applied sciences

MDPI

Article

A Universal Privacy-Preserving Multi-Blockchain Aggregated Identity Scheme

Nigang Sun [1], Yuanyi Zhang [2,*] and Yining Liu [3]

[1] School of Microelectronics and Control Engineering, Changzhou University, Changzhou 213164, China
[2] School of Computer Science and Artificial Intelligence, Changzhou University, Changzhou 213164, China
[3] School of Computer and Information Security, Guilin University of Electronic Technology, Guilin 541004, China
* Correspondence: revanton@icloud.com

Abstract: Cryptocurrencies offer various benefits in terms of privacy protection and cross-border transactions, but they have also been used for illicit activities such as money laundering due to their anonymous nature and the difficulty of cross-border regulation. Additionally, the unethical actions of some virtual asset service providers (VASPs), such as rug pulls and the embezzlement of user funds, have further eroded the trust between users and VASPs. Implementing identity management on blockchains can help restore trust between users and VASPs. However, current solutions have privacy concerns as identity providers have access to the asset balances and transaction records of each user's wallet account, and no solution can support all public blockchains unconditionally. To address these issues, this paper proposes a multi-chain aggregated identity scheme. In this scheme, the identity provider will issue a non-fungible token (NFT) for users who have undergone verification, and wallet accounts from different blockchains will be added to a cryptographic accumulator. The accumulator value is then bound to the identity NFT through a smart contract by the user. This allows the user to prove to others that the identity of the wallet account owner has been verified. The use of accumulators also allows users to combine proof for multiple wallets into a single proof, which significantly improves the efficiency and provides a way for VASPs such as centralized exchanges to demonstrate Proof of Reserves (PoR) to users. Importantly, this scheme preserves privacy as neither the identity provider nor the VASPs can link the user's real identity with the wallet accounts. Only regulators can access the user's identity data held by the identity provider and the user's wallet account held by the VASP to link real identities with wallet accounts for the purpose of sanctions or criminal investigations. Additionally, the scheme supports all blockchains by allowing wallet accounts from any public blockchain to be added to the accumulator. Furthermore, the NFT implementation in the scheme helps prevent identity loss or theft, as it can only be transferred by the identity provider.

Keywords: blockchain; cryptocurrency; identity; privacy protection; regulation

Citation: Sun, N.; Zhang, Y.; Liu, Y. A Universal Privacy-Preserving Multi-Blockchain Aggregated Identity Scheme. *Appl. Sci.* **2023**, *13*, 3806. https://doi.org/10.3390/app13063806

Academic Editors: Konstantinos Demertzis, Hui Li and Shancang Li

Received: 6 January 2023
Revised: 4 March 2023
Accepted: 13 March 2023
Published: 16 March 2023

1. Introduction

1.1. Background and Related Solutions

While cryptocurrencies bring convenience, they are also used by criminals due to their anonymity [1]. Cryptocurrency-related crimes such as rug pull [2], extortion, money laundering, and terrorist financing, involve huge amounts of money and have a negative impact on the blockchain ecosystem. For example, the attack on the bnb chain on 7 October 2022 [3], resulted in a total loss of USD 570 million. The attackers used the cross-chain bridge Stargate Protocol to transfer about USD 100 million in assets, and these losses are irreversible. For the problems brought on by anonymity, deanonymization [4] is the straightforward solution. Deanonymization solutions combine on-chain and off-chain data to analyze on-chain transactions and infer their identity. However, due to the difficulty of

obtaining useful off-chain data and the existence of some anonymous cross-chain exchange services, it is not easy to find the real world identity associated with a wallet account with these solutions. In addition, deanonymization can only investigate crimes after they have occurred but cannot prevent crimes from happening. Compared to deanonymization schemes, identity management schemes can help regulators determine rather than speculate on the identities of accounts on the blockchain. In addition, identity management solutions can effectively reduce the occurrence of related crimes by creating a circle of trust between verified virtual asset service providers (VASPs) and verified users. So, identity management is a better way to deal with cryptocurrency-based crimes.

Ateniese et al. [5] pointed out that the absence of certification creates obstacles to its wider acceptance in e-commerce and official uses. They proposed a bitcoin authentication system, in which a central trusted authority controls the user's bitcoin address. In this way, the cryptocurrency asset balances and transaction records of these users are completely exposed to the regulator, which threatens privacy. In the scheme proposed by Li, Peili, et al. [6], the generation of the user's certificate is independent of the public keys of the wallet accounts, which avoids the need for registration when the user wants to use a new account. However, the corresponding certificate should be attached to the transaction, which requires an additional cost. Biryukov et al. [7] proposed KYCE on Ethereum. In this scheme, the identity provider adds the dependency data of the user who passed the Know your customer (KYC) check to a cryptographic accumulator, and provides the user with a witness. However, each transaction requires the witness described above to be put into the transaction data, which also requires an additional cost. Decentralized identity (DID) [8], Self-Sovereign Identity (SSI) [9], and Soulbound tokens [10] can also serve the purpose of identity tracking. DID schemes [11–13] identify identities by using each wallet account as a DID and provide verifiable credentials of user attributes. Vitalik et al. proposed SBT to build the identity system of Web3. SBT schemes [14–16] bind identity credentials to specific wallet accounts by issuing a non-transferable Non-Fungible Token (NFT). However, these schemes can only be built on blockchains that support smart contracts. As mentioned above, there is currently no identity management solution that unconditionally supports all public blockchains. If identity management can be implemented on all blockchains, it will be more difficult for criminals to transfer assets. Besides, these schemes present a privacy threat because the identity provider knows each user's asset balance and transaction history, which discourages users from actively registering their identities with regulators.

1.2. Proposed Solution

In our previous work [17], we proposed a public blockchain-enabled identity scheme based on Merkle tree and smart contract. However, when applying the solution to non-custodial wallets, we found batch identity proofs to be a necessary requirement, and the Merkle tree was inefficient for batch membership proofs. Besides, due to the large number of VASPs, it is difficult to ask all VASPs to cooperate with the investigation, and requiring VASPs to verify the users' attributes increases their cost. To address these issues, we propose a new multi-chain aggregated identity system. The scheme replaces Merkle trees with RSA accumulators to implement constant-size batch membership proofs. In addition, the solution sets up a third-party verifier to verify the user's identity witness, and provides an interface for querying the white list of verified wallet accounts, so that a large number of small VASPs do not need to repeatedly verify the user's identity witness. This not only saves costs but also facilitates the criminal investigation of the regulatory authorities. Finally, the scheme abandons the attribute proof in the previous scheme, instead, the allowed business scopes, such as participation in initial coin offering (ICO), are recorded in the identity NFT. Therefore, VASPs can decide which services are allowed to be provided to the user based on the identity NFT without checking the user's attributes. The costs of VASPs are saved and the privacy of the users is better protected.

We will here provide a brief description of the new scheme. First, the identity provider checks the user's identity and mints an identity NFT for the user. Then, the user aggregates

wallet accounts of multiple blockchains into a cryptographic accumulator and associates the accumulator value with the identity NFT through the smart contract. Then the user can prove that these wallet accounts correspond to a verified identity. Therefore, the scheme supports the identity management of all public blockchains. The solution only needs to be deployed on one EVM-compatible blockchain instead of all blockchains, and the identity credentials are only on this blockchain. So, for other blockchains, the verification of identity is off-chain. Since only the accumulator value is stored on the blockchain, the identity provider only knows the user's real identity but not the user's wallet account, while VASPs only know the user's wallet account but not the user's identity. So this scheme protects the user's privacy. The identity provider and VASP know which identity NFT corresponds to the user identity and wallet account, respectively. Thus, regulators can trace the identity of suspicious accounts by correlating user identities held by identity providers with wallet accounts held by VASPs. Therefore, this scheme realizes the identity tracking of addresses on the chain without affecting the anonymity of the blockchain, which provides a new solution for the government to supervise the cryptocurrency.

2. Preliminaries

2.1. Blockchain

At the end of 2008, Nakamoto first proposed the blockchain in the article "Bitcoin: A Peer-to-Peer Electronic Cash System" [18]. A blockchain is made up of sequentially linked blocks that contain many transactions. In 2013, Ethereum [19] was proposed by Vitalik Buterin. Ethereum allows people to deploy immutable decentralized applications onto it. Decentralized Finance applications based on smart contracts offer a wide range of financial services without the need for typical financial intermediaries.

2.2. Smart Contracts

Smart contracts [20] were first proposed in 1995 by Nick Szabo [21]. Due to the lack of a trusted execution environment, there was no significant development until the emergence of Ethereum in 2014. A smart contract is an automatically executed computer program or protocol that allows trusted transactions without a trusted third party. Smart contracts built on the blockchain are guaranteed to be traceable and irreversible, due to the blockchain consensus mechanism. Smart contracts have many applications, including decentralized exchanges, NFT marketplaces, and flash loans [22].

2.3. NFT

Digital tokens are different from coins such as Bitcoin and Dogecoin. They are digital assets represented on a blockchain by means of a smart contract, so they have greater utility than coins, which are used solely as stores of value and currency. Tokens can be divided into Fungible Tokens (FT) and Non-Fungible Tokens (NFT). On Ethereum, ERC20 is the basic standard for fungible tokens [23], which is a token that is interchangeable and can be split into nearly infinite pieces. Non-fungible tokens are digital assets that contain identifying information recorded in smart contracts. This information makes each NFT unique, and as such, they cannot be directly replaced by another token. They cannot be swapped like for like, as no two NFTs are alike. Banknotes, in contrast, can be simply exchanged one for another; if they hold the same value, there is no difference to the holder between, say, one dollar bill and another.

The data of NFT is often stored on the Inter-Planetary File System (IPFS) [24] to ensure that its content is immutable. IPFS is a globally oriented, peer-to-peer distributed file system network. IPFS generates a hash value based on the content of the file as the file's address on the IPFS network, thus preventing file content tampering.

2.4. Cryptographic Accumulator

In cryptography, an accumulator is a one-way membership hash function. Cryptographic accumulators can prove that an element belongs to a certain set without ex-

posing other elements of the set. Some accumulators also support the construction of non-membership proofs.

In 1993, Josh Benaloh and Michael de Mare [25] proposed an accumulator scheme based on the RSA hypothesis construction, which is simple to construct and supports membership proofs. Camenisch, J. et al. proposed the application of dynamic accumulators and the effective revocation of anonymous credentials in 2002 [26]. Then, in 2009, they proposed an accumulator based on bilinear mapping and effective revocation of anonymous credentials [27].

Membership proofs can also be implemented with the Merkle tree [28]. Proving that a leaf node is part of a given Merkle tree requires computing several hashes proportional to the height of the tree. This method has been widely used for blockchain pruning and simplified payment verification (SPV) in blockchains. The advantage of accumulators is that their proof size is constant and does not increase with the number of members.

Ozdemir et al. have implemented an RSA accumulator inside the SNARK system [29], which they used to replace the Merkle tree. Their experiments show that the new system significantly reduces the cost required to commit the current state, compared to the existing approach of using Merkle trees.

3. System Design

3.1. System Architecture

This scheme comprises users, identity providers, VASPs, and regulators. Identity providers are government departments that manage the identities of users. VASP refers to crypto asset service providers such as centralized exchanges and NFT marketplaces. For some Dapps that support the whitelist function and some small VASPs, a third-party validator checks the user's identity witness and provides a whitelist for them, which can improve the efficiency of regulation and reduce the cost of these VASPs. Regulators include some government departments and criminal investigation agencies. The IDNFT contract is deployed on an EVM-compliant blockchain network. The architecture of the scheme is shown in Figure 1.

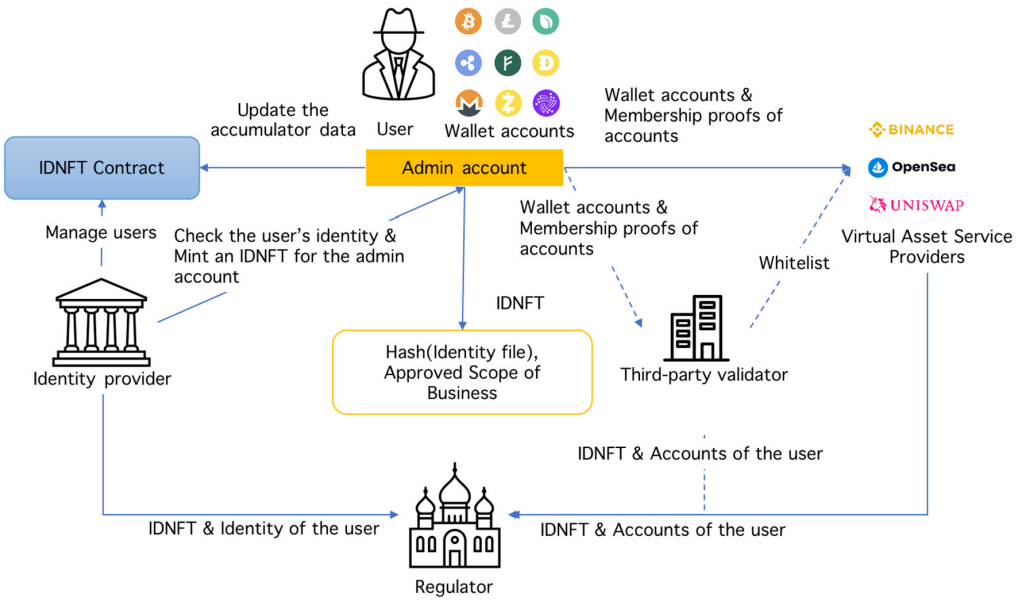

Figure 1. System architecture.

3.2. Program Flow

This section describes the details of the solution.

3.2.1. Identity Registration

First, the user interacts with the identity provider online or offline for identity checks and submits an Ethereum address as an admin account. The identity provider stores the necessary non-sensitive information (such as the hash digest of the user identity and the approved business scope) into IPFS, and the storage path is denoted as fileUrl. The identity provider calls the IDNFT contract to mint an identity NFT to the admin account, where the tokenUrl of the NFT is set to fileUrl.

3.2.2. Identity Authorization

The next step is to bind the wallet account to the identity using the RSA accumulator, which is done by the user. The scheme builds on Josh Benaloh and Michael de Mare's approach to constructing the RSA accumulator.

Before the user proves to the verifier (a VASP or a third-party validator) that several wallet accounts have been bound to a verified identity, the user generates two safe primes p and q ($(p-1)/2$ and $(q-1)/2$ should also be primes) and modulus $n = pq$ is taken. A prime number g is chosen as the generator. After that, the user generates a batch membership proof for a set of wallet accounts. First, the user initializes the accumulator as an empty accumulator. For each wallet account, the user specifies an authorized blockchain scope and maps the wallet account to a prime number. The user adds these prime numbers to the accumulator and obtains the new accumulator value, denoted as acc. Then the user generates a batch membership witness for all these prime numbers, denoted as wit. Finally, the user updates n, g, and acc to the corresponding UserData in the contract, and sets the parameter verified of the UserData to False. The verifier in UserData is set by the user as the verifier's account, so only the designated verifier can verify the user's membership proof.

3.2.3. Identity Verification

When a user proves to the verifier (a VASP or a third-party validator) that several wallet accounts addr have been bound to a verified identity, he needs to provide the membership witness wit of the wallet account, the authorized blockchain scope, and the id of the identity NFT NFTId. The VASP verifies wit and checks if the identity is expired or locked. If the verification is passed, it proves that the wallet account has been authorized by a verified identity. The verifier calls the contract to set the value of the verified property in UserData to True, which guarantees that the accumulator data will not be used for another verification, so malicious actors will not be able to use the forged member witness. Then, the user can access the service. The VASP records NFTId, addr, and wit to provide these data to regulators when required. The VASP sets the login validity period of the wallet account. After the user passes the verification, he can directly access the service within the validity period without repeated verification.

3.2.4. Identity Supervision

The regulator requests to obtain the user identity information held by the identity provider and the wallet account held by the VASP (or the third-party validator), so that the real world identity can be associated with the wallet account through NFTId.

3.3. Smart Contracts

The IDNFT contract is the core part of implementing the business logic of the identity management system. The contract is shown in Figure 2.

IDNFT.sol

```
mapping(uint256 => uint256) public NFTIdToExpirationTime;
mapping(address => UserData) private AdminToUserData;
mapping(address => UserData) private NFTIdToAdmin;
mapping(uint256 => bool) private NFTIdToAvailable;
struct UserData{uint256 NFTId; uint256 g; bytes acc; bytes n; address verifier; bool verified}
```

```
function mintIDNFT(address to, string tokenURI, uint256 expirationTime) public onlyOwner
function unlockIDNFT(uint256 NFTId, bool approveOrLock) public onlyOwner
function modifyAdminAddr(address oldAdmin, address newAdmin) public onlyOwner
function updateExpirationTime(uint256 NFTId, uint256 timestamp) public onlyOwner
function updateUserData(bytes acc, bytes n, uint256 g, address verifier) public
function setVerified(uint256 NFTId) public
function approveOwner() public
function isContract(address account) public view returns (bool)
function _beforeTokenTransfer(address from, address to, uint256 tokenId) internal override
```

Figure 2. IDNFT contract.

The four mappings are used to record the expiration time of the identity, record the corresponding relationship between the user's admin account and the user data, record the corresponding relationship between the user's admin account and the identity NFT, and control the validity of the identity. The struct userData includes the id of an identity NFT, an accumulator value, and the public key parameters of the accumulator.

Algorithm 1 is used to mint the NFT to store some immutable data. The NFT data is stored in IPFS with the storage path fileUrl, which will be used as the token URL of the NFT. The function will then set the identity NFT's expiration time and unlock the identity.

Algorithm 1: mintIDNFT () public onlyOwner

Input:
string tokenURL,
address to,
uint expirationTime
1: _mint(to, tokenURL)
2: updateExpirationTime(NFTId, expirationTime)
3: unlockIDNFT (NFTId, True)
4: AdminToUserData[to].NFTId = NFTId

Some of the main functions of the IDNFT contract are shown in Algorithms 2–6. The modifier onlyOwner is used to restrict the caller of the function to be the owner of the IDNFT Contract, which is the identity provider.

Algorithm 2 is used to control the validity of the user's identity. This function uses a mapping variable to record whether the current identity is valid. If the identity provider needs to freeze an identity due to reasons such as identity data expiration, they can set the mapping value corresponding to NFTId to False.

Algorithm 2: unlockIDNFT () public onlyOwner

Input:
uint NFTId,
bool approveOrLock
1: NFTIdToAvailable[NFTId] = approveOrLock

Algorithm 3 can only be called by the Owner to transfer the identity NFT to a new admin account. Before calling this function, the owner needs the user's approval to transfer the IDNFT. To avoid unexpected errors, the function performs a check to ensure that the new

admin account is not a contract address. The function then updates the old admin account in the relevant data to the new admin account and transfers the NFT to the new account.

Algorithm 3: modifyAdminAddr() public onlyOwner

Input:
address oldAdmin,
address newAdmin
1: require(isContract(newAdmin) == false)
2: UserData memory userdata = AdminToUserData[oldAdmin]
3: uint256 nftId = userdata.NFTId
4: require(nftId > 0)
5: AdminToUserData[newAdmin] = userdata
6: transferFrom(oldAdmin, newAdmin, nftId)

The function _beforeTokenTransfer() is overridden in the Algorithm 4 so that functions such as transfer() and mint() can only be called by the contract owner. Users cannot transfer their IDNFTs by themselves, which avoids the loss or theft of the IDNFT caused by the loss or theft of the private key of the admin account.

Algorithm 4: _beforeTokenTransfer() internal override

Input:
address from,
address to,
uint tokenId
1: super _beforeTokenTransfer(from, to, tokenId)
2: require(msg.sender == owner())

Algorithm 5 is used to update the accumulator value (acc) and the public key parameters (n, g), and to specify the verifier (verifier). When a user wants to prove identity authorization for certain wallet accounts, a new accumulator value will be calculated. The user then update this new value into the contract data through this function. When the user presents the proof of identity authorization to the verifier, the verifier will use the latest accumulator value stored in the smart contract for verification.

Algorithm 5: updateUserData() public

Input:
bytes acc,
bytes n,
uint g,
address verifier,
1: UserData storage userdata = AdminToUserData[msg.sender]
2: userdata.acc= acc
3: userdata.n = n
4: userdata.g = g
5: userdata.verifier = verifier
6: userdata.verified = False

Algorithm 6 is called by the verifier specified by the user in Algorithm 5, which is used to mark the accumulator value and public key parameters as verified to prevent them from being reused. The function first obtains userData according to the input parameter NFTId, and then checks whether the caller of the function is the verifier set in userData, and if so, sets the verified parameter to true, which means that this parameter has been used and cannot be used again.

Algorithm 6: setVerified() public

Input:
uint NFTId,
1: UserData storage userdata = AdminToUserData[NFTIdToAdmin[NFTId]]
2: require(userdata.verifier == msg.sender)
3: userdata.verified = True

3.4. Management of the Authorized Addresses

The identity authorization of the wallet account in the scheme is realized by using the RSA accumulator. First, the authorized blockchain scope is added to each wallet account as a suffix. They are then mapped to prime numbers and added to the accumulator value. The accumulator value is associated with the user's admin account through the IDNFT contract. The user proves the correspondence between the wallet account and the identity by showing the membership witness of the prime number corresponding to the wallet account in the accumulator. This section is divided into two parts: the first part explains how to map a wallet account to a prime number. The second part describes the construction of the RSA accumulator. The scheme uses Josh Benaloh and Michael de Mare's approach to construct RSA accumulators.

3.4.1. Map a Wallet Account to a Prime Number

To prevent falsification of membership proofs, the RSA accumulator requires that the elements are all prime. If an element is not prime (i.e., $m = pq$), if m is in the RSA accumulator, then it will be easy to construct a witness that p or q is in the accumulator. Therefore, we need to map the address information to the prime number first.

The same address can be used for different blockchains; for example, Bitcoin's P2PKH (Pay-to-PubKeyHash) address starts with the number 1, as shown below:

$$1A1zP1e \ldots mv7DivfNa$$

This address can be used to receive Bitcoin or Bitcoin Cash. To limit the scope of authorization, a suffix is added to the end of the wallet account to indicate that the blockchain is authorized. For example, the above address for identity authorization on the Bitcoin chain is expressed as:

$$1A1zP1e \ldots mv7DivfNa@BTC$$

Authorization of an address on both Ethereum and the bnb smart Chain is expressed as:

$$0x715cCB9B \ldots A89eEe@ETH@BSC$$

To support batch authentication of non-custodial wallets compatible with the BIP44 protocol [30], the extended public key can be added to the accumulator as an element. Using the extended public key can only derive the public key of the account, but not the private key, so the extended public key can be disclosed. VASPs can directly derive all wallet accounts corresponding to the user's mnemonic phrase from the extended public key without checking each account. Authorization of the extended public key is expressed as:

$$xpub6DDo \ldots 24hJ@BTC@ETH@BNB$$

Next, the wallet account information is mapped to a hash value, denoted as *Hex*, using the secure hash function Keccak-256. It takes the first l characters (including the two prefixes 0x) of the *Hex*, converts it to a decimal number and uses the Miller–Rabin primality test algorithm to determine whether it may be a prime number. If the result is not prime, the new hex is obtained by hashing the previous result as input using Keccak-256. Finally, we will get a prime number. The estimated average number of cycles of the method is $ln(16^{l-2} - 1)$. The above process is shown in Figure 3.

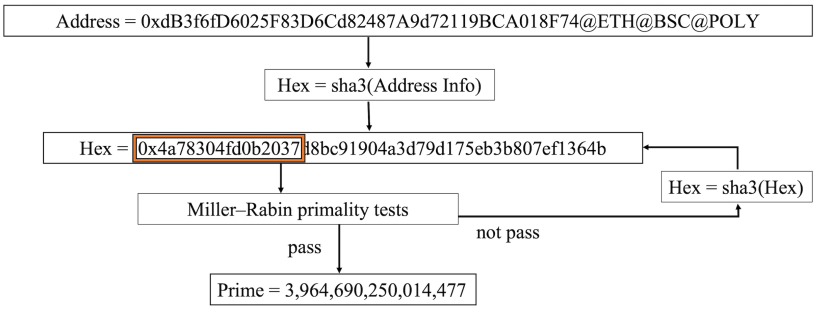

Figure 3. Mapping the address information to a prime number.

The Miller–Rabin algorithm detects whether the target is likely to be a prime number by random sampling multiple times. The scheme uses the Miller–Rabin deterministic primality test, where the selection of a specific integer can determine (rather than guess based on probability) whether an integer is prime within a certain range. For l = 18, the average runtime for mapping wallet accounts to prime numbers is about 20,000 ms on a computer configured with a 2.3 GHz Intel Core i9 16 GB 2667 MHz DDR4, which is sufficient for security, and the prime number obtained will be less than 2^{64}. For cases within 2^{64}, Jim Sinclair finds the best set of test credentials: [2, 325, 9375, 28,178, 450,775, 9,780,504, 1,795,265,022] [31]. The scheme uses these credentials instead of randomly selected credentials. The following Algorithm 7 shows the algorithm.

3.4.2. Construction of the Accumulator

After mapping the address information to prime numbers, we add these prime numbers to an RSA accumulator and then store the obtained accumulator value in the smart contract. As the prime elements can be very large, using the construction of Benjamin Wesolowski [32] may lead to overflow due to excessive exponents. Therefore, the RSA accumulator is constructed in the following way: Initialize the accumulator as g, then add the mapped primes P = $\{p_1, p_2, \ldots, p_m\}$ to the accumulator to obtain a new accumulator value acc, where n and g are the public keys generated by the user, as shown in Equation (1):

$$acc = g^s \pmod{n}, \quad S = \prod_{i=1}^{m} p_i. \tag{1}$$

For an element x in the set P, a membership witness is generated, as shown in Equation (2):

$$\pi = g^{s/x} \pmod{n}. \tag{2}$$

The membership witness is then verified. If the equation holds, x must belong to the set P, as shown in Equation (3):

$$acc = \pi^x \pmod{n}. \tag{3}$$

When the user wants to authorize a new address, they map it to a prime number x' then update the accumulator value, as shown in Equation (4). The user invokes the contract to update the accumulator value. The witness of membership of other elements should be updated, as shown in Equation (5):

$$acc' = acc^{x'} \pmod{n}, \tag{4}$$

$$\pi' = \pi^{x'} \pmod{n}. \tag{5}$$

When the user revokes the authorization of an address, the extended Euclidean algorithm is first used to find the inverse r of x with respect to $\phi(n)$. Then, r is used to

update the accumulator value and the membership witnesses of other elements, as shown in Equations (6)–(8):

$$r = x^{-1} \pmod{\phi(n)}, \tag{6}$$

$$acc' = acc^{r} \pmod{n}, \tag{7}$$

$$\pi' = \pi^{r} \pmod{n}. \tag{8}$$

Algorithm 7: Deterministic variant of the Miller–Rabin primality test.

Miller–Rabin primality test
Input:
bigNumber n,k
Output:
bool *isPrime*
//Lines 1–4 are edge cases that return immediately. If n is 2 or 3, return true. If n is less than 2 or even, return false.
1: if ($n == 2 \,||\, n == 3$)
2: return true;
3: if ($n \pmod 2 == 0 \,||\, n < 2$)
4: return false;
//Lines 5–9 compute d and s, such that $n - 1 = 2^s * d$, where d is odd. This is done so that we can write n-1 as a power of 2 times an odd number.
5: $s = 0, d = n - 1$;
6: while ($d \pmod 2 == 0$) {
7: $d = d/2$;
8: $++s$;
9:}
//Line 10 initializes an array of base values to test. These are predefined values that have been shown to be effective in practice.
10: $bases = [2, 325, 9375, 28, 178, 450, 775, 9, 780, 504, 17, 955, 265, 022]$;
//The following loop (lines 11–22) is the main body of the test. It checks if n is prime. The outer loop (//line 11) repeats k times, using a different base value each time.
11: WitnessLoop: do{
//Line 12 selects the base value to use for this iteration of the test.
12: $a = bases[k - 1]$;
13: //if n is prime, n must satisfy A or B
//Lines 14–15 handle the case A. If this is true, skip to the next iteration of the outer loop.
14: if ($a^d \pmod n == 1$)//A
15: continue;
//Lines 16–19 handle the case B. If this is true, skip to the next iteration of the outer loop.
16: for($j = 0$ to $s - 1$) do {//B
17: if ($a^{2^j d} \pmod n == 1$)
18: continue WitnessLoop;
19: }
//If neither of the above cases is true, then n is composite and return false.
20: if($j == s$)
21: **return** false;
22:} while($k--$);
//If all k iterations of the outer loop without returning false, then n is probably prime and return true.
23: **return** true;

4. Analysis and Discussion

In this scheme, the identity provider only has information about each user's identity, but does not know which wallet account is authorized for that user. On the other hand, the VASP (virtual asset service provider) only knows the user's wallet account, but does not know the user's real-world identity. Therefore, neither party can link the user's identity to their wallet account, allowing the wallet to remain anonymous.

An identity NFT contains information such as the hash of the user's identity data and the scope of business in which the user is allowed to participate. These data are stored on IPFS which ensures that the identity data cannot be tampered with or denied. The NFT does not contain sensitive information, protecting the user's privacy.

The user cannot transfer their identity NFT. If the private key of the user's original admin address is lost, the user can simply contact the identity provider for brief authentication and designate a new admin address. If the user believes that their identity may be used fraudulently, they can contact the identity provider and use the contract to lock their identity. If the private key of a wallet account is lost or stolen, the user can revoke the authorization of that wallet account by updating the accumulator value in the contract data.

VASPs include NFT markets, OTC (over-the-counter) platforms, etc. By requiring users to show proof that their wallet accounts for deposits and withdrawals on the OTC platform have been bound to an identity NFT, the conversion of criminal funds between fiat and cryptocurrencies can be reduced. By requiring virtual asset service providers to verify the identity of users associated with wallet accounts, we can reduce the occurrence of crimes such as fraud and protect the rights and interests of investors. This requirement also helps to prevent money laundering, as criminals would need to use verified accounts to receive funds. By verifying the identities of users, we can ensure the security and integrity of the virtual asset ecosystem.

Regulatory agencies and criminal investigation agencies, acting as regulators, can access the user's identity data held by the identity provider and the user's wallet account held by the VASP to link real identities with wallet accounts for the purpose of sanctions or criminal investigations. This helps to ensure the security and integrity of the virtual asset ecosystem. Note that the regulator must provide evidence of malicious behavior in the account to be investigated before the identity provider grants access to its corresponding identity data. This approach ensures that innocent users are not subject to investigations by regulatory agencies. Corresponding laws, regulations, and supervisory mechanisms are necessary to prevent the concentration of power in a small number of regulatory agencies. To avoid situations where the supervisory authority has too much power and forces the identity provider to disclose user data, the identity provider should have at least the same level of power as the supervisory authority.

Tax issues pose a significant challenge in the field of crypto. This challenge arises from the volatility of crypto assets, the possibility of users holding multiple blockchain accounts, and the difficulty of valuing NFT assets, making it a challenging task to evaluate user assets. The primary objective of this solution is to track the identity of blockchain accounts that may engage in criminal behavior, and it is difficult to deal with the issue of taxation. Tax-related issues can be addressed by tracking non-custodial wallets, centralized exchanges, and other relevant platforms.

5. Conclusions

In this paper, we propose an aggregate identity authorization scheme for blockchain wallet accounts. By using the RSA accumulator instead of Merkle tree, this scheme solves the problem that the previously proposed identity management scheme cannot support batch identity authorization certificates, and the new scheme reduces the cost of provers and verifiers. Furthermore, the proposed scheme relieves the burden on VASPs by introducing third-party verifiers and eliminating attribute credentials, making the system more suitable for practical applications. Our solution establishes a permissioned environment between public blockchains and VASPs and contributes to a healthy cryptocurrency industry. In the

future, we plan to optimize the solution for efficiency and other aspects and develop an example project for demonstration purposes.

Author Contributions: N.S. was the advisor. Y.Z. designed the scheme. Y.Z. carried out the implementation. Y.Z. wrote the manuscript. N.S. and Y.L. revised the final version of the text. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Postgraduate Research and Practice Innovation Program of Jiangsu Province. Grant number: KYCX21_2832.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

VASP	Virtual asset service provider
PoR	Proof of Reserves
RSA	Rivest Shamir Adleman
ICO	Initial Coin Offering
KYC	Know your customer
NFT	Non-fungible token
FT	Fungible tokens
SNARK	Succinct Non-Interactive Argument of Knowledge
DID	Decentralized identifier
SSI	Self-Sovereign Identity
SBT	Soulbound token
EVM	Ethereum virtual machine
IPFS	InterPlanetary File System
SPV	Simplified payment verification
P2PKH	Pay-to-PublicKeyHash
OTC	Over-the-counter
IDNFT	Identity NFT

References

1. The Policy Environment for Blockchain Innovation and Adoption: 2019 OECD Global Blockchain Policy Forum Summary Report, OECD Blockchain Policy Series. Available online: https://www.oecd.org/finance/2019-OECD-Global-Blockchain-Policy-Forum-Summary-Report.pdf (accessed on 15 November 2021).
2. Binance. Rug Pull. Available online: https://academy.binance.com/en/glossary/rug-pull (accessed on 15 November 2021).
3. $570 Million Worth of Binance's BNB Token Stolen in Another Major Crypto Hack. Available online: https://www.cnbc.com/2022/10/07/more-than-100-million-worth-of-binances-bnb-token-stolen-in-another-major-crypto-hack.html (accessed on 15 November 2021).
4. Averin, A.; Samartsev, A.; Sachenko, N. Review of Methods for Ensuring Anonymity and De-Anonymization in Blockchain. In Proceedings of the 2020 International Conference Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS), Yaroslavl, Russia, 7–11 September 2020; pp. 82–87. [CrossRef]
5. Ateniese, G.; Faonio, A.; Magri, B.; de Medeiros, B. Certified Bitcoins. In *Applied Cryptography and Network Security. ACNS 2014. Lecture Notes in Computer Science*; Boureanu, I., Owesarski, P., Vaudenay, S., Eds.; Springer: Cham, Switzerland, 2014; Volume 8479, pp. 80–96. [CrossRef]
6. Li, P.; Xu, H.; Ma, T. An efficient identity tracing scheme for blockchain-based systems. *Inf. Sci.* **2021**, *561*, 130–140. [CrossRef]
7. Biryukov, A.; Khovratovich, D.; Tikhomirov, S. Privacy-preserving KYC on Ethereum. In Proceedings of the 1st ERCIM Blockchain Workshop 2018, Amsterdam, The Netherlands, 9 May 2018. [CrossRef]
8. W3c. Decentralized Identifiers (DIDs) v1.0. Available online: https://www.w3.org/TR/2021/PR-did-core-20210803/ (accessed on 15 November 2021).
9. Bernal Bernabe, J.; Canovas, J.L.; Hernandez-Ramos, J.L.; Torres Moreno, R.; Skarmeta, A. Privacy-Preserving Solutions for Blockchain: Review and Challenges. *IEEE Access* **2019**, *7*, 164908–164940. [CrossRef]

10. Weyl, E.; Puja Ohlhaver, G.; Buterin, V. Decentralized Society: Finding Web3's Soul. 2022. Available online: https://ssrn.com/abstract=4105763 (accessed on 15 November 2021).
11. Binance DID Method Specification. Available online: https://github.com/ontology-tech/DID-method-specs/blob/master/did-bnb/DID-Method-bnb.md (accessed on 15 November 2021).
12. Celo DID Method Specification. Available online: https://github.com/ontology-tech/DID-method-specs/blob/master/did-celo/DID-Method-celo.md (accessed on 15 November 2021).
13. ABT DID Protocol. Available online: https://arcblock.github.io/abt-did-spec/ (accessed on 15 November 2021).
14. Renoun. Available online: https://github.com/Jon-Becker/renoun (accessed on 15 November 2021).
15. Huffbound. Available online: https://github.com/PraneshASP/huffbound (accessed on 15 November 2021).
16. ERC1155S. Available online: https://github.com/0xGravityLabs/ERC1155S (accessed on 15 November 2021).
17. Sun, N.; Zhang, Y.; Liu, Y. A Privacy-Preserving KYC-Compliant Identity Scheme for Accounts on All Public Blockchains. *Sustainability* **2022**, *14*, 14584. [CrossRef]
18. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Bus. Rev.* **2008**, *21260*, 3440802. [CrossRef]
19. Buterin, V. A next-generation smart contract and decentralized application platform. *White Pap.* **2014**, *3*, 37.
20. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
21. Szabo, N. Formalizing and securing relationships on public networks. *First Monday* **1997**, *2*, 1. [CrossRef]
22. Yi, X.; Yang, X.; Kelarev, A.; Lam, K.Y.; Tari, Z. Bitcoin, Ethereum, Smart Contracts and Blockchain Types. In *Blockchain Foundations and Applications*; Springer: Cham, Switzerland, 2022; Volume 121, pp. 22–65. [CrossRef]
23. Fabian, V.; Buterin, V. EIP-20: Token Standard. Available online: https://eips.ethereum.org/EIPS/eip-20 (accessed on 20 August 2022).
24. Benet, J. Ipfs-Content Addressed, Versioned, p2p File System. *arXiv* **2014**, arXiv:1407.3561.
25. Benaloh, J.; de Mare, M. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In Proceedings of the EUROCRYPT'93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, 23–27 May 1993. [CrossRef]
26. Camenisch, J.; Lysyanskaya, A. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 13–15 August 2002. [CrossRef]
27. Camenisch, J.; Kohlweiss, M.; Soriente, C. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Proceedings of the International Workshop on Public Key Cryptography, Irvine, CA, USA, 18–20 March 2009. [CrossRef]
28. Method of Providing Digital Signatures. Available online: https://patentimages.storage.googleapis.com/69/ab/d9/2ff9f94fada6ea/US4309569.pdf (accessed on 5 November 2021).
29. Ozdemir, A.; Wahby, R.; Whitehat, B.; Boneh, D. Scaling verifiable computation using efficient set accumulators. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Boston, MA, USA, 12–14 August 2020. [CrossRef]
30. Marek, P.; Rusnak, P. *Multi-Account Hierarchy for Deterministic Wallets*. Bitcoin. Available online: https://bips.xyz/44 (accessed on 12 March 2023).
31. Deterministic Variants of the Miller-Rabin Primality Test. Available online: http://miller-rabin.appspot.com/ (accessed on 15 November 2021).
32. Wesolowski, B. Efficient verifiable delay functions. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, 19–23 May 2019. [CrossRef]