) \mod q$ and $\phi_1 = g^{a_1} \mod p$ and $\phi_0 = g^{a_0} \mod p$, so $(\phi_1)^{s_{ID'_u}} \times \phi_0 (\mod p) = g^{a_1 \times s_{ID'_u} + a_0} (\mod p)$. Therefore, if $f(ID'_u)$ is the correct value, it becomes the same value as $g^{f(ID'_u)} (\mod p)$.

Security Analysis: It must be shown that each participant cannot obtain other participants' share information without the help of other participants. In the proposed protocol, the only participant who receives information related to other participants' shares is the User. Therefore, the User should not be able to know the share of the TTP or Backup Device from the information he or she receives. For this purpose, TTP and Backup Device share random number information α and use it to protect their share information.

5. Proposed Framework

In this section, we propose a key management framework in a DID environment using the key recovery protocol proposed in Section 4. The proposed framework consists of four protocols: key generation and distribution, user public key registration and verification with the Relying Party (RP), distributed key recovery using the key recovery protocol in Section 4, and finally, distribution key change and destruction.

5.1. Distribution Key Generation and Distribution

Figure 3 below shows the process in which the User, TTP, and Backup Device generate and store distributed keys for the critical signature. At this time, TTP plays a leading role and relays communication between the User (=Identity Holder) and Backup Device. A brief description of each procedure is as follows. When the User, the credential holder, requests a trusted third party (TTP) to generate a distributed key, all participants generate their own share and public key using FROST's key generation and distribution protocol. In Figure 3, in Step 1, when the user requests TTP to generate a key, in Step 2, each participant has their share using the FROST key distribution protocol. In Step 3, information to verify the received shares is received, and then verification is performed in Step 4.

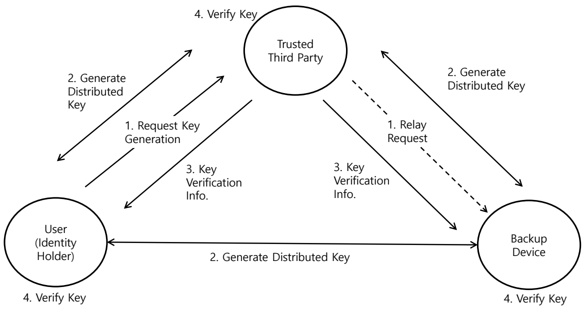
4. Verify Key
Trusted Third Party
2. Generate Distributed Key
1. Request Key Generation
3. Key Verification Info.
3. Key Verification Info.
1. Relay Request
2. Generate Distributed Key
User (Identity Holder)
2. Generate Distributed Key
Backup Device
4. Verify Key
4. Verify Key

Figure 3. Key Generation and Distribution Step.

5.2. Public Key Registration and Authentication

Figure 4 below shows the User's public key registration process (A) and the process of using the registered public key for authentication (B) in the DID environment. In the case of the public key registration process (A), in Step 1, the User requests the TTP for the threshold signature used to generate the public key registration request message, and the TTP delivers the partial signature signed with its share to the User. In Step 3, the User creates a complete threshold signature using the partial signature received from TTP, attaches all PKs (Public Keys), and delivers a registration request message to the RP (Relying Party). In Step 4, the RP verifies the signature using the received public key, and if verification is successful, it sends a registration success message to the User.

In the authentication process (B) using the registered public key, in Step 1, the User sends a login request message including the User's threshold signature to the SP (Service Provider) who wants to log in. The signature creation process is assisted by TTP. The SP delivers the request message to the RP, and the RP verifies the threshold signature in the request message with the registered public key and then delivers the result to the SP. When verification is successful, the SP delivers a login success message to the User.

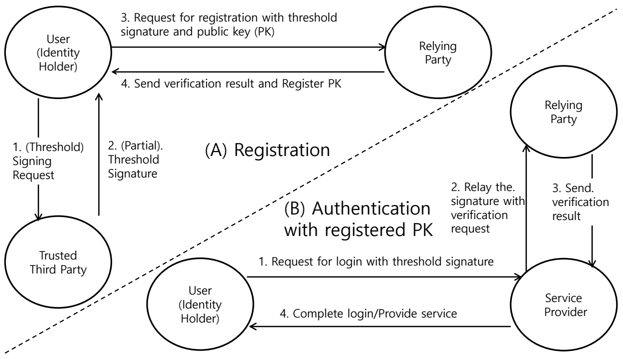
User (Identity Holder)
3. Request for registration with threshold signature and public key (PK)
Relying Party
4. Send verification result and Register PK
1. (Threshold) Signing Request
2. (Partial). Threshold Signature
(A) Registration
Trusted Third Party
Relying Party
2. Relay the. signature with verification request
3. Send. verification result
(B) Authentication with registered PK
1. Request for login with threshold signature
User (Identity Holder)
4. Complete login/Provide service
Service Provider

Figure 4. Key Registration to Relying Party.

5.3. Distributed Key Recovery

Figure 5 below shows the procedure for Users to recover their keys if the distributed key is damaged or lost. TTP, which received the request message in Step 1, forwards the request to the BD (Backup Device) and delivers the recovered key to the User through the protocol proposed in Section 4. Step 3 in Figure 5 means performing the protocol in Section 4.

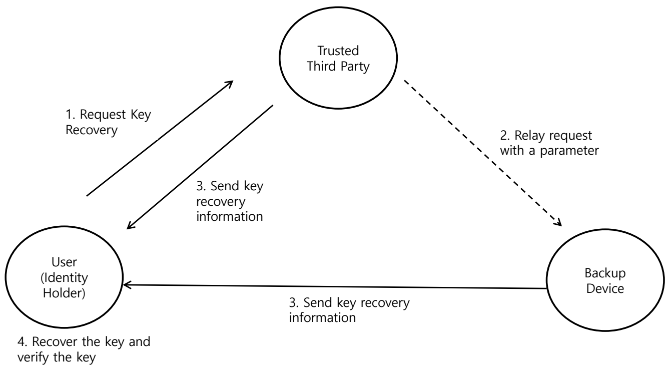
Trusted Third Party
1. Request Key Recovery
2. Relay request with a parameter
3. Send key recovery information
User (Identity Holder)
Backup Device
3. Send key recovery information
4. Recover the key and verify the key

Figure 5. Key Recovery Step.

5.4. Renew and Discard Public Key

Figure 6 shows the process of registering the changed public key with the RP and discarding the existing public key when the User changes the public key through generation of a new public key. It is used when registering an updated public key when the User has already registered the old public key through the transfer process. In order to perform this work, the following must be provided first. When renewing a public key, the TTP and Backup Device must maintain the public key before renewal and the previously shared shares corresponding to the public key. The period for maintaining these public keys should be long enough so that all RPs have updated public keys. Additionally, the RP also has a period of time to maintain the registered public key, so when the period expires, the registered public key is removed and the user must register a new public key with the RP.

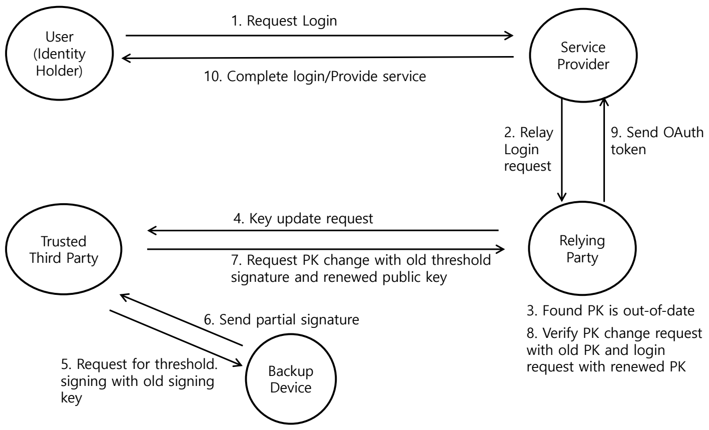
User (Identity Holder)
1. Request Login
10. Complete login/Provide service
Service Provider
2. Relay Login request
9. Send OAuth token
Trusted Third Party
4. Key update request
7. Request PK change with old threshold signature and renewed public key
Relying Party
6. Send partial signature
5. Request for threshold. signing with old signing key
Backup Device
3. Found PK is out-of-date
8. Verify PK change request with old PK and login request with renewed PK

Figure 6. Key Renewal for Relying Party.

Under the above assumptions, the proposed public key change and destruction protocol operates as shown in Figure 6. In Step 1, 1$\sim$3 represents a normal login request and failure process. Since the RP has an old public key, authentication cannot succeed because the signature verification process fails. In this case, if the key ID in the request is different from the key ID stored when registering the public key, the RP recognizes that the key has been updated and sends a public key renewal request to the User to the TTP (Step 4). Upon receiving this request, the TTP cooperates with the Backup Device to create a verifiable threshold signature for the key renewal request message with the old public key (Step 5$\sim$6), and delivers the generated message, signature, and the User's new PK to the RP (Step 7). The RP verifies the signature with the currently registered PK and updates it with the newly delivered PK when verification is successful. Afterwards, the login request received in Step 2 is verified with the updated PK, and if verification is successful, a login success message is delivered to the User as in Step 10.

6. Analysis and Comparison

In this section, we discuss the computation measurement and analysis results of the proposed key recovery protocol. We additionally implemented a key recovery protocol based on the distributed key generation and distribution implementation source [37] of FROST, a Schnorr-based threshold signature system, and confirmed that they are efficient and fast enough for actual use. In the implementation, we used the secp256k1 elliptic curve whose group size is 256 bit, and it is implemented so that the settings of the number of participants n and the threshold t can be dynamically changed. The details of the experiment of the proposed framework are divided into (1) heavy time performance of the proposed framework, (2) comparison with the performance of recovery backup, and (3) computation traffic measurement. The pseudocode for performing the specific experiment is given in Algorithm 1.

Algorithm 1 KEY RECOVERY

```
1:  procedure KEY RECOVERY($P1 : User$)
2:      $Q \leftarrow FROST.secp256k1.Q$
3:      $G \leftarrow FROST.secp256k1.G()$
4:      $al \leftarrow secrets.randombit(256)\ \%\ Q$
5:      $mal \leftarrow (Q - al)\ \%\ Q$
6:
7:      Init dummy participant P4
8:      Calculate P1's partial share: $sh1$
9:      Calculate P2's partial share: $sh2$
10:     Compute P4 combine partial share(sh1, sh2)
11:
12:     Calculate P1's lagrange coefficient: $l1$
13:     Calculate P4's lagrange coefficient: $l4$
14:     Compute new P4's secret: $secret4$
15:         $secret4 \leftarrow ((p1.share \times l1) + (p4.share \times l4))\ \%\ Q$
16:
17:     Compute verify new secret: $self.assertEqual(secret4 \times G,\ pk)$
18:     if $result = True$ then
19:         Set P1's secret $\leftarrow$ P4's secret
20:     end if
21: end procedure
```

First, in the proposed framework, we measure the time taken for the generation/ distribution phase of the segregated key and the key recovery and signing phase using the recovery key.

Second, we compare its performance with Kim, Myungsun, et al.'s study [16] and Herzberg, Amir, et al.'s [28] study, studies that perform key recovery using a secret sharing technique similar to the proposed method.

Finally, we discuss the results of measuring the computation overhead that occurs between participants as the proposed framework is implemented.

The experiment is conducted in a Google Cloud Platform environment using an Intel(R) Xeon(R) CPU @ 2.80 GHz, 32 GB RAM, and CentOS 7.

6.1. Execution Time of Each Step of the Proposed Framework

The proposed framework (refer to Section 5) consists of a total of 4 steps, and the operation time for each step was measured and summarized in Table 2. The following is a brief explanation for the case (n = 3, t = 2). The first step is Distribution Key Generation and Distribution, which takes 0.74 s to generate and distribute distribution keys between participants. However, network communication time for secret sharing between participants is excluded, and only data transmission time between participants within the same instance is included. The second step is the Public Key Registration and Authentication step. Among the distributed signatures of n participants, the number of thresholds t is collected and a combined signature is generated. Afterwards, the combined signature and public key are verified to obtain the result. The time taken at this time is 0.28 s. The third step is the Distributed Key Recovery step, where the most core key recovery protocol is performed. This step changes only the private key of a specific participant without changing the public key, and takes 0.002 s. Since the secret share of the participants (TTP and Backup Device) is used when generating the initial key, the computational time required to recover the key is extremely minimal. This indicates that the recovery protocol works very quickly, and this fast recovery time contributes to the stability and reliability of the system. In the case of the last step, Renew and Discard Public key, it is excluded from the performance measurement because it is a procedural part of the SSI environment rather than a signature protocol and is affected by the network environment.

Table 2. Execution time of each step of the proposed framework for various n conditions.

Participants (t = 2)	Key Generation (Section 5.1)	Threshold Sign (Section 5.2)	Key Recovery (Section 5.3)
n = 3	0.74 s	0.28 s	0.002 s
n = 5	1.23 s	0.27 s	0.002 s
n = 10	19.69 s	0.28 s	0.002 s

n: num of participant, t: threshold.

The analysis results confirm that the FROST-based threshold signature scheme can operate at practical speeds while maintaining reliability and security. This is especially important when rapid key destruction and recovery is required in urgent situations, such as when a critical mass of distributed keys has been stolen. This is an essential characteristic for safely managing important assets in distributed systems such as blockchain.

6.2. Performance Comparison of Key Recovery Protocols

To understand our key recovery protocol described in Section 4, we illustrate an example using Figure 5. Assume that there are three participants, User (P1, Identity Holder), Trusted Third Party (P2), and Backup Device (P3), and that User has lost the distributed secret key sk1. To recover the User's (P1) distributed secret key, proceed with the following procedures.

The verification results for the key recovery protocol based on the FROST signature algorithm are as follows. Among various (t, n) threshold settings, we perform computation comparison for (2, 3) threshold settings, which can be compared with related work [16,28]. Our experimental results were performed 100 times, and Figure 7 is a graph comparing the average of the execution time with the experimental results of related studies. The time required, which is a data label, is the accumulated value of the time required for each step.

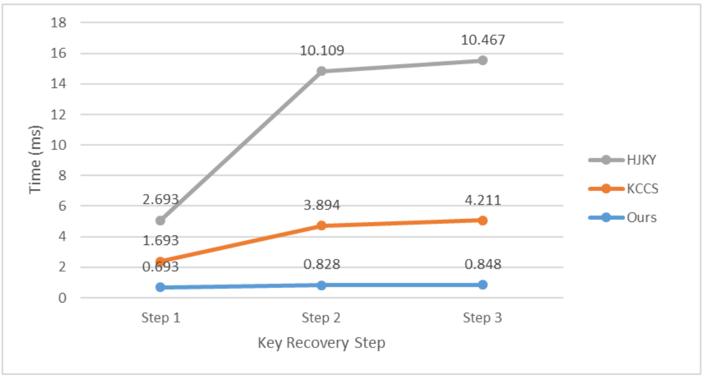

Figure 7. Comparison of experimental results of the proposed protocol and other protocols (Unit: ms) [16]: KCCS, [23]: HJKY.

6.3. Computation and Communication Overhead Analysis

In terms of the communication and computation overhead, the threshold signature generation mechanisms of the existing threshold distributed signature scheme and the proposed scheme are almost equivalent. This is because distributed signatures require interaction between participants to exchange secret information and create a combined signature. The proposed threshold signature system is a Schnorr-based signature system that can generate a threshold signature in two rounds through round optimization. In the first round, we generate a commitment r_i for each participant. In the second round, shares, which are secret information, are created, and verified shares are combined through exchange. At this time, a public key is generated, and the process ends by sharing the public key with each participant. The computation overhead is divided into each round for the distributed key generation and distribution stages, and the computational cost of the key recovery stage is analyzed. The participant information exchanged at each stage is 56 bytes and includes the public key and individual secret information. The commitment generated through calculation is created in 96 to 128 bytes (Table 3).

Table 3. Computation Overhead of DKG and Key Recovery.

	DKG Round 1	**DKG Round 2**	**Key Recovery**	**Size**
Computation	n^2	$2n^2 + n$	$n(n-1) + 2$	Participant.info: 56 byte Commitment: 96∼128 byte Public key: 16 byte

n: num of participant.

In addition, the proposed threshold signature algorithm requires a process in which everyone who wishes to participate in the signature mechanism must generate and share public keys and personal secret information in advance. Therefore, because the precomputation process is based on mathematical hardness, the computational overhead increases compared to the ECDSA signature algorithm.

6.4. Proposed Framework Security Analysis

We discuss the security of the proposed framework in this section. The FROST signature algorithm used to protect confidential information in the proposed framework has been verified in studies [22,38] for proof of security against selected message attacks and cryptographic security against ROS attacks. In more detail, the security of each detailed procedure of the framework proposed in Section 5 is explained as follows.

First, in Section 5.1, a data transaction occurs between n participants to generate a distributed key, which satisfies security through the secret sharing technique in FROST [22]. Second, in Sections 5.2 and 5.4, a variety of data transactions, including signing and verifications, occur during the process of registering and discarding the Relying Party and public key. At this time, since the threshold signature is required in the process of registering and discarding the public key, security is not violated even if the secret information of t-1 threshold signatures is stolen. In addition, the DIDComm v2.0 protocol [39], a secure communication protocol within the WACI protocol [21], operates for communications required between users and verification agencies such as Relying Party to enhance security. Lastly, Section 5.3 of the proposed framework is the key recovery step, and cryptographic security is satisfied based on the FROST algorithm [22].

7. Conclusions

This paper recognizes the need for efficient digital identity and private key management in a self-sovereign identity (SSI) environment and presents a new approach to support this. We utilized FROST, a threshold signature scheme based on Schnorr signatures, to develop a key management scheme including distributed key renewal and recovery protocols to increase the reliability and efficiency of key management. This approach promotes the interoperability and anonymity of credentials and focuses on overcoming the limitations of traditional threshold signature schemes while improving privacy protection and interoperability of digital identity management systems. In addition, based on previous research [19] applying the Universal Wallet standard and WACI protocol to strengthen the interoperability of digital wallets, we applied the proposed framework by identifying actual applicable cases and established the FROST threshold for credential verification.

These research results will contribute to the development of a self-sovereign identity environment and will serve as an important reference in the design and implementation of future digital identity management systems.

However, the FROST algorithm we used to protect confidential information is not post-quantum secure because it is based on Shamir's secret sharing technique and Elliptic Curve Digital Signature Algorithm. We leave it as future work to apply algorithms such as lattice encryption and quantum encryption that can solve this problem.

Author Contributions: Conceptualization, P.K. and J.K.; methodology, J.K. and Y.L.; software, J.K.; validation, Y.L.; data curation, P.K.; writing—original draft preparation, P.K. and J.K.; writing—review and editing, P.K. and Y.L.; supervision, Y.L. and D.C.; project administration, Y.L. and D.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean Government (MSIT) (No. 2021-0-00565, Development of user identity certification and management technology for self-sovereign identity applications).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

References

1. SOVRIN. What Is Self-Sovereign Identity? Available online: https://sovrin.org/faq/what-is-self-sovereign-identity/ (accessed on 30 October 2023).
2. W3C Recommendation; Sporny, M.; Longley, D.; Sabadello, M.; Reed, D.; Steele, O.; Allen, C. Decentralized Identifiers (DIDS) V1.0. Core Architecture, Data Model, and Representations. 2021. Available online: https://www.w3.org/TR/did-core/#dfn-decentralized-identifiers (accessed on 30 December 2023).
3. MIT; ERCIM; Keio, B. Verifiable Credentials Data Model v1.1. W3C. 2021. Available online: https://www.w3.org/TR/vc-data-model/ (accessed on 3 March 2022).
4. Yildiz, H.; Küpper, A.; Thatmann, D.; Göndör, S.; Herbke, P. Towards Interoperable Self-sovereign Identities. *IEEE Access* **2023**, *11*, 114080–114116 [CrossRef]
5. Čučko, Š.; Bećirović, Š.; Kamišalić, A.; Mrdović, S.; Turkanović, M. Towards the classification of Self-Sovereign Identity properties. *IEEE Access* **2022**, *10*, 88306–88329. [CrossRef]
6. Teuschel, M.; Pöhn, D.; Grabatin, M.; Dietz, F.; Hommel, W.; Alt, F. 'Don't Annoy Me with Privacy Decisions!'—Designing Privacy-Preserving User Interfaces for SSI Wallets on Smartphones. *IEEE Access* **2023**, *11*, 131814–131835. [CrossRef]
7. Naik, N.; Jenkins, P. Self-sovereign identity specifications: Govern your identity through your digital wallet using blockchain technology. In Proceedings of the 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), Oxford, UK, 3–6 August 2020; pp. 90–95.
8. Gennaro, R.; Jarecki, S.; Krawczyk, H.; Rabin, T. Secure distributed key generation for discrete-log based cryptosystems. In Proceedings of the Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; Proceedings 18; Springer: Berlin/Heidelberg, Germany, 1999; pp. 295–310.
9. Li, C.M.; Hwang, T.; Lee, N.Y. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In Proceedings of the Advances in Cryptology—EUROCRYPT'94: Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, 9–12 May 1994; Proceedings 13; Springer: Berlin/Heidelberg, Germany, 1995; pp. 194–204.
10. Steele, O.VC Proof Formats Test Suite—VC Data Model with JSON Web Signatures. Available online: https://identity.foundation/JWS-Test-Suite/ (accessed on 30 December 2023).
11. Desmedt, Y. Threshold cryptosystems. In Proceedings of the International Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Australia, 13–16 December 1992; Springer: Berlin/Heidelberg, Germany, 1992; pp. 1–14.

12. Doerner, J.; Kondi, Y.; Lee, E.; Shelat, A. Secure two-party threshold ECDSA from ECDSA assumptions. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–23 May 2018; pp. 980–997.
13. Bae, K.; Park, J.; Ryou, J. Secure Recovery Protocol of (1, 3) Distributed Key Share with Trustless Setup for Asset Management in Blockchain. *J. Korea Inst. Inf. Secur. Cryptol.* **2021**, *31*, 863–874.
14. Dufka, A.; Sedlacek, V.; Svenda, P. SHINE: Resilience via Practical Interoperability of Multi-party Schnorr Signature Schemes. In Proceedings of the Secrypt: Proceedings of the 19th International Conference on Security and Cryptography, SCITEPRESS, Lisbon, Portugal, 11–13 July 2022; pp. 305–316.
15. Boneh, D.; Drijvers, M.; Neven, G. Compact multi-signatures for smaller blockchains. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, Australia, 2–6 December 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 435–464.
16. Kim, M.; Cho, S.; Choi, S.; Cho, Y.S.; Kim, S.; Lee, H.T. A Key Recovery Protocol for Multiparty Threshold ECDSA Schemes. *IEEE Access* **2022**, *10*, 133206–133218. [CrossRef]
17. Gennaro, R.; Goldfeder, S. Fast multiparty threshold ECDSA with fast trustless setup. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1179–1194.
18. Ostrovsky, R.; Yung, M. How to withstand mobile virus attacks. In Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing, Montreal, QC, Canada, 19–21 August 1991; pp. 51–59.
19. Kim, J.; Kim, P.; Choi, D.; Lee, Y. A Study on the Interoperability Technology of Digital Identification Based on WACI Protocol with Multiparty Distributed Signature. *Sensors* **2023**, *23*, 4061. [CrossRef] [PubMed]
20. Steele, O.; Johnson, M. Guillaume Dardelet "Universal Wallet 2020". Available online: https://w3c-ccg.github.io/universal-wallet-interop-spec/ (accessed on 15 January 2024).
21. Aman, A.; Hedges, E. Wallet and Credential Interactions. DIF. 2022. Available online: https://identity.foundation/wallet-and-credential-interactions/ (accessed on 16 January 2024).
22. Komlo, C.; Goldberg, I. FROST: Flexible round-optimized Schnorr threshold signatures. In *Proceedings of the Selected Areas in Cryptography: 27th International Conference, Halifax, NS, Canada (Virtual Event), 21–23 October 2020, Revised Selected Papers 27*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 34–65.
23. Crites, E.; Komlo, C.; Maller, M. Fully Adaptive Schnorr Threshold Signatures. Cryptology ePrint Archive. 2023. Available online: https://eprint.iacr.org/2023/445 (accessed on 15 January 2024).
24. Lindell, Y. Simple Three-Round Multiparty Schnorr Signing with Full Simulatability. Cryptology ePrint Archive. 2022. Available online: https://eprint.iacr.org/2022/374 (accessed on 15 January 2024).
25. Bellare, M.; Crites, E.; Komlo, C.; Maller, M.; Tessaro, S.; Zhu, C. Better than advertised security for non-interactive threshold signatures. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–18 August 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 517–550.
26. Schnorr, C.P. Efficient signature generation by smart cards. *J. Cryptol.* **1991**, *4*, 161–174. [CrossRef]
27. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
28. Herzberg, A.; Jarecki, S.; Krawczyk, H.; Yung, M. Proactive secret sharing or: How to cope with perpetual leakage. In Proceedings of the Advances in Cryptology—CRYPT0'95: 15th Annual International Cryptology Conference, Santa Barbara, CA, USA, 27–31 August 1995; Proceedings 15; Springer: Berlin/Heidelberg, Germany, 1995; pp. 339–352.
29. Maxwell, G.; Poelstra, A.; Seurin, Y.; Wuille, P. Simple schnorr multi-signatures with applications to bitcoin. *Des. Codes Cryptogr.* **2019**, *87*, 2139–2164. [CrossRef]
30. Canetti, R.; Makriyannis, N.; Peled, U. Uc Non-Interactive, Proactive, Threshold Ecdsa. Cryptology ePrint Archive. 2020. Available online: https://eprint.iacr.org/2020/492 (accessed on 15 January 2024).
31. Ricci, S.; Dzurenda, P.; Casanova-Marqués, R.; Cika, P. Threshold Signature for Privacy-Preserving Blockchain. In Proceedings of the International Conference on Business Process Management, Münster, Germany, 11–16 September 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 100–115.
32. Soltani, R.; Nguyen, U.T.; An, A. Practical key recovery model for self-sovereign identity based digital wallets. In Proceedings of the 2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Fukuoka, Japan, 5–8 August 2019; pp. 320–325.
33. Doerner, J.; Kondi, Y.; Lee, E.; Shelat, A.; Tyner, L. Threshold bbs+ signatures for distributed anonymous credential issuance. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–25 May 2023; pp. 773–789.
34. Kim, J.; Kim, G.; Ryou, J. Blockchain based DID System Design using Threshold Signature Scheme. In Proceedings of the Symposium of the Korean Institute of Communications and Information Sciences (KICI), Seoul, Republic of Korea, 6 October 2022; pp. 180–181.
35. Mühle, A.; Grüner, A.; Gayvoronskaya, T.; Meinel, C. A survey on essential components of a self-sovereign identity. *Comput. Sci. Rev.* **2018**, *30*, 80–86. [CrossRef]
36. Li, Z. A verifiable credentials system with privacy-preserving based on blockchain. *J. Inf. Secur.* **2022**, *13*, 43–65. [CrossRef]
37. FROST-BIP340. Available online: https://github.com/jesseposner/FROST-BIP340 (accessed on 28 December 2023).

38. Leng, J.; Zhou, M.; Zhao, J.L.; Huang, Y.; Bian, Y. Blockchain security: A survey of techniques and research directions. *IEEE Trans. Serv. Comput.* **2020**, *15*, 2490–2510. [CrossRef]
39. Curren, S.; Looker, T.; Terbu, O. DIDComm Messaging v2.x Editor's Draft. DIF. Available online: https://identity.foundation/didcomm-messaging/spec/ (accessed on 16 January 2024).