sustainability

Article

A Privacy-Preserving KYC-Compliant Identity Scheme for Accounts on All Public Blockchains

Nigang Sun [1], Yuanyi Zhang [1,2,*] 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 have the potential to enable socioeconomic growth throughout the world by offering easier access to capital and financial services. However, many virtual asset service providers (VASPs) that offer cryptocurrency services lack identity management and can be accessed anonymously, which has led to their services being exploited by criminal activities such as money laundering and illegal foreign exchange. Such crimes have a negative impact on socioeconomic sustainability. Building identity systems on blockchains can help VASPs improve their identity management to combat cryptocurrency-based crimes so VASPs can better serve the social economy and achieve their sustainability goals. However, existing solutions have privacy problems because the identity provider can associate users' identities with their wallet accounts. In addition, there is currently no solution that can support all public blockchains unconditionally, as current solutions can only support EVM-compliant blockchains or require additional work to support new blockchains. This article proposes a KYC (know your customer)-compliant identity scheme based on Ethereum using Merkle trees and smart contracts. The identity and wallet accounts are linked by the user rather than the KYC provider so, in general, no one but the user knows the association between the wallet accounts and the identity, which protects privacy. For suspicious accounts, supervisors can trace their identities and thus achieve supervision. In addition, the scheme supports identifying accounts on all public blockchains by using Merkle trees and smart contracts to bind accounts on multiple blockchains to one identity and no extra work is required. Moreover, the scheme supports users to prove that their attributes meet the requirements of VASPs by adopting the BBS+ signature and the Sigma protocol.

Keywords: anti-money laundering; blockchain; digital identity; privacy protection; smart contract; supervision

check for updates

Citation: Sun, N.; Zhang, Y.; Liu, Y. A Privacy-Preserving KYC-Compliant Identity Scheme for Accounts on All Public Blockchains. *Sustainability* **2022**, *14*, 14584. https://doi.org/10.3390/su142114584

Academic Editor: Cheolho Yoon

Received: 3 October 2022
Accepted: 3 November 2022
Published: 6 November 2022

1. Introduction

The rapid growth of cryptocurrency usage has led to regulatory inconsistency in jurisdictions across the world and has created opportunities for money launderers and terrorists to commit financial crimes using cryptocurrencies. Criminals laundered USD 8.6 billion of cryptocurrency in 2021, up by 30% from the previous year, according to a report by blockchain data company Chainalysis [1]. In addition, the borderless nature of cryptocurrencies also allows for unrestricted foreign exchange transactions and leads to capital flight, which can affect the sustainability of a country's economy [2]. Virtual asset service providers (VASPs) are required for almost all cryptocurrency-based crimes such as money laundering and illegal currency exchange. Therefore, as long as VASPs check the identities of their users, such crimes can be reduced. However, the reality is that most VASPs lack identity management, which becomes the biggest problem in fighting cryptocurrency-based crimes.

To solve the various crimes caused by the lack of identity management, many solutions have been proposed, which can be divided into the following three categories: identity tracing, de-anonymization, and digital identity systems. Schemes [3–7] use the method of identity tracing. After users register with the supervision center, the senders of the transactions on the blockchain can be traced back to real-world identities. Schemes [8–12] de-anonymize blockchain accounts by using methods such as address clustering based on on-chain or off-chain data. Compared with identity-tracing and de-anonymization schemes, building digital identities on blockchains not only enables identifying accounts on blockchains but also offers on-chain permission to build trust within communities of verified users [13]. Users can prove to VASPs that their identities have been verified and meet certain attribute requirements so VASPs can allow only these verified users access, which helps reduce crimes.

Digital identity schemes typically include three entities: users, identity providers (IdPs), and VASPs [14]. IdPs are responsible for checking the real identity of the user and linking it to the user's wallet account. VASPs refer to various cryptocurrency-related service providers such as exchanges. ERC725, ERC1056, etc. [15] are proposed standards for blockchain-based identity. They are standard interfaces for a smart contract-based account with an attachable key-value store and many decentralized applications (dapps) are building identity systems based on them, such as Origin Protocol [16]. W3C proposed a standard of self-sovereign identity [17]—decentralized identifiers (DIDs) [18]. Blockchain-based DID schemes [19] use each blockchain account as a DID and IdPs verify users' attributes and issue verifiable credentials so that users can present verified attribute values to validators. Soul-bound tokens (SBTs) [20] are non-transferable digital identity tokens that represent the traits, features, and achievements that make up a person or entity. BAB [21] is a soul-bound token used as proof of identity for Binance users who have completed KYC (know your customer) verification. Reputation systems [22–24] have established an on-chain credit system. They score wallet accounts by correlating off-chain data or mark the on-chain behavior of accounts by issuing certificates. As described above, IdPs in all solutions know the wallet accounts of all their clients so the asset balances and transaction records of wallet accounts are exposed, which poses a threat to personal privacy. In addition, all of the above solutions implement identity systems only on the blockchains where they are deployed. No solution currently supports all public blockchains, and criminals can still launder money using blockchains without an established identity system.

This article proposes a KYC-compliant identity scheme for Ethereum-based blockchain wallet accounts using Merkle trees and smart contracts. First, in order to solve the above privacy issues, the scheme divides the IdP of the three-party model into supervisors and KYC providers and uses a four-party model consisting of supervisors, KYC providers, users, and VASPs. The supervisor is responsible for identity tracking, and the KYC provider is responsible for checking the user's identity. In the existing solutions, since the identity provider is responsible for binding the user's identity credentials to the user's wallet account, the identity provider knows the user's asset balance and transaction records, which threatens their privacy. In this scheme, the identity provider is not responsible for binding the user's identity to wallet accounts but authorizes the user to do this. The user does not directly bind the identity to the wallet accounts but uses the Merkle tree to hide specific wallet accounts. So, no one knows which wallet account the identity is associated with unless the user shows proof. The privacy issue mentioned above is solved by this mechanism. Therefore, the scheme does not affect the anonymity of the blockchain, which is conducive to encouraging users to cooperate with the supervision scheme. For wallet accounts with illegal activities, the supervisor needs to apply to the KYC provider using procedures prescribed by law to obtain the real-world identity corresponding to the identity NFT. The supervisor then asked the relevant VASPs to provide the wallet accounts linked to the identity NFT. In this way, the supervisor can find the real-world identity corresponding to the wallet account. If the account never provides identification to any VASP, the supervisor can issue a blacklist and require all VASPs to deny service to the account to prevent the transfer of criminal funds. Second, there are currently solutions to support multiple public blockchains but there is no solution to support all public blockchains. When crypto assets are transferred to unregulated blockchains, it will be difficult to trace identities. So, all public blockchains should be regulated. In this scheme, users use multiple public blockchain accounts as leaf nodes to generate a Merkle tree and then bind the root of the Merkle tree to an identity NFT on Ethereum. By presenting the membership proof of a leaf node, the user can prove to a third party that a wallet account on Bitcoin or other blockchains is linked to the identity NFT. In this way, the scheme can support all public blockchains. In addition, the scheme adopts the BBS+ signature and the Sigma protocol to support users in proving that their attributes meet the requirements of VASPs. This scheme helps VASPs to achieve identity management and avoid becoming a tool for criminals so that VASPs can promote the social economy and achieve their sustainability goals.

2. Preliminaries

2.1. Blockchain

The article *Bitcoin: A Peer-to-Peer Electronic Cash System* [25] by Satoshi Nakamoto in 2008 introduced the concept of the blockchain. The blockchain is a chain structure composed of sequential links of blocks. The blockchain has the characteristics of decentralization, non-tampering, and transparency. Blockchain technology is consistently evolving and being applied in various fields of activity [26]. In 2014, Ethereum was proposed [27]. Ethereum allows people to deploy decentralized applications onto it based on smart contracts. Smart contracts have greatly enriched the application of blockchains.

2.2. Smart Contracts

Smart contracts were first proposed by Nick Szabo in 1995 [28]. After the emergence of Ethereum in 2014, smart contracts began to be used on a large scale. A smart contract can be defined as an agreement within the blockchain between two people. The most important part of this agreement is that it is self-executing, secure, and does not require any third-party involvement such as lawyers or banks. It also eliminates counterparty risk by allowing users to check their balances at any point in time and makes sure no one has changed their balances without mutual consent.

2.3. NFT

Fungible tokens (FT) are divisible and non-unique. For instance, fiat currencies such as the dollar are fungible: A USD 1 bill in New York City has the same value as a USD 1 bill in Miami. A fungible token can also be a cryptocurrency, such as Bitcoin: BTC 1 is worth BTC 1, no matter where it is issued. The most representative fungible token protocol is ERC20 [29]. Non-fungible tokens (NFTs) are cryptographic assets on a blockchain with unique identification codes and metadata that distinguish them from each other. Since storing data on the chain is very expensive, the metadata of NFTs are often stored on decentralized storage systems such as an Inter-Planetary File System (IPFS) [30].

2.4. Merkle Tree

A Merkle tree is a tree structure that takes the hash value of all child nodes as the value of the current node [31]. It can be proved that a leaf node belongs to a Merkle tree by providing all the hashes with the entire path from the leaf node to the root node. Merkle trees can be used for blockchain pruning and simplified payment verification (SPV).

2.5. Pedersen Commitment

The cryptographic commitment scheme is a two-party interactive protocol divided into two steps. The first step is to commit. The committer selects the message m, generates a commitment value, and sends it to the receiver to ensure that m will not be changed. The second step is the opening of the commitment. The committer discloses the message m and the blinding factor, and the receiver uses these data to verify whether m is consistent with the commitment value.

The Pederson commitment [32] is a type of commitment in cryptography. It was introduced by Torben Pryds Pedersen in 1992. The Pederson commitment is of the form $c = g^m h^r$, where m is the committed value, r is the blinding factor, and $\log_h g$ is unknown.

2.6. Bilinear Mapping

In mathematics, a bilinear map is a function combining elements of two vector spaces to yield an element of a third vector space and is linear in each of its arguments. Its specific definition is as follows.

$G1$, $G2$, and GT are three cyclic groups of order n. The bilinear mapping $e : G1 \times G2 \rightarrow GT$ satisfies three properties.

Bilinear: For all $g_1 \in G1, g_2 \in G2, a, b \in Z, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.

Non-degenerate: There exists $g_1 \in G1, g_2 \in G2$, such that $e(g_1, g_2) \neq 1$, where 1 is the identity of GT.

Efficient: There exist efficient polynomial-time algorithms for computing the values of bilinear pairs.

3. System Design

This section describes the implementation details of the scheme. Section 3.1 introduces the architecture of the scheme. Section 3.2 describes the overall flow of the scheme. Section 3.3 describes how BBS+ signatures can be used to achieve the selective disclosure of attributes. Sections 3.4 and 3.5 describe the details of using OR Proof in the Sigma protocol to implement obfuscation proofs. Section 3.6 describes how users link wallet accounts with their digital identities. Section 3.7 describes the design of the KYC smart contracts.

3.1. System Architecture

The scheme includes four entities: users, the KYC provider, VASPs, and supervisors. In the scheme, there is only one KYC provider but there may be multiple supervisors. The KYC provider is a government department that manages users' identities. The supervisors are government departments different from the KYC provider, such as inspection agencies responsible for crime investigation. VASPs include exchanges, OTC (over-the-counter) platforms, etc. The scheme architecture is shown in Figure 1.

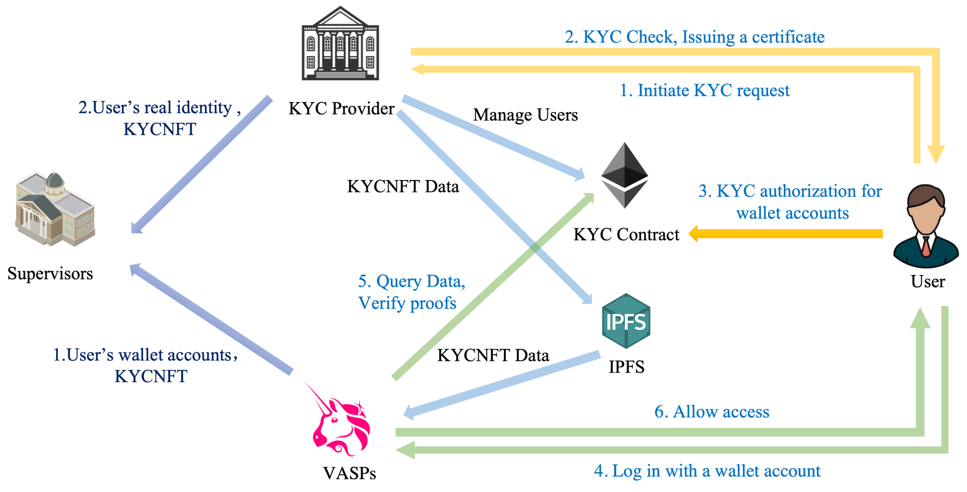


Figure 1. Scheme architecture.

3.2. Program Flow

This section describes the flow of the scheme. A brief description of the symbols used is listed in Table 1. In the following, variables are written in italics.

Table 1. Notation: Short description.

Abbreviation	**Description**
BBSSK	The private key of the BBS + signature.
BBSPK, PK	The public key of the BBS + signature.
BBSSig	The BBS+ signature.
ManagerAddr	The manager account.
MerkleProof	A membership proof for a Merkle tree leaf node.
MerkleRoot	The root of the Merkle tree.
SigmaData	Public data for the Sigma protocol.
fileUrl	The storage path of a file in IPFS.
NFTId	The id of the user's identity NFT.

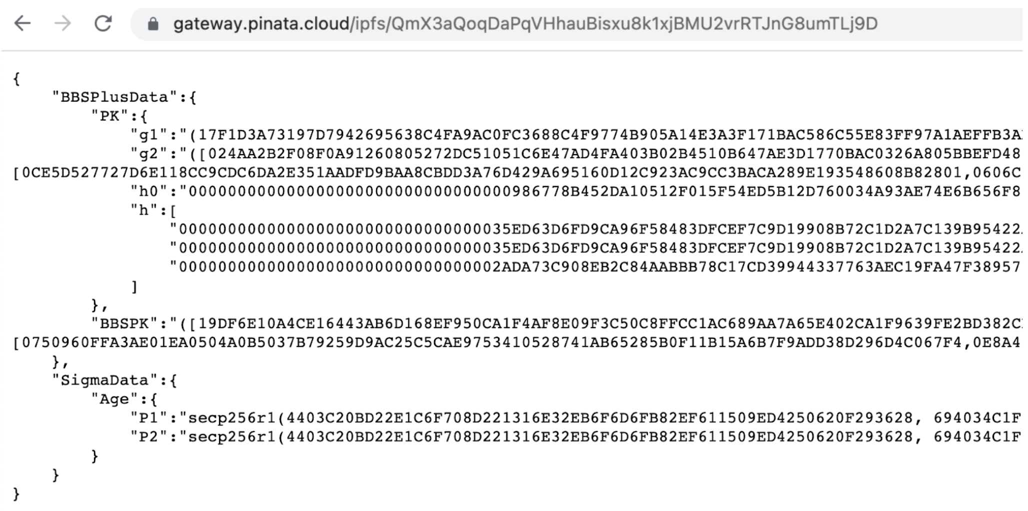
(1) The KYC provider deploys the KYCManager contract and the KYCNFT contract to the Ethereum network.

(2) The user initiates a KYC request to the KYC provider and submits the identity information required for KYC, such as an ID card. Then, the KYC provider checks the user's identity information. After passing the KYC check, the user generates an Ethereum wallet account, denoted as *ManagerAddr*. This account will be used as the user's manager account for the KYC authorization of wallet accounts and will be submitted to the KYC provider.

(3) If the user passes the KYC check, the KYC provider does the following. The KYC provider generates a BBS+ signature for a list of the user's attribute values, denoted as *BBSSig*, and sends it to the user. The public keys used for the BBS+ signature are *BBSPK* and *PK*.
Disjunctive proofs (OR proofs) in the Sigma protocol can be used to construct the obfuscating proofs of the attribute values, proving that the actual attribute value belongs to a specific set without exposing its value. Building an obfuscation proof requires some fake data for obfuscation so the KYC provider generates some fake data for the user, denoted as *SigmaData*.
The KYC provider stores the *BBSPK*, *PK*, and *SigmaData* in the IPFS and obtains the storage path *fileUrl*, as shown in Figure 2.
Then, the KYC provider calls the function createKYCNFT() of the KYCManager contract to mint a KYCNFT, where the tokenURL of the KYCNFT is *fileUrl* and the owner of the KYCNFT is the KYCManager contract. In addition, the function createKYCNFT() also sets the validity of the user's identity and binds the user's manager account *ManagerAddr* to *NFTId*.
The process of KYC registration is shown in Figure 3.

(4) After obtaining the response from the KYC provider, the user authorizes his wallet accounts with the identity. First, the user adds a blockchain scope to be authorized as a suffix to each wallet account (e.g., 0x123@ETH) and then generates a Merkle tree with each wallet account as a leaf node. The root node value of the Merkle tree is *MerkleRoot*. The user's manager account calls the KYCManager contract to link *MerkleRoot* with *NFTId*. The KYC authorization process of the wallets is shown in Figure 4.

(5) The user initiates a login request to the VASP using a wallet account and provides a Merkle membership proof *MerkleProof* that proves the account is in the Merkle tree corresponding to a certain *NFTId*. According to the attribute requirements announced by the VASP, the user selectively discloses some attributes using the BBS+ signature of all attribute values. According to the *SigmaData* provided by the KYC provider, the user can generate an obfuscating proof for an attribute to prove that the attribute value belongs to a certain set without revealing which value it is.


gateway.pinata.cloud/ipfs/QmX3aQoqDaPqVHhauBisxu8k1xjBMU2vrRTJnG8umTLj9D

```
{
    "BBSPlusData":{
        "PK":{
            "g1":"(17F1D3A73197D7942695638C4FA9AC0FC3688C4F9774B905A14E3A3F171BAC586C55E83FF97A1AEFFB3A
            "g2":"([024AA2B2F08F0A91260805272DC51051C6E47AD4FA403B02B4510B647AE3D1770BAC0326A805BBEFD48
[0CE5D527727D6E118CC9CDC6DA2E351AADFD9BAA8CBDD3A76D429A695160D12C923AC9CC3BACA289E193548608B82801,0606C
            "h0":"000000000000000000000000000000000986778B452DA10512F015F54ED5B12D760034A93AE74E6B656F8
            "h":[
                "00000000000000000000000000000000035ED63D6FD9CA96F58483DFCEF7C9D19908B72C1D2A7C139B95422
                "00000000000000000000000000000000035ED63D6FD9CA96F58483DFCEF7C9D19908B72C1D2A7C139B95422
                "000000000000000000000000000000002ADA73C908EB2C84AABBB78C17CD39944337763AEC19FA47F38957
            ]
        },
        "BBSPK":"([19DF6E10A4CE16443AB6D168EF950CA1F4AF8E09F3C50C8FFCC1AC689AA7A65E402CA1F9639FE2BD382C
[0750960FFA3AE01EA0504A0B5037B79259D9AC25C5CAE9753410528741AB65285B0F11B15A6B7F9ADD38D296D4C067F4,0E8A4
    },
    "SigmaData":{
        "Age":{
            "P1":"secp256r1(4403C20BD22E1C6F708D221316E32EB6F6D6FB82EF611509ED4250620F293628, 694034C1F
            "P2":"secp256r1(4403C20BD22E1C6F708D221316E32EB6F6D6FB82EF611509ED4250620F293628, 694034C1F
        }
    }
}
```

Figure 2. Data used to generate or verify attribute proofs.

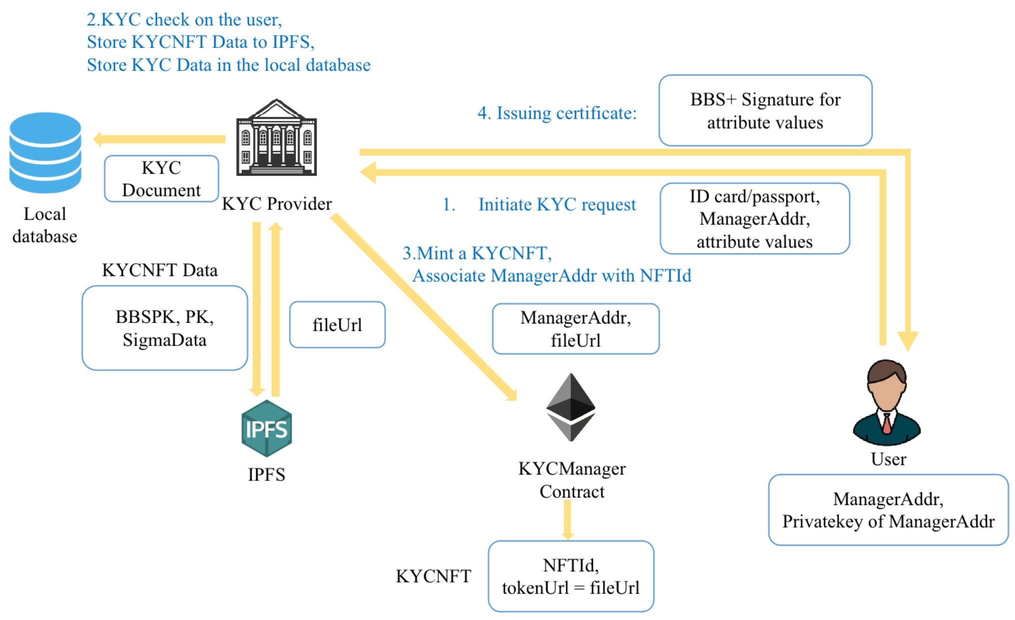

2.KYC check on the user,
Store KYCNFT Data to IPFS,
Store KYC Data in the local database
4. Issuing certificate:
BBS+ Signature for attribute values
Local database
KYC Document
KYC Provider
1. Initiate KYC request
ID card/passport, ManagerAddr, attribute values
3.Mint a KYCNFT, Associate ManagerAddr with NFTId
KYCNFT Data
BBSPK, PK, SigmaData
fileUrl
ManagerAddr, fileUrl
IPFS
IPFS
KYCManager Contract
User
ManagerAddr, Privatekey of ManagerAddr
KYCNFT
NFTId, tokenUrl = fileUrl

Figure 3. KYC Registration.

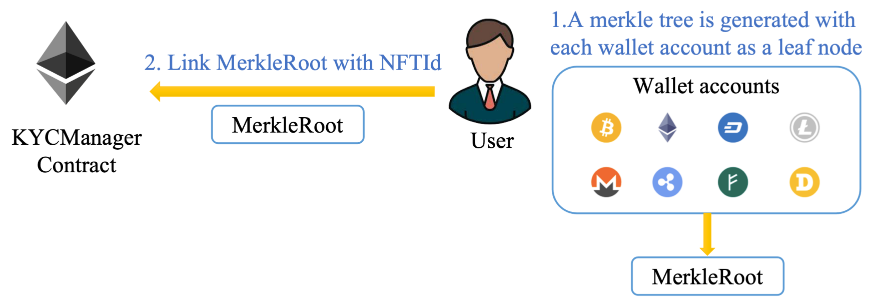

1.A merkle tree is generated with each wallet account as a leaf node
2. Link MerkleRoot with NFTId
KYCManager Contract
MerkleRoot
User
Wallet accounts
MerkleRoot

Figure 4. KYC authorization of wallet accounts.

When receiving the user's login request, the VASP verifies the KYC authorization proof (the membership proof that the wallet account belongs to a Merkle tree), the attribute signature, and the attribute proofs. Then the VASP calls the KYCManager contract to calculate the root hash according to the *MerkleProof* and check if the root hash is equal to the *MerkleRoot* corresponding to the *NFTId* of the user. The VASP also needs to check if the blockchain scope suffix contains the target blockchain. Finally, the VASP calls the contract to check whether the user identity is valid. If the user passes the check, the VASP allows the user access. The process of user access to the VASP is shown in Figure 5.

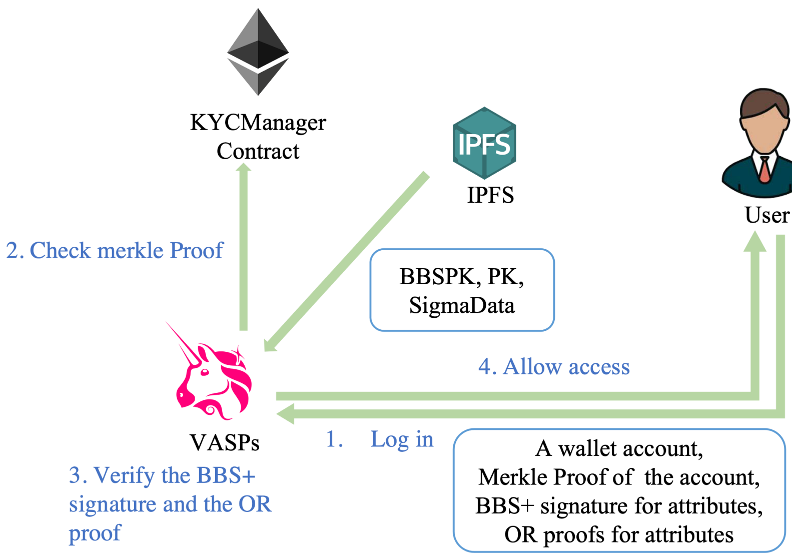

KYCManager Contract
IPFS
User
2. Check merkle Proof
BBSPK, PK, SigmaData
4. Allow access
VASPs
1. Log in
3. Verify the BBS+ signature and the OR proof
A wallet account, Merkle Proof of the account, BBS+ signature for attributes, OR proofs for attributes

Figure 5. Access to VASPs.

(6) By obtaining the user's real identity data and *NFTId* from the KYC provider and the user's wallet account and *NFTId* from the VASP, the supervisor can associate the wallet account with the real identity.

3.3. BBS+ Signature

In addition to the core properties of digital signature schemes, BBS+ signatures provide additional unique properties, namely selective disclosure, unlinkable, and proof of possession. Selective disclosure means that the prover can disclose only part of the message while not revealing other information. Unlinkable implies that the verifier who receives the proof cannot determine which signature it comes from so it is impossible to correlate any two proofs. A prover provides the verifier with a proof that they have the signature without revealing it, called proof of possession. This section briefly describes the process of the BBS+ signature, and the scheme is from Section 4.5 of the paper [33].

(1) Key generation: The public key data include three finite groups, $G1$, $G2$, and GT, where the order of $G1$ and $G2$ are both prime p. It also includes a bilinear pair $e : G1 \times G2 \to GT$. Take generators $g_1 \in G1$, $g_2 \in G2$. Let the number of signed messages be l, the KYC provider randomly selects $h_0, \ldots, h_l \in G1^{l+1}$ and $x \in {Z_p}^*$. Then, the KYC provider sets the private key $BBSSK = x$ and calculates the public keys $BBSPK = {g_2}^x$ and $PK = (g_1, g_2, h_0, \ldots, h_l)$.

(2) Signature: The KYC provider inputs the user-encoded attributes $m_1, \ldots, m_l \in {Z_p}^*$, where each m_i is obtained by mapping the statement of the attribute to Z_p using a hash function. For example, mapping the attribute statement "Birthyear = 1970" to Z_p will obtain a number m = 2922 . . . 6465. If two users have the same attribute value, they will receive the same m. Therefore, it does not hide the attribute value. Randomly select $\varepsilon, s \in Z_p$ and calculate A, as shown in Equation (1):

$$A = \left[g_1 \cdot {h_0}^s \cdot \prod_{i=1}^{l} {h_i}^{m_i} \right]^{\frac{1}{\varepsilon + x}} \tag{1}$$

Then, the signature $BBSSig = (A, \varepsilon, s)$ is sent to the user.

(3) Verification: The VASP finds the *tokenURL* of the KYCNFT corresponding to the user's *NFTId*. Get $BBSPK$ and PK from *tokenURL*. Verify the signature using the

attribute statements, the attribute-encoded set $\{m_1, \ldots, m_l\}$, and the signature $BBSSig$, as shown in Equation (2):

$$e(A, BBSPK \cdot g_2{}^{\varepsilon}) = e(g_1 \cdot h_0{}^{s} \cdot \prod_{i=1}^{l} h_i{}^{m_i}, g_2) \tag{2}$$

(4) Users can only disclose some of the attributes. Suppose the user discloses some attribute values to the verifier; their index is denoted by the set D and the verifier maps these attributes to Z_p and obtains the set of messages to be verified, denoted as $\{m_i | i \in D\}$.
Randomly select $r_1, r_2 \in Z_p$, set $r_3 = \frac{1}{r_1}$, $s' = s - r_2 \cdot r_3$, $b = g_1 \cdot h_0{}^{s} \cdot \prod_{i=1}^{l} h_i{}^{m_i}$, and calculate A', A'', d, as shown in Equations (3)–(5).

$$A' = A^{r_1} \tag{3}$$

$$A'' = A'^{-\varepsilon} \cdot b^{r_1} \tag{4}$$

$$d = b^{r_1} \cdot h_0{}^{-r_2} \tag{5}$$

The blinded signature (A', A'', d) is obtained. The following two Equations (6) and (7) need to be proved using any suitable zero-knowledge protocol ZKP.

$$\frac{A''}{d} = A'^{-\varepsilon} \cdot h_0{}^{r_2} \tag{6}$$

$$g_1 \cdot \prod_{i \in D} h_i{}^{m_i} = d^{r_3} \cdot h_0{}^{-s'} \cdot \prod_{i \in \{1,2,\ldots,l\} \setminus D} h_i{}^{-m_i} \tag{7}$$

The verifier can compute the left side of the above two equations but not the right side of the equations. If the zero-knowledge proof ZKP passes the verification, the verifier can ensure that (A', A'', d) is created using the elements of the BBS+ signature, which means that A in the signature before blinding is in the correct form, as shown in Equations (8) and (9).

$$A = b^{\frac{1}{\varepsilon + x}} \tag{8}$$

$$b = g_1 \cdot h_0{}^{s} \cdot \prod_{i \in \{1,2,\ldots,l\}} h_i{}^{m_i} \tag{9}$$

In addition to verifying the ZKP, the verifier also needs to verify $A' \neq 1_{G1}$ and the blinded signature, as shown in Equation (10).

$$e(A', BBSPK) = e(A'', g_2) \tag{10}$$

If Equations (6) and (7) pass verification, it proves that the attributes disclosed by the user are indeed a subset of all signed attribute values. If the blinded signature (A', A'', d) also passes verification, the verifier can trust the attributes claimed by the user.

3.4. Extend the OR Proof of the Sigma Protocol

Sigma protocols are used to prove knowledge of values in some relation without disclosing the values themselves. The earliest Sigma protocols were interactive and were divided into three phases: commitment, challenge, and response [34]. The interactive proof requires both the prover and the verifier to be online at the same time. The Fiat–Shamir heuristic [35] can be used to eliminate the challenge step, thus converting the interactive protocol into a non-interactive one and adapting it to practical applications. This article uses the non-interactive disjunctive proof (OR Proof) in the Sigma protocol to implement the obfuscating proof of the attribute values.

The Sigma protocol describes a non-interactive OR proof that can prove that the prover knows one of two secret values but cannot determine exactly which one. Extending the protocol to the case of n secret values, the verifier can believe that the prover knows at least one of the n secrets. There are n relations $\{{g_i}^{a_i} = P_i, i \in [1, n]\}$. Assuming that the prover knows the secret value a_k in the kth relation. The protocol process is as follows:

(1) The prover generates a random number r and then computes the commitment t_k, as shown in Equation (11).

$$t_k = {g_k}^r \tag{11}$$

Choose $\{c_i \mid i \in [1, n], i \neq k\}$ randomly as the challenges and random values $\{s_i \mid i \in [1, n], i \neq k\}$ as the responses. Calculate the commitments $\{t_i \mid i \in [1, n], i \neq k\}$, as shown in Equation (12).

$$t_{i\in[1,n],i\neq k} = {g_i}^{s_i} \cdot {P_i}^{-c_i} \tag{12}$$

Then, calculate c and c_k, as shown in Equations (13) and (14).

$$c = Hash(g_1, \ldots, g_n, P_1, \ldots, P_n, t_1, \ldots, t_n) \tag{13}$$

$$c_k = c - \sum_{i\in[1,n],i\neq k} c_i \tag{14}$$

Using c_k as the challenge for the kth proof, calculate the response for the kth proof, as shown in Equation (15).

$$s_k = r + a_k \cdot c_k \tag{15}$$

The prover sends the proof to the verifier, as shown in Equation (16).

$$\{(t_i, c_i, s_i), i \in [1, n]\} \tag{16}$$

(2) The verifier checks whether all proofs hold, as shown in Equations (17) and (18).

$${g_i}^{s_i} = {P_i}^{c_i} \cdot t_i \tag{17}$$

$$\sum_{i\in[1,n]} c_i = Hash(g_1, \ldots, g_n, P_1, \ldots, P_n, t_1, \ldots, t_n) \tag{18}$$

For the term $i \neq k$, the equation holds, as shown in Equation (19).

$${g_i}^{s_i} = {P_i}^{c_i} \cdot {g_i}^{s_i} \cdot {P_i}^{-c_i} = {P_i}^{c_i} \cdot t_i \tag{19}$$

For the kth term, the equation also holds, as shown in Equation (20).

$${g_k}^{s_k} = {g_k}^{a_k \cdot c_k + r} = {P_k}^{c_k} \cdot t_k \tag{20}$$

The verifier can only verify the proof and cannot distinguish which secret the verifier possesses.

3.5. Obfuscating Proofs for Discrete Finite Set Attributes

The protocol described in Section 3.4 can prove that an attribute value belongs to a certain set without revealing which value it is. The following describes how to use this protocol to generate an obfuscation proof for an attribute.

(1) Let g be a generator of the cyclic group G. The user randomly generates a secret key $sk = a$ and sends $P_k = g^a$ to the KYC provider.
(2) The KYC provider randomly generates $P_1, \ldots, P_{k-1} \in G$ as fake data. Each P_i corresponds to an attribute value $Attr_i$. The value P_k corresponds to the user's real attribute value. Write this k tuple $\{(P_i : Attr_i)|i \in [1, k]\}$ to a JSON file as *SigmaData*; the *SigmaData* will be stored in the data of the user's KYCNFT.

(3) According to the intersection of the attribute set published by the VASP and the attribute set in *SigmaData*, the user creates an OR proof as described in Section 3.4 to prove that his attribute value belongs to the set of attribute values that meets the requirements.
(4) The VASP verifies the proof, checks whether the attribute values in the proof all belong to the required set of attribute values and if so, allows the user to access it.

If the real attribute value of the prover belongs to the set of attribute values in the proof, the proof can be correctly generated, otherwise, a valid proof cannot be generated.

3.6. KYC Authorization of Wallet Account

The solution uses the Merkle tree to manage the authorization of wallet accounts. Many different blockchains use the same form of wallet accounts. For example, on Ethereum and other networks compatible with the Ethereum Virtual Machine (EVM), public addresses all share the same format: they begin with 0x and are followed by 40 alphanumeric characters (numerals and letters), adding up to 42 characters in total. The scheme adds the blockchain scope to be authorized to the wallet account in the following form:

$$AddrInfo = 0x4c944\ldots C8A7@ETH@BSC$$

By default, all available blockchains are authorized without setting the authorization scope. A Merkle tree is generated by taking all *AddrInfo* as leaf nodes, and the Merkle tree's root node value is *MerkleRoot*. The processes are shown in Figure 6.

When a user wants to prove that his wallet account has been KYC authorized, he provides all the hashes along the entire path from the leaf node to the root node as proof. The verifier calls the verification function provided by the KYCManager contract to verify that the resulting hash value is equal to the *MerkleRoot* value associated with the user. For example, if you want to prove *AddrInfo1* is in the Merkle tree, you need to provide *(Node2,Node6)* as proof. The verifier computes $root = hash(hash(hash(AddrInfo1), Node2), Node6)$, and if the value *root* is equal to the value of the *MerkleRoot* associated with the user's KYCNFT, it can be believed that the user has indeed authorized the wallet account.

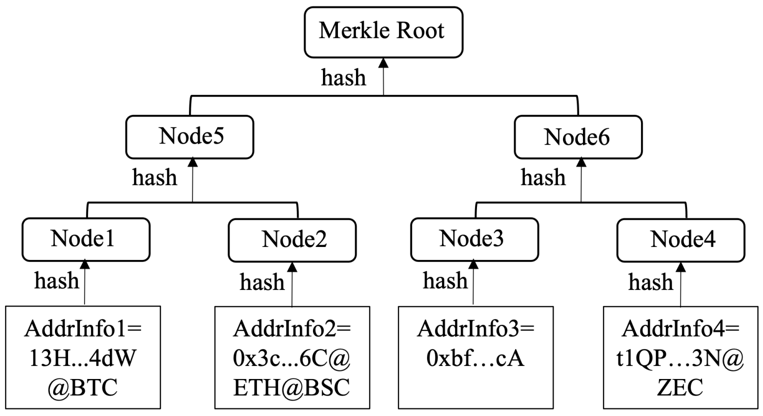
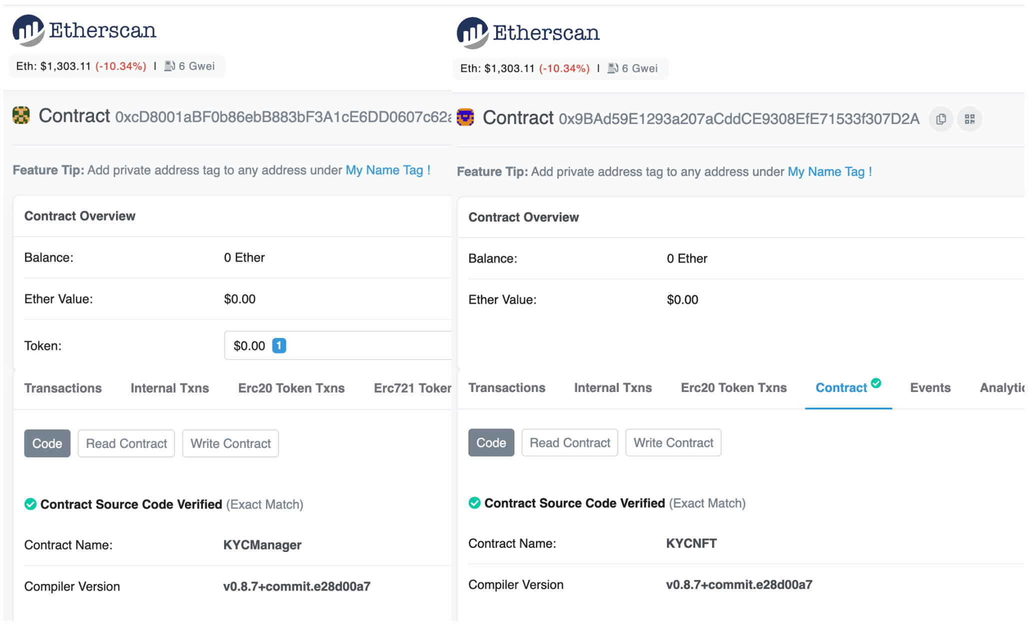


Figure 6. Merkle tree generated from wallet accounts.

3.7. KYC Smart Contracts

The scheme includes two contracts, the KYCNFT and the KYCManager. The smart contracts were written in the solidity language and the solidity version used was 0.8.7+. The two contracts were deployed on an Ethereum mainnet, as shown in Figure 7.


Etherscan
Eth: $1,303.11 (-10.34%) | 6 Gwei

Contract 0xcD8001aBF0b86ebB883bF3A1cE6DD0607c62a

Feature Tip: Add private address tag to any address under My Name Tag !

Contract Overview

Balance: 0 Ether

Ether Value: $0.00

Token: $0.00 1

Transactions | Internal Txns | Erc20 Token Txns | Erc721 Toker

Code | Read Contract | Write Contract

Contract Source Code Verified (Exact Match)

Contract Name: KYCManager

Compiler Version v0.8.7+commit.e28d00a7

Etherscan
Eth: $1,303.11 (-10.34%) | 6 Gwei

Contract 0x9BAd59E1293a207aCddCE9308EfE71533f307D2A

Feature Tip: Add private address tag to any address under My Name Tag !

Contract Overview

Balance: 0 Ether

Ether Value: $0.00

Transactions | Internal Txns | Erc20 Token Txns | Contract | Events | Analytic

Code | Read Contract | Write Contract

Contract Source Code Verified (Exact Match)

Contract Name: KYCNFT

Compiler Version v0.8.7+commit.e28d00a7

Figure 7. Smart contracts deployed on the Ethereum mainnet.

The KYCNFT contract is used to mint the user's identity NFT, similar to a general ERC721 standard NFT contract. The KYCManager contract has three purposes: the KYC provider manages the user's identity NFT through it, users use it to manage the KYC authorization of their wallet accounts, and VASPs use it to verify the KYC authorization proofs submitted by users, as shown in Figure 8.

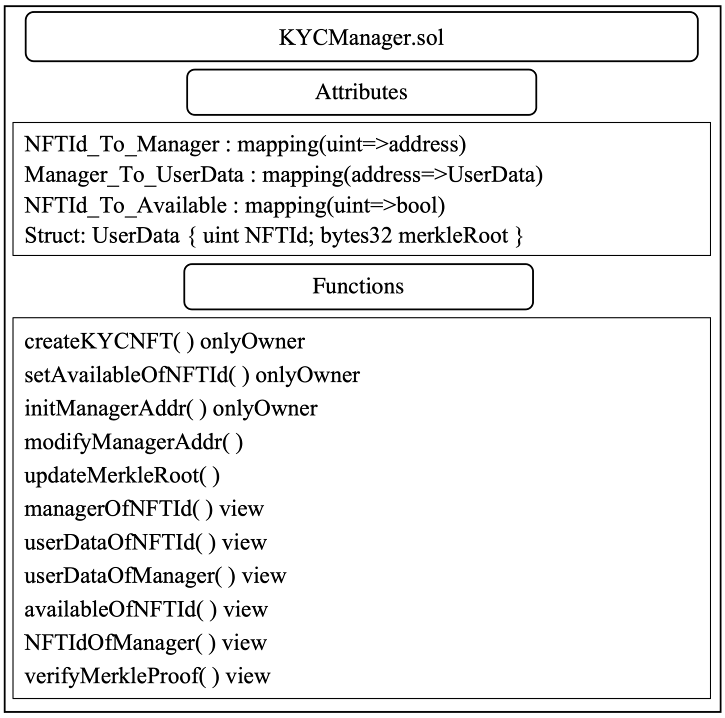
The function section lists the main functions of the contract. The functions modified with *onlyOwner* can only be called by the owner of the contract and the owner of the KYCManager contract is the KYC provider.

In the function createKYCNFT(), a KYCNFT is minted and the owner of the KYCNFT is the KYCManager contract. Then, the user's manager account *ManagerAddr* is linked to the KYCNFT via the mapping $NFTId\_To\_Manager$. Finally, the validity of the identity is set to true by setting the mapping $NFTId\_To\_Available$, as shown in Algorithm 1.

Algorithm 1 createKYCNFT()

```
Input: $string\ tokenUrl,\ address\ ManagerAddr$
  $KYCManagerContract \leftarrow KYCNFTContract.mintNFT(tokenUrl)$    ▷ Mint a KYCNFT.
  $NFTId\_To\_Manager[NFTId] \leftarrow ManagerAddr$ ▷ Bind the ManagerAddr to the KYCNFT.
  $NFTId\_To\_Available[NFTId] \leftarrow True$    ▷ Set identity to valid.
```

The transaction record of createKYCNFT() is shown in Figure 9, where the value of *tokenUrl* is set to the value of *fileURL* (a hash address starting with QM) from Figure 2.



KYCManager.sol

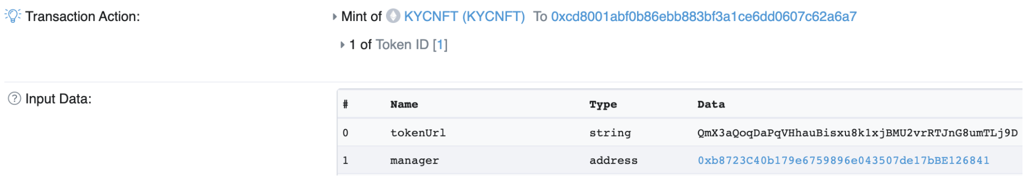
Attributes

NFTId_To_Manager : mapping(uint=>address)
Manager_To_UserData : mapping(address=>UserData)
NFTId_To_Available : mapping(uint=>bool)
Struct: UserData { uint NFTId; bytes32 merkleRoot }

Functions

createKYCNFT( ) onlyOwner
setAvailableOfNFTId( ) onlyOwner
initManagerAddr( ) onlyOwner
modifyManagerAddr( )
updateMerkleRoot( )
managerOfNFTId( ) view
userDataOfNFTId( ) view
userDataOfManager( ) view
availableOfNFTId( ) view
NFTIdOfManager( ) view
verifyMerkleProof( ) view

Figure 8. KYCManager Contract.



Transaction Action: Mint of KYCNFT (KYCNFT) To 0xcd8001abf0b86ebb883bf3a1ce6dd0607c62a6a7
1 of Token ID [1]

Input Data:

#	Name	Type	Data
0	tokenUrl	string	QmX3aQoqDaPqVHhauBisxu8k1xjBMU2vrRTJnG8umTLj9D
1	manager	address	0xb8723C40b179e6759896e043507de17bBE126841

Figure 9. The transaction record of createKYCNFT().

The function setAvailableOfNFTId() is used to set the user's identity validity by setting the mapping $NFTId\_To\_Available$. This function allows the KYC provider to freeze or unfreeze a user's identity at any time, as shown in Algorithm 2.

Algorithm 2 setAvailableOfNFTId()

```
Input: $bool\ ifAvailable$
  $NFTId\_To\_Available[NFTId] \leftarrow ifAvailable$    ▷ Set the validity of the identity.
```

The functions initManagerAddr() and modifyManagerAddr() are used to update the manager account of the user; the first function initManagerAddr() can only be called by the owner of the contract and the second function modifyManagerAddr() allows users to modify the manager account themselves. The two functions have the same logic, as shown in Algorithm 3.

Algorithm 3 modifyManagerAddr()

```
Input: $address\ newManagerAddr$
  $userData \leftarrow Manager\_To\_UserData[msg.sender]$    ▷ Find the user data of the user, the msg.sender should be the current $ManagerAddr$ of the user.
  $Manager\_To\_UserData[newManagerAddr] \leftarrow userData$    ▷ Bind the $newManagerAddr$ to the user data.
  $NFTId\_To\_Manager[userData.NFTId] \leftarrow newManagerAddr$    ▷ Bind the $newManagerAddr$ to the KYCNFT.
```

The user can call updateMerkleRoot() to update the Merkle root value of the authorized wallet accounts, as shown in Algorithm 4.

Algorithm 4 updateMerkleRoot(bytes32 *MerkleRoot*)

```
Input: $bytes32\ MerkleRoot$
  $userData \leftarrow Manager\_To\_UserData[msg.sender]$    ▷ Find the user data of the user, the msg.sender should be the ManagerAddr of the user.
  $userData.merkleRoot \leftarrow MerkleRoot$    ▷ Update the merkleRoot.
```

Functions declared with *view* can only read the state data of the blockchain but cannot modify it. The function verifyMerkleProof() is used to verify the proof of membership on a Merkle tree, as shown in Algorithm 5. The parameter positions represent the positional relationship between the two child node values; $positions = 0$ means it is on the left, and $positions = 1$ means it is on the right. The rest of the functions declared with *view* are used to query the data.

Algorithm 5 verifyMerkleProof()

```
Input: $bytes32\ leaf,\ bytes32[]\ proof,\ uint256[]\ positions,\ uint256\ NFTId$
  $computedHash = leaf$
  for $(i = 0; i \leq proof.length; i++)$ do
    if $(positions[i] == 1)$ then
      $computedHash \leftarrow keccak256(computedHash, proof[i])$
    else
      $computedHash \leftarrow keccak256(proof[i], computedHash)$
    end if
  end for    ▷ Compute the Merkel root from user-submitted proof data.
  $managerAddr \leftarrow NFTId\_To\_Manager[NFTId]$
  $userdata \leftarrow Manager\_To\_UserData[managerAddr]$    ▷ Get the user data associated with NFTId.
Output: $userdata.merkleRoot == computedHash$
```

4. Privacy and Security Analysis

(1) Privacy Analysis
Blockchain digital identity schemes mainly include two aspects of privacy issues—identity privacy and wallet account privacy. The IdP knows both users' identities and wallet accounts in all existing systems, which poses a threat to privacy. The scheme divides IdPs into supervisors and the KYC provider. Supervisors are responsible for tracing the identities of suspicious accounts, and the KYC provider is responsible for checking the identity of users. The association of identity and wallet accounts is done by the user. Supervisors can investigate the identities corresponding to suspicious wallet accounts through the information held by the KYC provider and VASPs. Therefore, for normal users, their privacy is preserved because no one but themselves knows the association between their wallet account and their real-world identity.

(2) Security Analysis
Impersonation of identity and loss of private keys are the main threats to this type of solution.
Due to the collision resistance of the hash function, it is difficult to find two wallet accounts that can generate the same hash value, so the Merkle membership proofs of wallet accounts are difficult to forge or fraudulently use. In addition, if the private key of the manager account is stolen, the attacker needs to initiate a transaction to update the *MerkleRoot* to authorize his wallet account. When users notice suspicious transactions, they can contact the KYC provider to call the contract to freeze their

identity and designate a new manager account. Therefore, the risk of a user's identity being maliciously linked to other wallet accounts is low.

In existing schemes, identities and wallets are generally statically bound, for example, blockchain-based DID schemes use wallet accounts as DIDs, whereas in SBT schemes, identities are bound to wallet accounts by issuing NFTs. Once the wallet's private key is lost, the identity will also be lost. In the scheme, the KYCManager contract owns each KYCNFT so the KYCNFT is non-transferable and will not be lost. The wallet account and the KYCNFT are dynamically bound so the identity is still available even if the private key of the wallet account is lost. If a user loses the private key of the manager account, he can contact the KYC provider to set up a new manager account. If the private key of an authorized wallet account is leaked or lost, the user can update *MerkleRoot* to revoke the authorization of this wallet account.

5. Discussion

According to the definition of identity, blockchain identity schemes can be classified as transaction identities based on wallet accounts, personality identities based on personalities, credential identities based on social behavior or asset ownership, reputational identities based on reputations, and data identities based on real-world identity information.

This article discusses the relationship between on-chain accounts and off-chain entities, but transaction identities are only related to on-chain transaction data so they are not within the scope of the discussion.

Personality identity schemes, such as BrightID [36] and Proof of Humanity [37], aim to prove that the owner of a wallet account is a real person. Such schemes can be used to prevent Sybil attacks. However, since there is no centralized third party to record identity data, these solutions are only used for on-chain governance and cannot meet the requirements of regulatory policies.

Credential identity schemes, such as POAP [38], Ethereum Name Service [39], etc., are based on social behavior and asset ownership. Reputational identities such as Project Galaxy are based on off-chain reputation data such as credit scores. These schemes generally identify identities by issuing NFTs to wallet accounts. However, if the wallet's private key is stolen, the identity NFT may also be stolen. SBT solutions solve this problem by issuing non-transferable NFTs but when the wallet's private key is lost, the NFT will also be lost. Social recovery [40] is based on multi-signatures and can be used for private key recovery or to approve a multi-signature public key to transfer the assets of the current account; the latter is only available in Turing-complete blockchains. In the case of a private key loss, the private key can be recovered by several private keys held by others. Since the process of recovering the key is independent of the blockchain, it can support any blockchain. In the case of private key theft, a signature can be generated by several private keys held by others to transfer assets to a new account, but this is only valid on blockchains that support smart contracts. As analyzed in the Privacy and Security Analysis section, in this scheme, the identity NFT is dynamically bound to the wallet account, which solves the problems of private key theft and loss.

Reputation identities generally only include some score indicators, whereas data identities include complete identity information and are more practical. The most representative data identity solutions are DIDs. The DID solutions use each wallet account as a decentralized identity. Such schemes do not require the blockchain to be Turing-complete and can be applied to almost all public blockchains. However, to support multiple blockchains, certificate issuers need to issue verifiable credentials separately for accounts on each blockchain. In addition, this creates privacy issues because the certificate issuer knows the asset balance and transaction history of each wallet account held by the user. Compared with these schemes, the scheme solves this privacy problem while supporting all public blockchains. The identity provider does not know which wallet accounts are bound to the identity. Additionally, the scheme does not require IdPs to do extra work to support newly launched blockchains. This allows the scheme to be directly applied to new blockchains without waiting for support from IdPs.

The scheme was compared with other blockchain identity schemes according to several evaluation metrics, as shown in Table 2.

Table 2. Comparison of schemes.

	1. Personality Identity	**2. Credential Identity**	**3. Reputational Identity**	**4. Data Identity**	**5. The Scheme**
Relational Pattern	Personality Verification	Ownership or Social Behaviors	Credit score, contribution value, etc.	Identity data	Identity data
Is it in compliance with the regulatory policy?	No	No	Yes for partial solutions	Yes for partial solutions	Yes
Is the wallet account anonymous to the IdP?	No	No	No	No	Yes
Is the wallet account anonymous to the VASP?	Yes	Yes	Yes	Yes	Yes
Does it support multiple blockchains?	No	Some name-service solutions support multi-chain	Only Turing-complete blockchains are supported	DID schemes support all public chains but require IdPs to do extra work	Supports all public blockchains and does not require IdPs to do extra work
Will the loss of the wallet's private key affect the use of the identity?	Partial solutions support social recovery	Partial solutions support social recovery	Partial solutions support social recovery	Partial solutions support social recovery	No
Will the theft of the wallet's private key affect the use of the identity?	Partial solutions support social recovery (only on Turing-complete blockchains)	Partial solutions support social recovery (only on Turing-complete blockchains)	Partial solutions support social recovery (only on Turing-complete blockchains)	Partial solutions support social recovery (only on Turing-complete blockchains)	No

6. Applications and Socioeconomic Implications

Cryptocurrencies built on public blockchains have been used in many fields, which has brought significant changes to societies around the world. However, since many VASPs lack identity checks on users who use their services, criminals can anonymously use the services provided by these VASPs for criminal purposes such as money laundering and illegal foreign exchange. According to related reports, the amount of money laundered by cybercriminals globally through cryptocurrencies reached USD 8.6 billion in 2021, a 30% increase from 2020. In addition, there are many scams such as Rug pull because many VASPs are anonymous and the interests of investors cannot be guaranteed. According to Chainalysis, in 2021, scams were once again the largest form of cryptocurrency-based crime by transaction volume, with over USD 7.7 billion worth of cryptocurrency taken from victims worldwide. As mentioned above, cryptocurrency-based crime is growing rapidly due to the lack of identity management, which has serious negative socioeconomic consequences.

This scheme establishes a permissioned environment in permissionless public blockchains by allowing users and VASPs to mutually prove that the owner of the wallet account is a verified identity so that the related crimes can be effectively solved. Requiring VASPs to prove to users that the identity associated with the wallet account has been verified can reduce the occurrence of crimes such as fraud and safeguard the rights and interests of investors. Requiring users to prove to VASPs that the owner of the wallet account is a verified identity can reduce crimes such as money laundering as the perpetrator must use a verified account to receive funds. This scheme not only helps prevent crime but also helps fight crime. For example, the scheme allows supervisors to track the identities of suspicious wallet accounts. By issuing a blacklist, supervisors can require VASPs to deny service to suspicious wallet accounts. Additionally, VASPs do not host KYC checks so they do not have to worry about malicious users passing censorship with fake documents and how to store users' identity data securely. As mentioned above, this scheme can help VASPs to improve their identity management to reduce cryptocurrency-based crimes so that VASPs can make a positive contribution to the social economy to achieve their sustainability goals.

Overly strict regulation will weaken the advantages of cryptocurrencies, especially privacy-preserving properties. In the regulatory frameworks proposed by the governments of Europe and South Korea, the governments require the sender of each transaction to provide the recipient with their personally identifiable information. In addition, the wallet account in the existing identity schemes is not anonymous to the identity provider. With the increasing use of cryptocurrencies, no one wants supervisors or third parties to know which transactions are associated with them or which crypto assets they own. A good regulatory scheme should protect the privacy of users. The scheme achieves regulation without compromising the anonymity of the blockchain, so it maintains users' data privacy rights.

The identity on the blockchain is simply an address generated from a private key and one can own multiple accounts by controlling multiple private keys. On-chain governance requires real participants, and the participation of wallet accounts is insufficient for refined governance, especially in the scenario of on-chain voting. "One address, one vote" cannot prevent cheating because address generation has almost no cost on the chain. Guaranteeing a unique identity on blockchains can bring positive results in DAO (decentralized autonomous organization) governance, the prevention of Sybil attacks, and airdrops. The scheme can determine that different accounts on a single blockchain or multiple blockchains correspond to one real-world identity, which can be used in these scenarios. This makes blockchain applications better applied to practical scenarios, such as charitable donations, etc., so these blockchain applications can contribute to socioeconomic development.

7. Concluding Remarks

This research proposes a KYC-compliant identity system. It solves the privacy issue of existing solutions and therefore does not affect the anonymity of the blockchain. The solution supports all public blockchains and only needs to be deployed on an EVM-compatible blockchain. It addresses the limitations of existing solutions that require extra work to support new blockchains. This scheme can help VASPs improve identity management so that VASPs can better contribute to socioeconomic development and achieve their sustainability goals. In the future, we will consider enriching the functions of attribute certificates, such as adding a reputation score based on on-chain and off-chain data.

Author Contributions: N.S. and Y.L. were the advisors. 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 the Postgraduate Research & 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:

KYC	know your customer
dapps	decentralized applications
IdP	identity provider
VASP	virtual asset service provider
AML	anti-money laundering
Defi	decentralized finance
FT	fungible token
NFT	non-fungible token
SPV	simplified payment verification
IPFS	Inter-Planetary File System
DAO	decentralized autonomous organization
W3C	World Wide Web Consortium
DID	decentralized identifier
SBT	soul-bound token
OTC	over-the-counter

References

1. The Chainalysis 2022 Crypto Crime Report. Available online: https://go.chainalysis.com/rs/503-FAP-074/images/Crypto-Crime-Report-2022.pdf (accessed on 7 September 2022).
2. Hu, M.; Lee, A.D.; Putniņš, T.J. Evading Capital Controls via Cryptocurrencies: Evidence from the Blockchain. 2021. Available online: https://ssrn.com/abstract=3956933 (accessed on 10 September 2022).
3. Li, Y.; Yang, G.; Susilo, W.; Yu, Y.; Au, M.H.; Liu, D. Traceable monero: Anonymous cryptocurrency with enhanced accountability. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 679–691. [CrossRef]
4. Peili, L.I.; Haixia, X.U.; Tianjun, M.A. An efficient identity tracing scheme for blockchain-based systems. *Inf. Sci.* **2021**, *561*, 130–140.
5. Ateniese, G.; Faonio, A.; Magri, B.; Medeiros, B.D. Certified bitcoins. In *International Conference on Applied Cryptography and Network Security*; Springer: Cham, Switzerland, 2014; pp. 80–96.
6. Ma, T.; Xu, H.; Li, P. Skyeye: A Traceable Scheme for Blockchain; Cryptology ePrint Archive. 2020. Available online: https://eprint.iacr.org/2020/034 (accessed on 10 September 2022).
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.
8. Narayanan, A.; Bonneau, J.; Felten, E.; Miller, A.; Goldfeder, S. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*; Princeton University Press: Princeton, NJ, USA, 2016; pp. 170–177.
9. Meiklejohn, S.; Pomarole, M.; Jordan, G.; Levchenko, K.; McCoy, D.; Voelker, G.M.; Savage, S. A fistful of bitcoins: Characterizing payments among men with no names. In Proceedings of the Conference on Internet Measurement Conference, Barcelona Spain, 23–25 October 2013; pp. 127–140.
10. Houben, R.; Snyers, A. *Cryptocurrencies and Blockchain: Legal Context and Implications for Financial Crime, Money Laundering and Tax Evasion*; Publications Office of the European Union: Luxembourg, 2018; pp. 53–56.
11. Monaco, J.V. Identifying bitcoin users by transaction behavior. In *Biometric and Surveillance Technology for Human and Activity Identification XII*; SPIE: Bellingham, WA, USA, 2015; pp. 25–39.
12. Aghaee, A. Identifying blockchain-based cryptocurrency accounts using investment portfolios. *arXiv* **2021**, arXiv:2110.04394.
13. Civic. Available online: https://www.civic.com/solutions/ (accessed on 24 September 2022).
14. Dunphy, P.; Petitcolas, F.A.P. A First Look at Identity Management Schemes on the Blockchain. *IEEE Secur. Priv.* **2018**, *16*, 20–29. [CrossRef]
15. Ethereum Identity Standards. Available online: https://docs.ethhub.io/built-on-ethereum/identity/ERC-EIP/ (accessed on 24 September 2022).
16. Origin Protocol. Available online: https://erc725.originprotocol.com/#/ (accessed on 24 September 2022).
17. Self-Sovereign Identity Report. Available online: https://en.wikipedia.org/wiki/Self-sovereign_identity (accessed on 24 September 2022).
18. W3C. Decentralized Identifiers (DIDs) v1.0. Available online: https://www.w3.org/TR/2021/PR-did-core-20210803/ (accessed on 7 November 2021).
19. DID-Methods. Available online: https://w3c.github.io/did-spec-registries/#did-methods (accessed on 24 September 2022).
20. Weyl, E.G.; Ohlhaver, P.; Buterin, V. Decentralized Society: Finding Web3's Soul. 2022. Available online: https://ssrn.com/abstract=4105763 (accessed on 15 September 2022).
21. Binance to Launch Binance Account Bound (BAB) Token, the First-Ever Soulbound Token on BNB Chain. Available online: https://www.binance.com/en/support/announcement/0fe1e7c8781844e29f56cb674231dfd7 (accessed on 21 August 2022).
22. Reputation DAO. Available online: https://twitter.com/reputationdao (accessed on 24 September 2022).

23. Project Galaxy. Available online: https://docs.galaxy.eco/ (accessed on 24 September 2022).
24. Sublime. Available online: https://sublime.finance/ (accessed on 24 September 2022).
25. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Bus. Rev.* **2008**, *21260*, 1–9.
26. Fülöp, M.T.; Gubán, M.; Gubán, Á.; Avornicului, M. Application Research of Soft Computing Based on Machine Learning Production Scheduling. *Processes* **2022**, *10*, 520. [CrossRef]
27. Buterin, V. A next-generation smart contract and decentralized application platform. *White Pap.* **2014**, *3*, 1–36.
28. Szabo, N. Formalizing and securing relationships on public networks. *First Monday* **1997**, *2*, 1. [CrossRef]
29. Fabian, V.; Buterin, V. EIP-20: Token Standard. Available online: https://eips.ethereum.org/EIPS/eip-20 (accessed on 20 August 2022).
30. Benet, J. Ipfs-Content Addressed, Versioned, p2p File System. *arXiv* **2014**, arXiv:1407.3561.
31. Method of Providing Digital Signatures. Available online: https://patentimages.storage.googleapis.com/69/ab/d9/2ff9f94fada6ea/US4309569.pdf (accessed on 5 November 2021).
32. Pedersen T P. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 129–140.
33. Camenisch, J.; Drijvers, M.; Lehmann, A. Anonymous attestation using the strong diffie hellman assumption revisited. In *Trust and Trustworthy Computing. Trust 2016*; Franz, M., Papadimitratos, P., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2016; Volume 9824. [CrossRef]
34. Damgård, I. *On Σ-Protocols*; Lecture Notes; University of Aarhus, Department for Computer Science: Aarhus, Denmark, 2002; p. 84.
35. Bernhard, D.; Pereira, O.; Warinschi, B. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 626–643.
36. BrightID. Available online: https://www.brightid.org/ (accessed on 19 October 2022).
37. Proof of Humanity. Available online: https://www.proofofhumanity.id/ (accessed on 19 October 2022).
38. POAP. Available online: https://poap.xyz/ (accessed on 19 October 2022).
39. Ethereum Name Service. Available online: https://ens.domains/ (accessed on 19 October 2022).
40. Naik, N.; Jenkins, P. Governing principles of self-sovereign identity applied to blockchain enabled privacy preserving identity management systems. In Proceedings of the IEEE International Symposium on Systems Engineering (ISSE), Vienna, Austria, 12 October–12 November 2020; pp. 1–6.