applied sciences

MDPI

Article

An Efficient Distributed Identity Selective Disclosure Algorithm

Guanzheng Wang [1] and Guoyan Zhang [1,2,*]

[1] School of Cyber Science and Technology, Shandong University, Qingdao 266237, China; 202237097@mail.sdu.edu.cn
[2] Shandong Institute of Blockchain, Jinan 250101, China
* Correspondence: guoyanzhang@sdu.edu.cn

Featured Application

Distributed identity management and privacy protection for user identities and electrical devices.

Abstract

Distributed digital identity is an emerging identity management technology aimed at achieving comprehensive interconnectivity between digital objects. However, there is still the problem of privacy leakage in distributed identities, and selective disclosure technology partially solves the privacy issue in distributed identities. Most of the existing selective disclosure algorithms use anonymous credentials or hash functions. Anonymous credential schemes offer high security and meet the requirements of unforgeability and unlinkability, but their exponential operations result in low efficiency. The scheme based on hash functions, although more efficient, is susceptible to man-in-the-middle attacks. This article proposes an efficient selective disclosure scheme based on hash functions and implicit certificates. The attribute values are treated as leaf nodes of the Merkle tree, and the root node is placed in a verifiable credential. According to the implicit certificate algorithm process, a key pair that can use the credential is generated. During the attribute disclosure process, the user autonomously selects the attribute value to be presented and generates a verification path from the attribute to the root node. The verifier checks the Merkle tree verification path. All operations are completed within 10 ms while meeting the unforgeability requirements and resisting man-in-the-middle attacks. This article also utilizes the ZK-SNARK algorithm to hide the validation path of the Merkle tree, enhancing the security of the path during the disclosure process. The experimental results show that the selective disclosure algorithm performs well in both performance and privacy protection, with an efficiency 80% faster than that of existing schemes. This enhances the proposed scheme's potential and value in the field of identity management; it also holds broad application prospects in fields such as the Internet of Things, finance, and others.

Keywords: selective disclosure; distributed identity; implicit certificate; merkle tree

check for updates

Academic Editors: Stefan Fischer and David Megías

Received: 13 January 2025
Revised: 14 February 2025
Accepted: 19 February 2025
Published: 11 August 2025

Citation: Wang, G.; Zhang, G. An Efficient Distributed Identity Selective Disclosure Algorithm. *Appl. Sci.* **2025**, *15*, 8834. https://doi. org/10.3390/app15168834

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The rapid development of blockchain technology has demonstrated its enormous potential in fields such as finance, supply chain management, and digital identity verification. Against this backdrop, on-chain and off-chain trustworthy interaction technology has emerged, aiming to facilitate the efficient and reliable interaction between blockchain and traditional network environments. This technology not only provides technical support for

expanding blockchain application scenarios, but also offers new solutions for data exchange and identity authentication [1]. The core objective of on-chain and off-chain trustworthy interaction is to ensure that data exchange between on-chain and off-chain environments maintains integrity and trustworthiness without compromising user privacy. Here, "on-chain" refers to the blockchain system, while "off-chain" refers to information systems outside the blockchain. Blockchain applications for on-chain and off-chain trustworthy interactions can fully leverage the advantages of blockchain, such as trustworthiness, traceability, and immutability, while taking advantage of the lower computational and storage costs and flexibility of off-chain information systems, thus achieving complementary advantages between the two [2]. However, blockchain technology can only guarantee the trustworthiness of the on-chain closed environment, and methods to ensure privacy security during interactions are a key technical challenge faced by on-chain and off-chain trustworthy interactions. The transparency feature of the blockchain, while ensuring data verifiability and resistance to tampering, can also lead to the leakage of sensitive information. For example, users' identity information, transaction amounts, and other data might be publicly available in on-chain environments, posing security risks for personal privacy and business secrets. Although the off-chain part can provide some degree of protection for data privacy, due to the security and trust mechanism issues still apparent in the connection between blockchain and traditional database systems, sensitive data processed by off-chain systems may also be exposed in insecure environments [3].

While Nakamoto's foundational work [4] established blockchain's transparency and immutability as pillars of trust, Zyskind et al. [5] highlighted that these very features risk exposing sensitive data—a concern echoed by 67% of enterprises in IDC's 2023 blockchain adoption survey [6]. This underscores the critical need for privacy-preserving mechanisms in cross-chain interactions.

Current hybrid architectures [7] typically employ blockchain for verifiable data anchoring while offloading computation to off-chain systems. However, these frameworks face two fundamental challenges as follows: Privacy vulnerabilities: The transparency of blockchain models (e.g., Bitcoin's UTXO [4]) combined with off-chain system vulnerabilities creates dual attack surfaces. For instance, medical diagnostic data in healthcare consortium chains can be deanonymized through metadata analysis [8]. Verification bottlenecks: Existing solutions like Hyperledger Fabric's channel isolation [7] introduce 2.3–4.1 $\times$ latency overhead for transaction validation [9], compromising real-time performance. Decentralized Identity (DID) systems, built on the W3C DID standard [10], offer promising solutions. The self-sovereign identity model pioneered by Sovrin [11] leverages zero-knowledge proofs (ZKPs) to enable minimal disclosure. Yet, as demonstrated by Camenisch et al. [12], current DID schemes suffer from >200 ms verification latency for 50+ attribute credentials, failing to meet real-time requirements. More alarmingly, Microsoft's analysis of Entra ID [13] revealed that 93% of traditional attribute disclosures expose redundant sensitive data. To address the above privacy issues related to on-chain and off-chain data, distributed identity management has become an important research direction. The aim of decentralized identity [10] (DID) is to give users ownership and control over their identity information, and to decentralize the storage and verification of data across a distributed network, allowing users to independently manage and control their identity information without relying on a single centralized identity verification agency. Distributed identity systems can effectively reduce reliance on centralized identity issuers and significantly reduce the risk of centralized data breaches.

To address this issue, selective disclosure technology [14] has emerged. Selective disclosure is a technique that uses encryption and verification mechanisms to allow users to disclose only specific identity information. Users can choose to provide only some identity

data to the verifier without revealing all information. This approach not only enhances the protection of user privacy but also increases the flexibility and adaptability of decentralized identity systems [15]. For example, when a user needs to prove that they are over 18 years old, they only need to disclose their age and do not need to reveal their birthdate or other personal details. Selective disclosure has several advantages in the implementation of decentralized identities. First, selective disclosure can effectively reduce the risk of data breaches and the potential for misuse of user privacy. Second, it implements the "minimal disclosure" principle in identity verification, which aligns with existing data protection laws (such as GDPR) requiring data minimization. Moreover, selective disclosure can also increase user trust in the decentralized identity system, further promoting its application across various domains. Therefore, selective disclosure is not only a key technology in decentralized identity systems but also an important method to safeguard user privacy and security.

This paper proposes a secure and efficient hash-based selective disclosure algorithm, MECQV, which introduces implicit certificates as a verifiable method for credential issuance and verification. Implicit certificates not only enhance the security of the hash-based selective disclosure method, ensuring unforgeability, but they also perform 80% faster than anonymous credential schemes. Additionally, this paper combines Merkle trees and zero-knowledge proof algorithms to hide the Merkle tree verification path, resisting man-in-the-middle attacks. Finally, this paper analyzes the security of the proposed scheme under the random oracle model, showing that the scheme can withstand known attacks and offer enhanced privacy protection.

To summarize, this article makes the following contributions.

1. We propose a secure and efficient hash-based selective disclosure algorithm, MECQV, which introduces implicit certificates as a verifiable method for credential issuance and verification.
2. We combine Merkle trees and zero-knowledge proof algorithms to hide the Merkle tree verification path, resisting man-in-the-middle attacks.
3. We evaluate MECQV both theoretically and experimentally. We compare the performance of MECQV with FlexDID. Our results show that all the operations of MECQV can be completed within 10 ms on a Linux Ubuntu 22.04 LTS machine. MECQV not only enhance the security of the hash-based selective disclosure method, ensuring unforgeability, but it also perform 80% faster than anonymous credential schemes.

The remainder of this article is organized as follows. We review related works in Section 2 and the preliminaries in Section 3. We provide our design in Section 4. We then show the workflow of MECQV in Section 5. Additionally, we provide a security proof and show our evaluation results in Section 6. Finally, we conclude our work in Section 7.

2. Related Work

Currently, the methods used to solve the selective disclosure problem are roughly divided into three categories as follows: atomic credentials, hash functions, and anonymous credentials [16].

Atomic credentials create different credentials for each user attribute. Although this approach provides more granular selection for users when constructing their credentials, the creation of atomic credentials is computationally and memory-intensive, as it requires as many signatures as there are attributes.

Another approach to implementing selective disclosure is through hash functions. The issuer creates a credential, but instead of directly inserting the attribute values into the credential, it combines the attribute values with a random number, computes a hash, and inserts the hash value into the credential. The credential is then signed, and both the

credential, attribute values, and associated random number are sent to the user via a secure channel. Therefore, when submitting the credential to the verifier, a subset of the attributes needs to be disclosed. The user must send these attribute values along with the relevant random numbers to prove the integrity and validity of the attribute. For instance, ref. [17] proposes a method that allows each element of an XML document to have a random number attribute, making it difficult for attackers to guess the original of the hash, and it uses a Merkle tree to prove the existence of multiple documents. However, this method increases the computational load. Ref. [18] proposes a selective disclosure method based on hash functions, which allows users to selectively disclose partial claims within a credential, with the issuer sending the hashed claims rather than the actual values. Users can disclose the values of the claims they want to disclose and the relevant random numbers to the verifier, who can verify the integrity and authenticity of the disclosed claims. This method provides fine-grained data-sharing control and, compared to atomic credentials or selective disclosure signatures, reduces computational overhead. A hash-based selective disclosure scheme for Self-Sovereign Identity (SSI) is proposed, which allows users to disclose only the necessary information during identity verification, reducing the risk of irrelevant data leakage. This scheme maintains high flexibility while implementing self-sovereign identity and is suitable for different application scenarios. A limitation to this approach is that practical deployment may face computational and storage overheads, and the security and scalability of multi-party verification still need further validation and optimization.

Currently, most solutions to the selective disclosure problem use anonymous credentials. In the 1980s, the concept of anonymous credentials was proposed by Chaum [19], and in 1999, the pseudonym system [20] was introduced. The first practical implementation was presented in 2001 by Jan Camenisch et al. [21], at the Eurocrypt conference. This scheme was based on the strong RSA assumption. It was the first practical solution that allowed users to prove ownership of a credential multiple times and in an unlinkable manner, without involving the issuing organization. A year after this proposal, Camenisch et al. designed a prototype based on this protocol, called the Idmix system. This scheme uses a large number of zero-knowledge proofs, and many later schemes have built on this approach. In 2008, Blanton et al. [22] proposed a scheme for anonymously obtaining online services. This scheme used the Camenisch–Lysyanskaya (CL) signature scheme [23], which relies on bilinear pairings. Although this scheme only discusses interactions between the user and the server, its anonymity features also make it applicable to constructing an anonymous credential system. Below is a detailed introduction to several research works closely related to this paper. Coconut [16] is an innovative selective disclosure credential scheme that supports distributed threshold issuance, public and private attributes, re-randomization, and multiple unlinkable selective attribute disclosures. It is integrated with blockchain and ensures confidentiality, authenticity, and availability even if some credential issuers are malicious or offline. Coconut is implemented as a general smart contract library for Chainspace and Ethereum, supporting applications such as anonymous payments, electronic petitions, and anti-censorship. BASS [24] proposes a blockchain-based anonymous authentication mechanism for privacy protection in smart industrial applications. BASS supports attribute privacy, selective revocation, credential integrity, and unlinkability in multiple presentations. It employs an efficient selective revocation mechanism based on dynamic accumulators and the Pointcheval–Sanders signature algorithm. BASS allows the selective revocation of credentials or users according to the issuer's needs. It also extends BASS from single-attribute privacy to multi-attribute privacy. Ref. [17] proposes a decentralized identity management system for industrial IoT (FLEXDID), which features high flexibility, scalability, and efficiency, supporting both vertical and horizontal credential disclosure methods. The proposed three-tier architecture can effectively handle the authen-

tication needs of a large number of industrial IIot devices. In terms of security, FLEXDID applies the PS signature scheme and zero-knowledge proofs to ensure credential unforgeability and unlinkability. However, the implementation of FLEXDID relies on multiple complex cryptographic techniques (such as PS signatures and RSA accumulators), which increases the difficulty of implementation and maintenance. Moreover, while FLEXDID runs efficiently on IIoT devices, its performance is still relatively low on low-performance devices, potentially due to hardware limitations.

Existing approaches to selective disclosure face several challenges. Atomic credentials, while providing fine-grained control, are computationally expensive and memory-intensive. Hash-based methods, although more efficient, are susceptible to man-in-the-middle attacks and often require significant computational resources for verification. Anonymous credential schemes, while offering high security and unlinkability, suffer from inefficiency due to their reliance on complex cryptographic operations such as bilinear pairings and zero-knowledge proofs. These limitations hinder their practical deployment in real-world applications, particularly in scenarios requiring high-frequency identity verification and low-latency responses.

To address these challenges, our proposed MECQV algorithm leverages implicit certificates and Merkle trees to achieve a balance between efficiency and security. By utilizing implicit certificates, we eliminate the need for complex bilinear pairings and reduce the computational overhead associated with credential generation and verification. The integration of Merkle trees allows for the efficient and secure verification of selective attributes, while the use of zero-knowledge proofs ensures that the verification path remains hidden, thereby enhancing privacy.

3. Preliminaries

3.1. Merkle Tree

A Merkle tree is an efficient data structure first proposed by Ronald Merkle in 1979 [25], and it is widely used in distributed systems and blockchain technology. It works by dividing data into multiple smaller blocks (i.e., leaf nodes) and using a hash function to combine these blocks in layers to form a tree structure, which eventually converges into a root hash. The core technical point of this structure lies in its ability to guarantee data integrity and security. The structure of a Merkle tree is shown in Figure 1, the letters A–H is the leaf nodes.

First, the construction of a Merkle tree takes advantage of the one-way nature and collision resistance of hash functions. The value of each node is the hash of its child node values, so any modification to a leaf node will cause the hash values of its parent nodes to change, ultimately affecting the root hash. This allows users to quickly verify the integrity of a specific data block using the root hash and the hash chain of the related data, without needing to access the entire dataset. This characteristic is known as “lazy verification”.

Second, Merkle trees support efficient data operations. In practical applications, when the validity of a particular data block needs to be verified, only the hash of that data block and the hashes of its sibling nodes need to be transmitted. By computing layer by layer, the final comparison is made with the root hash. This verification method significantly reduces data transmission and improves system efficiency.

Furthermore, Merkle trees also demonstrate advantages in data sharding and parallel processing. In distributed storage, data can be stored in fragments across different nodes, and the Merkle tree provides a mechanism for verifying the integrity of these fragments, ensuring that the data is not tampered with during transmission and storage.

Merkle trees are widely used, particularly in blockchain technology, where each block contains the hash of the previous block and the Merkle tree root hash of the current block’s

transaction data. This design ensures the order and immutability of transaction data. In the financial sector, Merkle trees are used to record and verify transaction histories, ensuring that all participants can trust the accuracy of the data. In supply chain management, it helps trace the origin and movement of products, enabling information transparency and traceability.

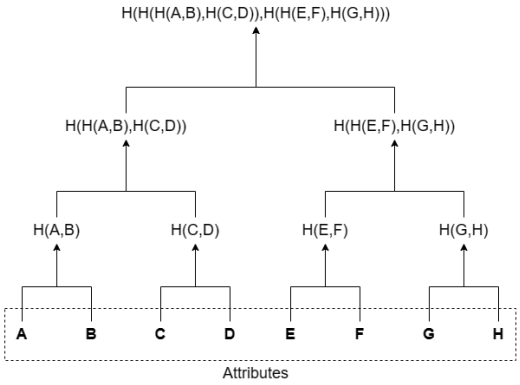

Figure 1. Merkle tree.

3.2. Implicit Certificate

The implicit certificate scheme consists of three entities as follows: the Certificate Authority (CA), the certificate requester (U), and the certificate verifier (V). The requester, U, obtains an implicit certificate from the CA to prove U's identity and allow V to obtain U's public key. The workflow of implicate certificate show in Figure 2.

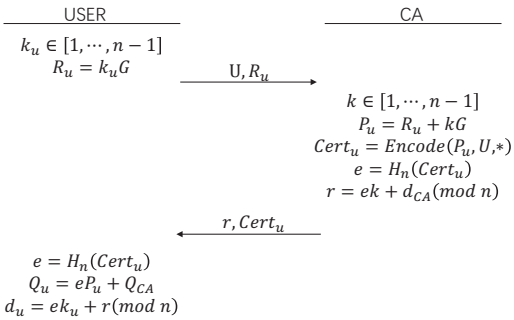

Figure 2. Workflow of implicate certificate.

First, U generates a random private key $k_U \in_R [1, \ldots, n-1]$, and calculates the corresponding public key $R_U = k_U G$. After generating the public-private key pair, U sends the public key and user information to the CA. The CA selects a random integer $k \in_R [1, \ldots, n-1]$ and computes the public key reconstruction value $P_U = R_U + kG$. After verifying U's identity, the CA generates the certificate Cert with the user information and P_U. Then, the CA computes the implicit signature value, which is the private key reconstruction value $r = ek + d_{\mathrm{CA}} (\mathrm{mod}\ n)$, where e is the certificate hash value. Subsequently, the CA returns the public–private key reconstruction value along with the certificate to U, who computes the final private key $d_U = ek_U + r (\mathrm{mod}\ n)$.

The implicit certificate scheme consists of the following five algorithms [26]:

1. ECQV Setup In this step, the CA establishes the elliptic curve domain parameters, hash functions, and certificate encoding formats, and each entity selects a random number generator. The CA generates key pairs, and each entity must keep a true copy of the CA's public key and domain parameters.

2. Cert Request The requester, U, must generate a certificate request, which is sent to the CA. The encryption component of this request is a public key, generated in the same way as the CA's public key during the ECQV Setup.
3. Cert Generate Upon receiving the certificate request from U, the CA first confirms U's identity and then creates an implicit certificate. The CA then sends a response to U.
4. Cert PK Extraction Given U's implicit certificate, domain parameters, the CA's public key, and a public key extraction algorithm, U's public key is calculated.
5. Cert Reception Upon receiving the certificate response, U must verify the validity of the implicit certificate key pair.

4. Selective Disclosure Algorithm Based on Implicit Certificates (MECQV)

This paper proposes a selective disclosure algorithm based on implicit certificates. The algorithm involves the following three roles: the user (U), who is the actual owner of the attributes; the certificate authority (A), which issues verifiable credentials to the user; and the verifier (V), which verifies the user's credentials. The user submits their attributes to the certificate authority (A). Upon receiving the attributes and confirming their authenticity, A generates a verifiable credential for U, following the algorithm, and stores the credential hash on the blockchain. When U needs to use the credential, they send the credential along with the signed information to the verifier (V). V verifies the credential and the corresponding signature to authenticate U's identity and provide the necessary service.

4.1. Algorithm Definition

$MECQV = (Setup, CertRequest, CertGenerate, CertExtraction, CertPresentation, CertVerification)$.

The algorithm components are described as follows:

- $Setup(1^\lambda) \rightarrow pp$: This algorithm is used to generate the system's public security parameters. λ is the system's security parameter, and the input is 1^λ. The output is the system's public parameters pp.
- $CertRequest(pp, \{m_i\}_{i=1}^n) \rightarrow (req)$: This algorithm is executed by the data owner (U). The input includes the public parameters pp and the set of attributes $\{m_i\}_{i=1}^n$. The output is a credential request req that U sends to the certificate authority (A). We present an algorithm as shown in Algorithm 1.
- $CertGenerate(\{m_i\}_{i=1}^n, P_{user}) \rightarrow (Pub_r, Prv_r, Cert)$: The credential generation involves the following two algorithms:
  - $Merkle(\{m_i\}_{i=1}^n) \rightarrow (root)$: This algorithm generates a Merkle tree and verifies the root node.
  - $Generate(root, P_{user}) \rightarrow (Pub_r, Prv_r, Cert)$: This algorithm takes the Merkle tree root node $root$ and the user's public key P_{user} as input to validate U's attributes. The output includes the public key reconstruction value Pub_r, the private key reconstruction value Prv_r, and the main credential $Cert$. We present an algorithm as shown in Algorithm 2.
- $CertExtraction(Pub_r, Prv_r) \rightarrow (Pub_u, Prv_u)$: This algorithm is executed by U, who takes the public key reconstruction value Pub_r and private key reconstruction value Prv_r as input. It outputs the credential's public key Pub_u and private key Prv_u. We present an algorithm as shown in Algorithm 3.
- $CertPresentation(S, root, \{P_i\}_{i\in S}, Prv_r) \rightarrow (Cert')$: This algorithm is executed by U. It takes the disclosed attribute set S, the Merkle tree root node $root$, and the proof path $\{P_i\}_{i\in S}$ from the leaf nodes to the root node as inputs. The output is the sub-credential $Cert'$. We present an algorithm as shown in Algorithm 4.

- $Proof(S, root, \{P_i\}_{i \in S}) \rightarrow (proof)$: This algorithm generates a zero-knowledge proof for the Merkle tree leaf node verification path.
- $Sign(proof, Prv_r) \rightarrow (Cert')$: This algorithm signs the proof generated by the previous algorithm.

- $CertVerification(Cert, Cert') \rightarrow (1/0)$: The verifier (V) executes this algorithm, which takes the main credential and the sub-credential as input. It outputs either 1 (valid) or 0 (invalid). We present an algorithm as shown in Algorithm 5.

Algorithm 1 Credential request

procedure CERTREQUEST($pp, \{m_i\}_{i=1}^{n}$)
  $K_{user} \xleftarrow{\$} \{0,1\}^n$
  $P_{user} \leftarrow K_{user} \cdot G$
  $req \leftarrow (K_{user}, \{m_i\}, P_{user})$
  return req

Algorithm 2 Credential generation

procedure CERTGENERATE($\{m_i\}, P_{user}$)
  Construct Merkle tree: $root \leftarrow H(m_1 \parallel \cdots \parallel m_n)$
  $K_A \xleftarrow{\$} \mathbb{Z}_n^*$
  $Pub_r \leftarrow K_A \cdot G + P_{user}$
  $Cert \leftarrow root \parallel Pub_r$
  $e \leftarrow H_1(Cert)$
  $Prv_r \leftarrow e \cdot K_A + d_A$ ▷ d_A: Authority's private key
  return $(Pub_r, Prv_r, Cert)$

Algorithm 3 Credential extraction

procedure CERTEXTRACTION(Pub_r, Prv_r)
  $e' \leftarrow H_1(Cert)$, $Pub_u \leftarrow e' \cdot Pub_r + Q_A$
  $Prv_u \leftarrow e' \cdot K_{user} + Prv_r$
  Verify $Pub_u \stackrel{?}{=} Prv_u \cdot G$
  return (Pub_u, Prv_u)

Algorithm 4 Credential presentation

procedure CERTPRESENTATION($S, root, \{P_i\}_{i \in S}$)
  Generate ZKP: $\pi \leftarrow$ ZK-SNARK($S, root, \{P_i\}$)
  $k \xleftarrow{\$} \mathbb{Z}_n^*$, $R \leftarrow k \cdot G$, $r \leftarrow x(R) \bmod n$
  $h \leftarrow H(\pi)$, $s \leftarrow k^{-1}(h + r \cdot k) \bmod n$
  return $Cert' \leftarrow (r, s) \parallel \pi$

Algorithm 5 Credential verification

procedure CERTVERIFICATION($Cert, Cert'$)
  Parse $\sigma = (r, s)$
  extract Pub_r
  Compute $w \leftarrow s^{-1} \bmod n$
  $u_1 \leftarrow h \cdot w$, $u_2 \leftarrow r \cdot w$
  $R' \leftarrow u_1 \cdot G + u_2 \cdot Pub_u$
  Verify $r \stackrel{?}{=} x(R') \bmod n$ and Merkle proof π
  return 1 if valid, 0 otherwise

4.2. Algorithm Description

This section provides a concrete example of the MECQV algorithm. Based on the flow of the credential, the process is divided into six stages as follows: initialization, credential request, credential generation, credential acceptance, credential presentation, and credential verification.

In the initialization stage, all entities involved in the credential process generate their DID identities, including their respective public and private key pairs. The certificate authority's public and private keys are represented as d_A and Q_A, the user's public and private keys as d_U and Q_U, and the verifier's public and private keys as d_V and Q_V. In addition to key generation, the hash functions and security parameters used in subsequent stages must also be defined.

1. Initialization stage
   $Setup(1^\lambda) \rightarrow pp$: Given the security parameter k, generate a k-bit prime number p and an elliptic curve E/F_p, where the curve is defined over the finite field F_p with a base point G and order n. Assume the existence of a collision-resistant hash function $H : \{0,1\}^* \rightarrow Z_p$.
2. Credential request stage
   $CertRequest(pp, \{m_i\}_{i=1}^n) \rightarrow (req)$: U randomly selects an n-bit number as the private key K_{user} and computes the public key $P_{user} = K_{user}G$. U generates the credential request $req = (K_{user}, \{m_i\}_{i=1}^n, P_{user})$ and sends req to certificate authority (A).
3. Credential generation stage
   $CertGenerate(\{m_i\}_{i=1}^n, root, P_{user}) \rightarrow (Pub_r, Prv_r, Cert)$: This stage consists of the following two algorithms:
   - $Merkle(\{m_i\}_{i=1}^n) \rightarrow (root)$: This algorithm takes the attribute set and treats them as the leaf nodes of a Merkle tree. It uses a hash function to generate the root node $root = H(m_1, \ldots, m_n)$.
   - $Generate(root, P_{user}) \rightarrow (Pub_r, Prv_r, Cert)$: A randomly selects a number K_A, computes $Pub_r = K_A G + P_{user}$, and generates the credential $Cert = root||Pub_r$. The certificate hash is computed as $e = H_1(Cert)$, and the private key reconstruction value is calculated as $Prv_r = eK_A + d_A$, where d_A is A's private key.
4. Credential acceptance stage
   $CertExtraction(Pub_r, Prv_r) \rightarrow (Pub_u, Prv_u)$: U computes $e' = H_1(Cert)$, the credential public key $Pub_u = e'Pub_r + Q_A$, and the credential private key $Prv_u = e'K_{user} + Prv_r$. Here, Q_A is the certificate authority's public key. The verification step is $Pub_u = Prv_u G$.
5. Credential presentation stage
   $CertPresentation(S, root, \{P_i\}_{i \in S}) \rightarrow (Cert')$: Credential presentation involves the following two algorithms:
   - $Proof(S, root, \{P_i\}_{i \in S}) \rightarrow (\pi)$: This algorithm generates a zero-knowledge proof for the leaf node verification path in the Merkle tree.
   - $Sign(\pi, Prv_r) \rightarrow (Cert')$: A random number k is chosen, and the point $R = (x_1, y_1) = k \cdot G$ is calculated. The signature parameter is $r = x_1 \mod n$. If $r = 0$, a new random number is chosen. The value $h = H(\pi)$ is computed, and the signature $s = k^{-1} \cdot (h + r \cdot k) \mod n$ is obtained. If $s = 0$, a new random number is selected. The signature $\sigma = (r, s)$ is generated, and the sub-credential $Cert' = \sigma||\pi$ is output.
6. Credential verification stage
   $CertVerification(Cert, Cert') \rightarrow (1/0)$: V first extracts the public key reconstruction value from the credential and computes $Pub_u = e'Pub_r + Q_A$. V verifies that the

signature $\sigma = (r, s)$ satisfies the conditions $1 \leq r \leq n-1$ and $1 \leq s \leq n-1$. The value $h = H(\pi)$ is computed, and $w = s^{-1} \mod n$. The values $u_1 = h \cdot w \mod n$,$u_2 = r \cdot w \mod n$ compute $R = (x, y) = u_1 \cdot G + u_2 \cdot Pub_u$. If the point $R = O$, the output is 0. Otherwise, check whether $v = x_1 \mod n$. Finally, verify the validity of the proof π.

4.3. Security Model

The legitimate user in the system is denoted as Bob, and the certificate authority is denoted as A, with its public key represented as Q_A. We assume that Bob possesses a true copy of A's public key. Bob's request to A for a certificate is represented as $(Pub_B, \{m_i\}_{i=1}^n)$. The response from A is denoted as $(Pub_r, Prv_r, root)$. Bob can issue multiple credential requests to A.

The adversary $\mathcal{A}$ is a probabilistic Turing machine with a runtime bounded by τ. It can interact with the user and the certificate authority through the following two operations:

- The adversary can observe Bob's credential request to the certificate authority, denoted as $(Pub_B, \{m_i\}_{i=1}^n)$.
- After observing Bob's request, the adversary can send $(Pub_B, \{m_i\}_{i=1}^n)$ to the certificate authority and receive the response $(Pub_r, Prv_r, root)$, where Prv_r is the private key reconstruction value generated by the certificate authority, and *root* is the Merkle tree root.

The probability that the adversary $\mathcal{A}$ outputs a tuple $(Pub_u, root, Prv_u)$ is at most ϵ, where Prv_u is the private key computed by U based on $(Pub_r, Prv_r, root)$, satisfying $Prv_u G = H(Pub_r || root) Pub_r + Q_A$. In this model, the adversary $\mathcal{A}$ is considered successful if it forges an implicit certificate under either of the following two scenarios:

1. $\mathcal{A}$ generates an implicit certificate and the corresponding private key that were not issued by the certificate authority. 2. $\mathcal{A}$ generates an implicit certificate and the corresponding private key, where the private key was issued by the certificate authority in response to a request from Bob.

If no such adversary $\mathcal{A}$ exists, the MECQV algorithm is considered secure.

5. System Model

The proposed system model consists of the following four main components: the verifier, the certificate authority, the user, and the blockchain network. The user is the entity that provides data and can obtain credentials based on their attributes. The certificate authority is responsible for issuing credentials to entities within the system and uploading credential digests to the blockchain to create verifiable records. The verifier uses the credentials to authenticate the user and provide the corresponding service.

The system process is divided into the following five stages: initialization, credential request, credential generation, credential presentation, and credential verification. The system architecture is shown in Figure 3.

This section uses a medical scenario as an example as follows: the patient acts as the user and has control over their attributes, while the hospital serves as the certificate authority and issues identity credentials to the patient. These credentials include the patient's identity information and medical data. The doctor acts as the verifier, who authenticates the patient's identity during a consultation. The patient can selectively disclose relevant information to the doctor as needed.

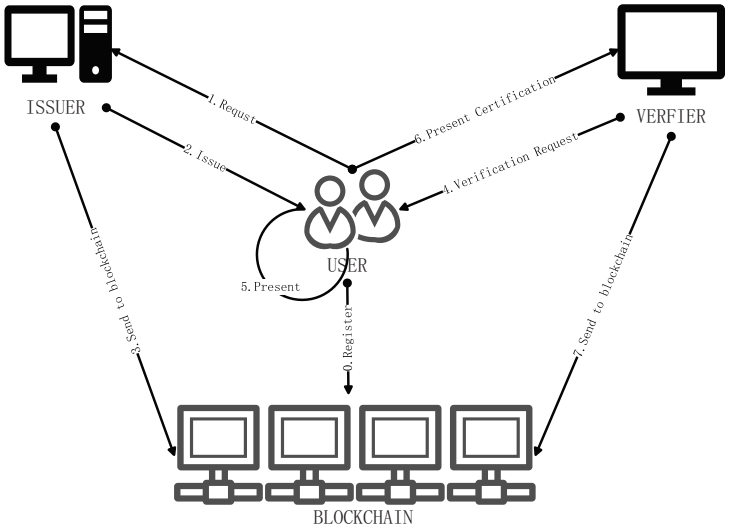

Figure 3. System architecture.

System Flow

1. Initialization phase
   In the key generation phase, the patient, hospital, and doctor each register their own DID identities, and the identity registration process is the process of generating their respective keys. Taking the patient's DID identity registration as an example, the steps are as follows:
   (a) The patient randomly selects $sk \in [1, n-1]$ and calculates $pk = skG$, where sk is the patient's private key and pk is the patient's public key.
   (b) The patient generates a DID identifier, $DID = hash(pk)$.
   (c) The patient generates a DID document, which includes the patient's identifier, public key, signature, and other information. The patient sends a registration request to the blockchain node, $request_{reg} = (DID, DIDDocument, Sign_{User})$.
   (d) After receiving the request, the blockchain node verifies the validity of the DID identifier and document, and it then packages them into a block for on-chain storage.
2. Credential request phase
   In the credential request phase, the patient sends their attributes to the hospital and applies for a credential related to those attributes.
   (a) The patient generates a temporary key pair (P_{user}, K_{user}) according to the steps in the above algorithm.
   (b) The patient uses the DID identifier registered on the blockchain to send a credential request to the hospital as follows: $request = (DID, \{m_i\}_{i=1}^{n}, P_{user}, Sign_{pk})$.
3. Credential generation phase
   In the credential generation phase, the hospital issues a credential for the patient based on the received attributes. The hospital must first determine whether the patient's identity attributes are valid. Then, the hospital uses the *CertGenerate* algorithm to generate the patient's credential, attaching the DID identifier to anchor the relationship between the patient's identity and the credential. After generating the credential, the hospital uploads it to the blockchain network.
   (a) Upon receiving the request, the hospital first queries the patient's DID identity from the blockchain network and retrieves the patient's public key information

from the DID document. Then, the hospital verifies whether the attributes submitted by the patient are valid.

(b) The issuing institution generates the public and private key reconstruction values Pub_{re}, Pri_{re} and the credential $Cert_{user}$ based on the credential generation algorithm from step three.

(c) The issuing institution returns the credential and the private key reconstruction value to the patient.

(d) After receiving the credential and private key reconstruction value, the patient calculates the credential public and private keys using the $CertExtraction$ algorithm in the credential acceptance phase.

4. Credential presentation phase
In the credential presentation phase, the doctor initiates an identity verification request. After receiving the request, the patient runs the $Proof$ and $Sign$ algorithms to generate proof of the attributes. The patient then sends the proof and the credential to the doctor.

(a) The doctor initiates an identity verification request to the patient.

(b) The patient selects the attributes (e.g., age) to disclose and generates a zero-knowledge proof π for that attribute. The purpose of this proof is to hide the Merkle tree verification path.

(c) The patient signs the zero-knowledge proof of the verification path using the credential's private key, $Sign(\pi, Prv_r)$, and sends the signature and credential to the doctor.

5. Credential verification phase
Upon receiving the proof and credential from the patient, the doctor first queries the blockchain to verify the patient's DID identity. Then, the doctor calculates the credential public key from the parameters in the credential to verify the signature in the proof. After the signature is verified, the doctor verifies the zero-knowledge proof by comparing the root node obtained from the zero-knowledge proof calculation with the root node contained in the credential to validate the attribute disclosed by the patient.

(a) The doctor verifies the credential $CertVerification(Cert, Cert') \rightarrow (1/0)$ by first checking the correctness of the signature, then verifying the validity of the credential, and finally verifying the validity of the proof π.

(b) In verifying the signature, the doctor can both verify whether the patient owns the private key corresponding to the credential and whether the credential was issued by the hospital. Ultimately, the doctor verifies whether the root node of the Merkle tree in the credential is valid based on the zero-knowledge proof.

6. Performance and Security Analysis

6.1. Performance Analysis

The experiment was conducted under the following hardware conditions: Processor: Intel(R) Core(TM) i7-13650HX, clock speed 2.2 GHz, 8 GB RAM, and 64-bit Linux operating system. In this experiment, the proposed selective disclosure algorithm, including the phases of initialization, credential request, credential generation, credential acceptance, credential presentation, and credential verification, was implemented in Python 3.11. To handle the encryption and hash operations in the algorithm, we used the fastecdsa GitHub repository (https://github.com/AntonKueltz/fastecdsa, accessed on 12 June 2014) for implicit certificate implementation. These settings provide a unified and reproducible

foundation for performance testing. A comparison of the key metrics is presented in Table 1.

Table 1. Key metrics comparison table.

Phase	Metrics	MECQV	FlexDID
Credential Generation	Execution time	2.1 ms ($n = 32$)	12.5 ms ($n = 32$)
	Computational complexity	$O(n)$	$O(n^2)$
	Resource consumption	58 KB	210 KB
Credential Presentation	Execution time	8.3 ms	22.7 ms
	Computational complexity	$O(n)$	$O(n^2)$
	Resource consumption	320 Bytes	850 Bytes
Credential Verification	Execution time	1.7 ms	9.8 ms
	Computational complexity	$O(n)$	$O(n)$

In this experiment, the performance evaluation focused on the following four key phases of the algorithm: credential request, credential generation, credential presentation, and credential verification. The running time for each phase was recorded and used to evaluate the computational overhead of the algorithm. Additionally, to comprehensively analyze the algorithm's performance in different application scenarios, we tested the impact of varying attribute set sizes (e.g., 8, 16, 32, 64, etc.) on the time consumption of each phase. Considering the real-world application scenarios of this selective disclosure scheme (e.g., finance, healthcare, and industrial internet), the efficiency of the credential generation and verification phases became key indicators for assessing the performance of the scheme.

To objectively evaluate the performance of the algorithm, this experiment compares the proposed MECQV algorithm with the existing FlexDID scheme. The FlexDID scheme employs an editable PS signature mechanism with the following three major phases: the generation phase, aggregation phase, and verification phase. In contrast, MECQV simplifies the credential presentation process using implicit certificates and Merkle trees, avoiding complex exponential and bilinear pairing computations, giving MECQV a computational advantage in terms of overall complexity. As shown in Figure 4, the experimental data for each phase indicate that MECQV performs significantly better in terms of average time consumption during the generation and verification phases compared to FlexDID. The complexity of the FlexDID signature scheme increases the overall runtime, whereas MECQV demonstrates a notable lightweight characteristic in terms of performance.

Based on the experimental results, further analysis was conducted on the impact of varying attribute quantities on time consumption for each phase of MECQV, as shown in Figure 5. The data suggests that the number of attributes has little effect on the credential generation and verification phases, as MECQV's design ensures that the attribute count primarily impacts the construction of the Merkle tree, which is highly efficient and does not cause significant time increases. However, the credential presentation phase, due to the computational overhead of the zero-knowledge proof algorithm, takes relatively longer but remains within an acceptable range.

The following is a detailed analysis of the experimental data for each phase:

- Credential request phase: In the request phase, the user does not need to perform complex calculations, and the number of attributes has very little impact on this phase.
- Credential generation phase: Due to the use of implicit certificate generation, the computational complexity is low, and the number of attributes has a minimal impact on this phase.
- Credential presentation phase: This phase is the most time-consuming due to the complexity of the zero-knowledge proof computation.

- Credential verification phase: The verification phase has a relatively short time consumption, and there is no significant increase in time as the number of attributes increases.

In summary, the experimental results show that the MECQV scheme offers significant advantages in efficiency, particularly in the credential generation and verification phases. The use of implicit certificates and Merkle trees in MECQV maintains a low system overhead in terms of computational complexity. Although the zero-knowledge proof algorithm introduces some computational cost in the credential presentation phase, the overall lightweight nature of the scheme still allows it to meet the performance requirements for high-frequency identity verification in practical applications.

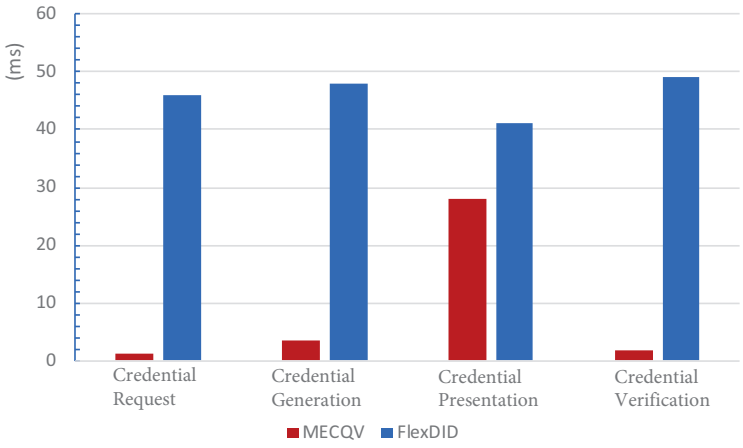

Figure 4. Efficiency comparison.

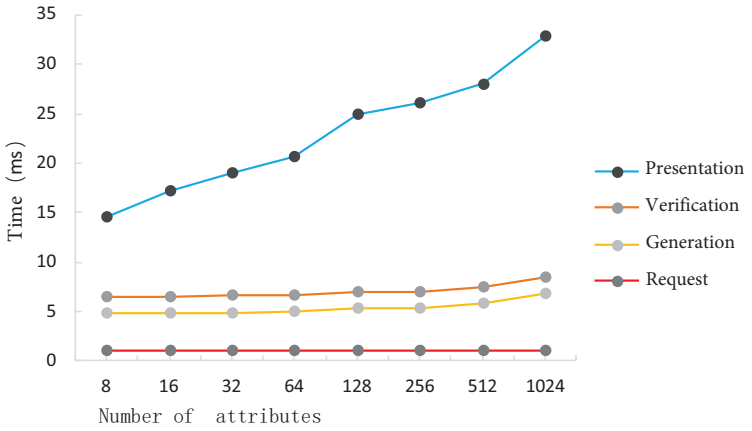

Figure 5. Efficiency data.

6.2. Security Analysis

In the current scenario, the attacker may intercept parameters sent to the issuer, such as the root of the attribute tree and public key. However, since the attacker cannot access the private key used by the user with the credential, they cannot use the credential. This mechanism ensures the security and confidentiality of the private key, effectively preventing the credential from being copied and reused by attackers to steal user information. Furthermore, during the use of the credential, even if the attacker intercepts the credential, the zero-knowledge proof ensures that the attacker cannot retrieve the credential's information. Ultimately, the attacker cannot recover the entire Merkle tree or obtain the root node information.

Theorem 1. *In the random oracle model, if the discrete logarithm problem on group G is hard and the Schnorr signatures are secure, then, the MECQV algorithm is secure.*

Proof. Assume that the MECQV scheme is insecure when the hash function H is treated as a random oracle. There exists a successful (τ, ϵ)-adversary $\mathcal{A}$. We construct a polynomial-time algorithm $S - (\tau, \epsilon)$ that can invoke $\mathcal{A}$ as a subroutine to solve the Elliptic Curve Discrete Logarithm Problem (ECDLP).

The input to S is a challenge value $((C \in G), C \neq O)$, and S expects an output of an integer $c \in [1, n-1]$ such that $C = cG$. First, S takes the input (C, m, H), and there are two possible expected outputs:

- (i) An integer $c \in [1, n-1]$, such that $C = cG$;
- (ii) A pair (P, b), where $bG = H(P, m)P + C$, meaning that (P, b) is the signature for the message m.

If case (i) occurs, S outputs c and terminates. If case (ii) occurs, Theorem 1 is used to simplify the signature forgery problem for S into solving the discrete logarithm problem and obtaining c. If this stage is successful, S outputs c and terminates.

To calculate c, S runs the subroutine $\mathcal{A}$. The algorithm $\mathcal{A}$ expects at least one issuer, each with a public key, but $\mathcal{A}$ does not have the corresponding private key. S randomly selects a public key C from one of the issuers as the challenge value and uses C as the input to S.

Since $\mathcal{A}$ can request credentials from the issuer and expect a valid response, S must provide a response that is at least valid from the perspective of $\mathcal{A}$. S can use the random oracle H to simulate the role of the issuer without being detected by $\mathcal{A}$. S follows this process to simulate the following: given a credential request $(Pub_B, \{m_i\}_{i=1}^n)$, S generates an integer $Prv_r, h_i \in R[0, n-1]$, and computes $Pub_r = Pub_B + h_i^{-1}(Prv_r G - C)$. S defines $H(Pub_r, \{m_i\}_{i=1}^n) = h_i$, and returns the triple $(Pub_r, root, Prv_r)$ as the response to $\mathcal{A}$'s request. Since $H(P, \{m_i\}_{i=1}^n)Pub_B + Prv_r G = H(P, \{m_i\}_{i=1}^n)P + C$, the response to the credential request is valid from $\mathcal{A}$'s perspective. Furthermore, from $\mathcal{A}$'s point of view, the hash function is random because h_i was initially chosen randomly.

$\mathcal{A}$ may also send hash queries to S. For example, if $\mathcal{A}$ sends a query $(Pub_{\mathcal{A}}, I_{\mathcal{A}})$, where $I_{\mathcal{A}} = \{m_i\}_{i=1}^n$ is a pair that has not been queried before, S outputs $H(Pub_{\mathcal{A}}, m)$, where m is the message that S attempts to forge a signature for. Clearly, the distribution of the simulated hash values generated by S is indistinguishable from the distribution of hash values generated by a random oracle from $\mathcal{A}$'s perspective.

Assume that $\mathcal{A}$ is successful. Then, $\mathcal{A}$ returns a triple $(Pub_u, root, Prv_u)$, where $Prv_u G = hPub_r + Q_A$, and $h = H(Pub_r, \{m_i\}_{i=1}^n)$, such that either of the following holds:

- (i) $(Pub_r, \{m_i\}_{i=1}^n)$ is a certificate created by $\mathcal{A}$ based on Bob's request; or
- (ii) $(Pub_r, \{m_i\}_{i=1}^n)$ is not a certificate issued by the issuer.

Assume we are in the first case. The private key Prv_u output by $\mathcal{A}$ satisfies $Prv_u = Prv_B h + Prv_r \mod n$, so $c = Prv_B$. Thus, S can compute $c = h^{-1}(Prv_u - Prv_r) \mod n$.

Assume we are in the second case. We can assume that $(Pub_r, \{m_i\}_{i=1}^n)$ is an input query to the random oracle hash H. Therefore, by the definition of the simulation, $H(Pub_r, \{m_i\}_{i=1}^n) = H(Pub_r, m)$. Now, (Pub_r, Prv_u) is a signature for message m, and S has completed the forgery of the signature for message m.

Clearly, if $\mathcal{A}$ runs in polynomial time and succeeds with non-negligible probability, then, S will also succeed. According to Theorem 1 and the above reasoning, if $\mathcal{A}$ runs in polynomial time and succeeds with non-negligible probability, then, S will also succeed. However, according to this assumption, there is no algorithm S that can solve the discrete logarithm problem in G. Therefore, unless the discrete logarithm problem in G can be solved efficiently, no adversary $\mathcal{A}$ exists in the random oracle model. □

7. Conclusions

This paper presents the MECQV algorithm, which synergizes implicit certificates with Merkle trees to achieve efficient and secure attribute-selective disclosure in distributed identity management. By embedding Merkle tree roots into verifiable credentials and leveraging zero-knowledge proofs (ZK-SNARKs) to conceal verification paths, the scheme enhances privacy protection while reducing computational complexity. Experimental results demonstrate that MECQV outperforms existing anonymous credential systems in efficiency, completing all operations within 10 milliseconds and achieving an 80% performance improvement, making it suitable for high-concurrency scenarios such as IoT and finance. However, the scheme exhibits certain limitations. Its reliance on elliptic curve cryptography (ECC) and Schnorr signatures introduces vulnerability to quantum computing attacks (e.g., Shor's algorithm), potentially compromising long-term security. While the overall efficiency is notable, the computational overhead from zero-knowledge proofs during credential presentation may hinder applicability on low-performance devices. Furthermore, practical deployment could face challenges in storage and communication costs due to the distributed management of implicit certificates and Merkle tree maintenance across multiple nodes.

Future research directions should focus on migrating to post-quantum cryptographic algorithms, such as lattice-based Dilithium or hash-based SPHINCS+, while upgrading Merkle tree hash functions to quantum-resistant variants like SHA-3. A hybrid encryption architecture combining traditional algorithms (e.g., ECDH) with post-quantum counterparts (e.g., Kyber) could ensure transitional compatibility and forward security. Optimizing zero-knowledge proof systems through lightweight constructions or Learning With Errors (LWE)-based approaches may reduce computational demands and align with the post-quantum requirements. Additionally, exploring cross-chain interoperability and standardization protocols for distributed identity systems could enhance scalability and multi-scenario integration. These advancements would enable MECQV to serve as a robust, privacy-centric solution in quantum-vulnerable environments, extending its applicability to emerging domains such as digital healthcare and decentralized IoT ecosystems.

Author Contributions: Conceptualization, G.Z. and G.W.; methodology, G.W.; software, G.W.; validation, G.W.; formal analysis, G.W. and G.Z.; investigation, G.W.; resources, G.W.; data curation, G.W.; writing—original draft preparation, G.W.; writing—review and editing, G.Z.; visualization, G.W.; supervision, G.Z.; project administration, G.Z.; funding acquisition, G.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China (2022YFB2702800).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

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

References

1. Pointcheval, D.; Sanders, O. Short randomizable signatures. In Proceedings of the Topics in Cryptology-CT-RSA 2016: The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, 29 February–4 March 2016; Proceedings; Springer: Berlin/Heidelberg, Germany, 2016; pp. 111–126.
2. Boneh, D.; Gentry, C.; Lynn, B.; Shacham, H. Aggregate and verifiably encrypted signatures from bilinear maps. In Proceedings of the Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, 4–8 May 2003; Proceedings 22; Springer: Berlin/Heidelberg, Germany, 2003; pp. 416–432.

3. Maram, D.; Malvai, H.; Zhang, F.; Jean-Louis, N.; Frolov, A.; Kell, T.; Lobban, T.; Moy, C.; Juels, A.; Miller, A. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 24–27 May 2021; pp. 1348–1366.
4. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 20 August 2023).
5. Zyskind, G.; Nathan, O.; Pentland, A. Decentralizing Privacy: Using Blockchain to Protect Personal Data. In Proceedings of the Proc. IEEE Symposium on Security and Privacy (S&P), San Jose, CA, USA, 21–22 May 2015; pp. 461–475.
6. International Data Corporation (IDC). *Blockchain Adoption Trends Report*; Technical Report IDC_TR-2023-BC01; IDC: Needham, MA, USA, 2023.
7. Hyperledger Foundation. *Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains*; Version 2.4 Documentation; Hyperledger Foundation: San Francisco, CA, USA, 2022.
8. Tobin, A.; Reed, D. The inevitable rise of self-sovereign identity. *Sovrin Found.* **2016**, *29*, 18.
9. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; Caro, A.D.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger Fabric Performance Characterization and Optimization. In Proceedings of the IEEE International Conference on Blockchain (ICBC), Toronto, ON, Canada, 2–6 May 2020; pp. 1–10.
10. W3C. *Decentralized Identifiers (DIDs) v1.0*; W3C Recommendation; W3C: Cambridge, MA, USA, 2022.
11. Sovrin Foundation. *Sovrin: A Protocol and Token for Self-Sovereign Identity*; White Paper; Sovrin Foundation: Provo, UT, USA, 2021.
12. Camenisch, J.; Dubovitskaya, M.; Lehmann, A.; Neven, G.; Paquin, C.; Preiss, F.S. An Architecture for Privacy-Enhancing Credential Systems. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), Virtual, 9–13 November 2020; pp. 477–494.
13. Microsoft Security Response Center. *Entra ID Vulnerability Analysis Report*; Technical Report; Microsoft: Redmond, WA, USA, 2023.
14. Soltani, R.; Nguyen, U.T.; An, A.; Galdi, C. A Survey of Self-Sovereign Identity Ecosystem. *Secur. Commun. Netw.* **2021**, *2021*, 8873429. [CrossRef]
15. Waters, B. Efficient identity-based encryption without random oracles. In Proceedings of the Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; Proceedings 24; Springer: Berlin/Heidelberg, Germany, 2005; pp. 114–127.
16. Sonnino, A.; Al-Bassam, M.; Bano, S.; Meiklejohn, S.; Danezis, G. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. *arXiv* **2018**, arXiv:1802.07344.
17. Bian, Y.; Wang, X.; Jin, J.; Jiao, Z.; Duan, S. Flexible and Scalable Decentralized Identity Management for Industrial Internet of Things. *IEEE Internet Things J.* **2024**, *11*, 27058–27072. [CrossRef]
18. Saito, K.; Watanabe, S. Lightweight selective disclosure for verifiable documents on blockchain. *ICT Express* **2021**, *7*, 290–294. [CrossRef]
19. Chaum, D. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM* **1985**, *28*, 1030–1044. [CrossRef]
20. Lysyanskaya, A.; Rivest, R.L.; Sahai, A.; Wolf, S. Pseudonym systems. In Proceedings of the Selected Areas in Cryptography: 6th Annual International Workshop, SAC'99 Kingston, Kingston, ON, Canada, 9–10 August 1999; Proceedings 6; Springer: Berlin/Heidelberg, Germany, 2000; pp. 184–199.
21. Camenisch, J.; Lysyanskaya, A. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Proceedings of the Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Innsbruck, Austria, 6–10 May 2001; Proceedings 20; Springer: Berlin/Heidelberg, Germany, 2001; pp. 93–118.
22. Blanton, M. Online subscriptions with anonymous access. In Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, Tokyo, Japan, 18–20 March 2008; pp. 217–227.
23. Camenisch, J.; Lysyanskaya, A. Signature schemes and anonymous credentials from bilinear maps. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 56–72.
24. De Salve, A.; Lisi, A.; Mori, P.; Ricci, L. Selective disclosure in self-sovereign identity based on hashed values. In Proceedings of the 2022 IEEE Symposium on Computers and Communications (ISCC), Rhodes, Greece, 30 June–3 July 2022; pp. 1–8.
25. Merkle, R.C. *Secrecy, Authentication, and Public Key Systems*; Stanford University: Stanford, CA, USA, 1979.
26. Campagna, M. *SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV)*; Standards for Efficient Cryptography, Version 1.0; SEC: Redwood City, CA, USA, 2013.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.