PBIDM: Privacy-Preserving Blockchain-Based Identity Management System for Industrial Internet of Things

Zijian Bao, Debiao He, *Member, IEEE*, Muhammad Khurram Khan, *Senior Member, IEEE*, Min Luo, and Qi Xie

***Abstract*—Industrial Internet of Things (IIoT) is revolutionizing plenty of industrial applications by utilizing large-scale smart devices in manufacturing and industrial processes. However, IIoT is facing the disclosure of identity privacy. The identity information is precious and critical, thereby inspiring a line of follow-up privacy-preserving studies, i.e., anonymous credential protocols, or privacy-preserving identity management schemes. However, they are either too anonymous to be used in the IIoT environment, or the system is highly centralized, which implies the risk of a single point of failure. In this article, we propose PBIDM, a privacy-preserving blockchain-based identity management scheme for IIoT. Specifically, by leveraging blockchain and diversified cryptographic tools, PBIDM can fully support the desirable properties, i.e., unforgeability, blindness, unlikability, traceability, revocability, and public verifiability. Then, we provide security analysis to ensure reasonable security assurance. Finally, we present a performance evaluation of the proposed scheme to demonstrate the practicability in IIoT applications.**

***Index Terms*—Anonymous credentials, blockchain, blockchain-based identity, industrial Internet of Things, PBIDM.**

Manuscript received 26 December 2021; revised 25 March 2022 and 10 June 2022; accepted 11 September 2022. Date of publication 15 September 2022; date of current version 13 December 2022. The work was supported in part by the National Key Research and Development Program of China (2021YFA1000600), in part by the National Natural Science Foundation of China under Grant 62172307, Grant U21A20466, Grant 61972294, Grant 61932016, in part by the Special Project on Science and Technology Program of Hubei Province 2020AEA013, in part by the Natural Science Foundation of Hubei Province 2020CFA052, and in part by the Wuhan Municipal Science and Technology Project 2020010601012187. The work of Muhammad Khurram Khan was supported by King Saud University, Riyadh, Saudi Arabia, under project number RSP-2022/12. Paper no. TII-21-5747. *(Corresponding author: Debiao He.)*

Zijian Bao is with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Shandong Provincial Key Laboratory of Computer Networks, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China (e-mail: baozijian@whu.edu.cn).

Debiao He is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Shanghai Key Laboratory of Privacy-Preserving Computation, MatrixElements Technologies, Shanghai 201204, China (e-mail: hedebiao@163.com).

Muhammad Khurram Khan is with the Center of Excellence in Information Assurance, King Saud University, Riyadh 11653, Saudi Arabia (e-mail: mkhurram@ksu.edu.sa).

Min Luo is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: mluo@whu.edu.cn).

Qi Xie is with the Key Laboratory of Cryptography of Zhejiang Province, Hangzhou Normal University, Hangzhou 311121, China (e-mail: qixie68@126.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3206798.

Digital Object Identifier 10.1109/TII.2022.3206798

I. Introduction

WITH the emergence of the fourth industrial revolution, the gradual maturity of Internet of Things technology and Intelligent manufacturing in an industrial setting give birth to the concept of industrial Internet of Things (IIoT) [1]. IIoT can help manufacturers in real-time data analysis to maintain agile and efficient information utilization, thereby achieving a more intelligent workflow.

Existing IIoT applications improve our lives, but they also bring us privacy threats. For instance, on the Internet of Vehicles (IoV), smart vehicles enjoy various intelligent services provided by service providers, such as digital maps, navigation services, and location services. But these providers (e.g., Google map and Baidu map) can easily learn the privacy information of the users, e.g., location and travel information. Once this information is leaked, it may cause irreparable consequences. In fact, similar privacy disclosure problems are common in IIoT.

One of the promising solutions is using *anonymous credentials*. Anonymous credentials scheme was proposed by Chaum [2], which allows users to prove possession of credentials without revealing any other information about themselves. For example, in the aforementioned case, smart vehicles first obtain anonymous credentials from the vehicle management organization. They can use their anonymous credentials to request services from the service providers without disclosing their identities.

However, anonymity brings several new challenges, i.e., the users' identity cannot be traced and revoked. As far as we know, *traceability* and *revocability* play an important role in the current IIoT identity management systems. Once a malicious user cannot be supervised, it will cause a huge negative impact on the credibility of the whole system. Yu et al. [3] designed a blockchain-based anonymous authentication with selective revocation for smart industrial applications and formalized the system components by using PS signature [4] and dynamic

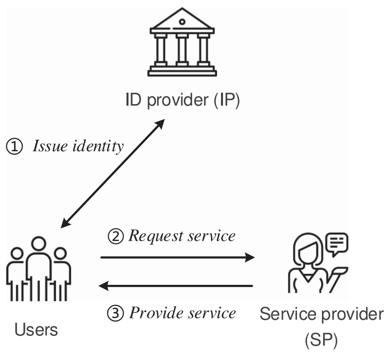


Fig. 1. Simple user-IP-SP model in IIoT.

accumulators. While it only supports *revocability*, *traceability* is still missing. Thus, we ask the following question "*Can we propose a privacy-preserving identity management system to support traceability and revocability for IIoT simultaneously?*"

To better solve this question, we first define a common and general model in the IIoT ecosystem as shown in Fig. 1. It divides entities into *users*, *identity providers* (IP), and *service providers* (SP). The user obtains the legal identity along with his attributes from the IP by checking the users' official documents, i.e., ID cards, or passports. Then if the user's attributes meet the requirements, the user can obtain corresponding services from the SP. The previous IoV case also applies to this model. Our core design goal is to build a *privacy-preserving* identity management system with *traceability* and *revocability* in such a model.

We positively answer the above question and make the following contributions:

1) We present PBIDM, a **p**rivacy-preserving **b**lockchain-based **id**entity **m**anagement system for IIoT. By taking various cryptographic techniques, PBIDM can simultaneously achieve *unforgeability, blindness, unlikability, traceability, revocability, and public verifiability*.
2) The additional point is that PBIDM adopts the World Wide Web Consortium (W3C) standard[1] [5], therefore it can be easily integrated into many open-source identity management frameworks, such as ION [6], WeIdentity [7].
3) Furthermore, we give a detailed security analysis of our scheme, compare the properties of PBIDM with several other schemes and present the performance evaluation to demonstrate PBIDM's feasibility.

The rest of this article is organized as follows. In Sections II and III, we review the related work and preliminaries. Then, we present the problem formulation in Section IV. In Section V, we provide the detailed construction. In Sections VI and VII, we give the security analysis, comparison, and performance evaluation. Finally, Section VIII concludes this article.

II. Related Work

Anonymous credentials: Camenisch et al. [8] proposed CL signatures, which became a powerful tool for constructing anonymous credentials. The CL signature scheme can provide anonymity and attribute privacy through "random then prove." Based on it, Microsoft proposed a user-centric identity management system. Yet, the size of the credential increases linearly with the number of attributes. To overcome the above shortcoming, Pointcheval and Sanders [4] proposed a new type of signature scheme, i.e., Pointcheval-Sanders (PS) signature. Its signature only consists of only two elements, which is more efficient than CL signatures. Sonnino et al. [9] proposed Coconut, a threshold issuance selective disclosure credentials based on PS signature and gave the application scenarios in a decentralized ledger. Coconut, however, suffers from the intrinsic anonymity of PS signatures, thereby making it hard to support *traceability* and *revocability*. Sanders [10] designed a redactable signature scheme that the resulting redacted signature from PS signatures only consists of four elements, further supporting *unlinkability* in an anonymous credential application. Unfortunately, it still encounters the same dilemma with Coconut. Garman et al. [11] presented an anonymous credential scheme that eliminates the need for a trusted credential issuer by using the decentralized ledger, commitment, and zero-knowledge proof. It is the first attempt to use blockchain technology in identity systems. Yet, it is hard to utilize such a completely unorganized credential system, as users' attributes are declared by themselves. Yu et al. [3] first gave a blockchain-based anonymous authentication with selective revocation in an industrial environment. It seems that their scheme is close to the answer we want to achieve, yet, *traceability* remains missing.

Blockchain-based identity systems: The traditional centralized identity is generated and managed by several independent IPs, such as Paypal, Facebook, Twitter, and other giant companies in our daily life. This architecture is vulnerable to various attacks, such as a single point of failure, phishing attacks, malicious tampering attacks, and internal attacks. As a new distributed ledger, blockchain can alleviate the disadvantages and make decentralized identity possible. Decentralized identity (i.e., DID) has gradually become a research hotspot, providing users with a means to control personal information in the process of data sharing. The international standards organization W3C, takes the lead in formulating relevant standards for distributed digital identity [5], including the organization forms of key data such as distributed identifiers and verified credentials. W3C approved the "Verifiable credentials data model implementation report 1.0" in 2019.

Microsoft established the decentralized identity authentication infrastructure project ION [6] on the basis of Bitcoin to support large-scale decentralized public key infrastructure. Meanwhile, WeBank [7] implemented a set of decentralized identity protocols conforming to W3C did specification on the Fisco-BCOS blockchain [12], such as WeIdentity, which claims to support selective disclosure of attributes. Yet, as industrial products, these projects do not fully consider the users' privacy. Maram et al. [13] proposed a decentralized identity management system, fixing four major challenges: legacy compatibility, sybil-resistance, accountability, key recovery. However, it relies on an *Oracle* system for credential issuance and key recovery, thereby limiting its applications. Yang et al. [14] improved the

[1]An international standards organization, which takes the lead in formulating relevant standards for digital identities.

identity model based on blockchain by using smart contract and zero-knowledge proof, and built a system prototype based on challenge-response protocols, which allows users to selectively disclose their attributes to service providers, thereby protecting users' behavior privacy. Yet, *traceability* and *revocability* are not considered in their scheme. Wang et al. [15] designed a blockchain-based anonymous authentication and key agreement protocol for the edge-computing-based smart grid system. Yet, their scheme is not suitable for the user-IP-SP model. Damgard et al. [16] proposed a new design principle of blockchain identity management, maintaining privacy while still allowing compliance with existing regulations. Buts, their design goal is to build an identity system for blockchain, while ours is to use blockchain for identity management in IIoT.

III. Preliminaries

In this section, we use the following notation, mathematical assumptions, and cryptographic building blocks, i.e., BBS+ signature, zero-knowledge proof, cryptographic accumulator, distributed ElGamal encryption, blockchain, and smart contracts.

A. Notation

Definition 1 (Bilinear Map): Let $(\mathbb{G}, \mathbb{G}_T)$ be a bilinear map such that $\tilde{e} : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$, where p is the order for both $(\mathbb{G}, \mathbb{G}_T)$. The bilinear map should satisfy the following properties:

1) *Bilinear:* Given any two element $a, b \in Z_q^*$ and $\forall x, y \in \mathbb{G}$, $\tilde{e}(x^a, y^b) = \tilde{e}(x, y)^{ab}$.
2) *Nondegenerate:* There exists at least one element $x \in \mathbb{G}$ satisfies $\tilde{e}(x, x) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ represents the identity element in $\mathbb{G}_T$.
3) *Efficient computability:* For $\forall x, y \in \mathbb{G}$, $\tilde{e}(x, y)$ is efficiently computable.

B. Mathematical Assumptions

Definition 2 (Discrete Logarithm): The Discrete Logarithm (DL) problem is defined as follows: Given a tuple $(g, g^a) \in \mathbb{G}^2$, outputs a. We say the DL assumption holds in $\mathbb{G}$ if there is no probabilistic polynomial time ($\mathcal{PPT}$) algorithm that can solve it with nonnegligible advantage.

Definition 3 (Decisional Diffie–Hellman): The Decisional Diffie–Hellman (DDH) problem is defined as follows: Given a quadruple $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, determines $g^c = g^{ab}$ or not. We say that DDH assumption holds if there is no $\mathcal{PPT}$ algorithm that can solve it with nonnegligible advantage.

Definition 4 (q-Strong Diffie–Hellman): The q-Strong Diffie–Hellman (q-SDH) problem is defined as follows: Given a (q+2) tuple $(g, g_0, g_0^x, g_0^{x^2}, \ldots, g_0^{x^q}) \in \mathbb{G}^{q+2}$, outputs a pair (A, c) satisfies $A^{x+c} = g_0$, where $c \in_R \mathbb{Z}_q^*$. We say that q-SDH assumption holds if there is no $\mathcal{PPT}$ algorithm that can solve it with nonnegligible advantage.

C. BBS+ Signature

We briefly review the BBS+ signature [17], which is usually used in the anonymous authentication protocols. Let $g_0, g_1, g_2 \in \mathbb{G}^3$. The signer randomly chooses $x \in_R \mathbb{Z}_q^*$, lets $w = g_0^x$ be the public key. The signature of message m is (A, e, s), where $e, s \in_R \mathbb{Z}_q^*$ and $A = (g_0 g_1^s g_2^m)^{\frac{1}{e+x}}$. Then, we can check whether $e(A, w g_0^e) = e(g_0 g_1^s g_2^m, g_0)$. If `true`, the signature is `valid`; otherwise, `invalid`. BBS+ signature is unforgeable against adaptively chosen message attack under the q-SDH assumption.

D. Zero-Knowledge Proof

Zero-knowledge protocol enables a prover to convince a verifier that a statement is `true` without exposing anything except the validity of the statement. Σ-protocol [18] is a three-move interactive zero-knowledge proof of knowledge system (ZKPOK), which can be transformed to noninteractive zero-knowledge argument via *Fiat–Shamir heuristic* [19] and a secure hash function $\mathcal{H}$. For instance, we define $\mathcal{PK}\{(x) : y = g^x\}$ is a Σ-protocol of proving a discrete logarithm relation. Namely, the prover has the witness x and wants to convince the verifier that he knows that the witness satisfies $y = g^x$ without revealing x. Similarly, the noninteractive signature proof of knowledge for message m is denoted as $\mathcal{SPK}\{(x) : y = g^x\}(m)$.

E. Cryptographic Accumulator

Cryptographic accumulator allows `multiple values` to be aggregated into a `single value` with the ability to generate a witness for each value. For example, we can accumulate multiple value, *e.g.*, $\{k_1, k_2, \ldots, k_n\}$, to a short accumulated value δ. For each value $k_i, i \in [1, n]$, we obtain a witness w_i that can prove the value k_i is accumulated in δ. We use the accumulator protocol in [17], modified from [20]. The algorithm is proposed as follows.

1) **Acc.Setup**. On input the system parameter, outputs the private/public key pair (sk, pk), where $sk \in_R \mathbb{Z}_q^*$, $pk = g^{sk}$.
2) **Acc.Eval**. On input a set $\mathcal{K} = \{k_1, k_2, \ldots, k_n\}$ of n data values, outputs an accumulator value $\delta = g^{\prod_{i=1}^{n}(k_i + sk)}$.
3) **Acc.Wit**. On input a dataset $\mathcal{K}$ and a value k_i, outputs the corresponding witness $w = g^{\prod_{i=1, i\neq j}^{n}(k_i + sk)}$, such that $\delta = w^{(k_i + sk)}$.
4) **Acc.Verify**. On input the accumulator value δ and the witness w, If the verification is `valid`, outputs 1; otherwise, outputs 0.

F. Distributed ElGamal Encryption

ElGamal encryption was proposed by Taher Elgamal in 1985 [21]. If the DDH assumption holds, ElGamal is semantically secure. A distributed ElGamal encryption was proposed in [22], defined as follows.

1) **Setup**. Suppose the number of users to d. Each user generates its key pair (sk_i, pk_i), where $sk_i = x_i$, $pk_i = g^{x_i}, x_i \in_R \mathbb{Z}_q^*$. Then, they publish their public keys to each other, thereby computing the common public key $pk_{COM} = \prod_{d}^{i=1} pk_i$.
2) **Enc**. the encrypted ciphertext is (c_1, c_2) of m, where $c_1 = g^r, c_2 = pk_{COM}^r \cdot g^m, r \in_R \mathbb{Z}_q^*$.

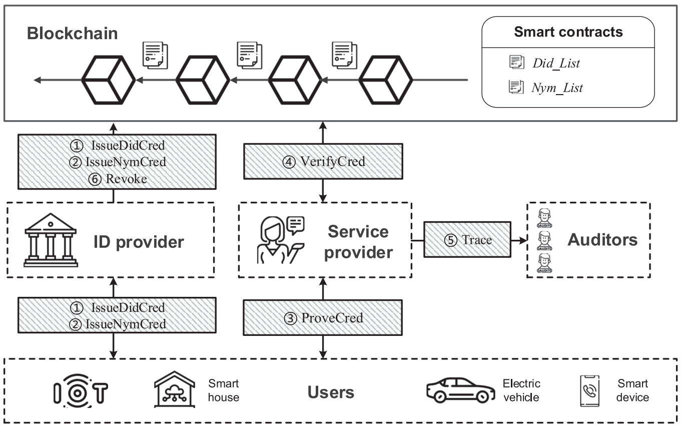
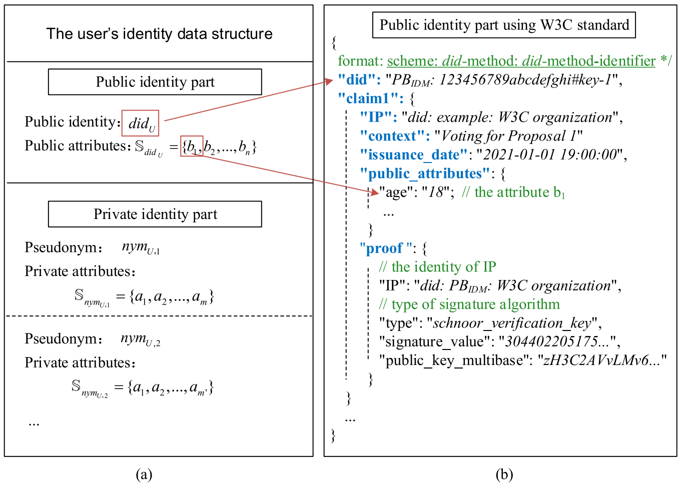


Fig. 2. Proposed system model.


The user's identity data structure
Public identity part
Public identity: did_U
Public attributes: $\mathbb{S}_{did_U} = \{b_1, b_2, ..., b_n\}$
Private identity part
Pseudonym: $nym_{U,1}$
Private attributes:
$\mathbb{S}_{nym_{U,1}} = \{a_1, a_2, ..., a_m\}$
Pseudonym: $nym_{U,2}$
Private attributes:
$\mathbb{S}_{nym_{U,2}} = \{a_1, a_2, ..., a_{m'}\}$
...
(a)
Public identity part using W3C standard
(b)

```
{
 format: scheme: did-method: did-method-identifier */
 "did": "PBIDM: 123456789abcdefghi#key-1",
 "claim1": {
    "IP": "did: example: W3C organization",
    "context": "Voting for Proposal 1"
    "issuance_date": "2021-01-01 19:00:00",
    "public_attributes": {
      "age": "18";  // the attribute b1
       ...
    }
    "proof ": {
      // the identity of IP
      "IP": "did: PBIDM: W3C organization",
      // type of signature algorithm
      "type": "schnoor_verification_key",
      "signature_value": "304402205175...",
      "public_key_multibase": "zH3C2AVvLMv6..."
    }
  }
  ...
}
```

Fig. 3. User's identity data structure.

3) **Dec**. Each user broadcasts their partial decryption key $c_1^{sk_i}$ to each others. Then, we can get $g^m = c_2 / \prod_{i=1}^{d} (c_1^{sk_i})$.

G. Blockchain and Smart Contracts

Blockchain technology supports a specific set of participants to share data. It can collect and share transaction data from multiple sources, subdivide the data into shared blocks linked together by unique identifiers in the form of hash. It also ensures data integrity, eliminates data duplication, and improves data security through a single information source.

Smart contracts maintained on a blockchain are critical programs that can be executed when certain conditions are met. They are usually used to automate the execution of an agreement so that all parties may be confident of the conclusion immediately, without the need for any intermediaries. A smart contract allows users to realize personalized code logic on the blockchain. Smart contract technology based on blockchain has the characteristics of decentralization, autonomy, observability, verifiability, and information sharing, which can effectively build programmable finance and society.

IV. Problem Formulation

A. System Model

As shown in Fig. 2, PBIDM involves five entities: blockchain, users, identity providers, service providers, and auditors.

1) **Blockchain:** We regard blockchain as an append-only ledger LEDGER. Hence, we ignore attacks against the blockchain itself (e.g., 51% attacks, sybil attacks). The usage of blockchain in PBIDM is to record several carefully designed contracts. Our design choice is to reduce the workload and avoid using complex cryptographic tools on the chain, e.g., zero-knowledge proof.
2) **Users:** Users, e.g., smart devices, smart houses, are entities who obtain identities along with attributes from the IP, then get the services provided by the SP. They perform legal certificates with the IP and complete registration on the blockchain.
3) **Identity providers:** By interacting with users, the IP confirms the user's legal identification (i.e., ID cards or passports) and signs the user's "identity." After confirmation on the blockchain, the user can request services from the SP later.
4) **Service providers:** The SP is responsible for providing online/offline services to users with required attributes included in its access policy, verifying the "identity" and attributes provided by users.
5) **Auditors:** Auditors are parties who are involved in cases when law enforcement or other authorized entities require the ability to extract the "identity" of a malicious user. In this article, we require all d auditors to fetch the user's "identity." Note that it is straightforward to extend our scheme into a threshold case, which requires at least t of d auditor involved in the identity tracing process.

B. User's Identity Data Structure

In PBIDM, we divide a user's "identity" into *public identity* credentials (i.e., did_U and public attributes set $\mathbb{S}_{did_U}$) and *private identity* credentials (i.e., nym_U and private attributes set $\mathbb{S}_{nym_U}$). For *public identity* credentials, all information is published on the blockchain. However, for *private identity* credentials, we only publish a nym_U to the blockchain, while keeping private attributes set $\mathbb{S}_{nym_U}$ locally. See Fig. 3(a) for the detailed data structure.

When using private attributes, the user proves the statement that he owns a legal pseudonym with required private attributes, while exposing nothing except the validity of the statement. Jumping ahead, when observing transactions on the blockchain, any adversary (Abbreviated as $\mathcal{A}$) cannot infer the user's nym and the corresponding private attributes.

We adopt the data structure of the W3C verifiable credentials specification's definition of a credential [13], similar to [5]. A credential is a set of claims made by an IP, which is detailed defined as below, see Fig. 3(b).

1) User identity: The identity of the user, i.e., did_U or nym_U.
2) Claims: The claim contains an IP, a context, a issuance date, a set of public attributes pairs, and a proof (a signature signed by IP over the did_U and claims). Context indicates the circumstances of credential usage, e.g., "*Voting for Proposal 1*."

C. Design Goals

We present the design goals of our identity system.

1) **Unforgeability:** A $\mathcal{PPT}$ adversary $\mathcal{A}$ without the legal authentication on public/private attributes cannot forge a credential to convince any honest verifier that it is `valid`, i.e., it cannot forge a credential to pass the verification (see the *VerifyCred* phase mentioned below).
2) **Blindness:** This property requires that the IP learns nothing about the user's private attributes except that the attributes satisfy the statement ϕ. Blindness implies the minimal disclosure of user credentials for issuing the credentials (see the *IssueNymCredyCred* phase mentioned below).
3) **Unlinkability:** Similar to the definition of blindness, unlinkability requires that the SP learn nothing about the user's private attributes except that the attributes satisfy the statement ϕ'. Unlinkability is defined for *ProveCred* phase mentioned below.
4) **Traceability:** Traceability means that we can finally track the user's "identity." Namely, a list of auditors can jointly recover the "identity" of a user, specifically, the user's nym_U or did_U.
5) **Revocability:** Revocability means that once we trace some malicious identities, we may revoke their credentials. Without revocability, some malicious users that have been discovered can continue to obtain SP services, which will bring disastrous damage to the system.
6) **Public Verifiability:** We hope that the proposed system can be publicly verifiable. That is, when the SP verifies the user's credentials, it does not need to interact with IP, but can do it through a publicly verifiable ledger at any time.

D. High Level Description

PBIDM is composed of a set of $\mathcal{PPT}$ algorithms, including {*Setup*, *KeyGen*, *IssueDidCred*, *IssueNymCred*, *ProveCred*, *VerifyCred*, *Trace*, *Revoke*}.

1) **Setup:** Given a security parameter n as an input, it outputs a set of system parameters $\mathbb{PP}$.
2) **KeyGen:** Given the system parameters $\mathbb{PP}$, it invokes three subalgorithms {*UserKeyGen, IPKeyGen, AuditorKeyGen*} to generate the corresponding public–private pairs for users, IPs, and auditors.
3) **IssueDidCred:** Given did_U, the public attributes $\mathbb{S}_{did_U} = \{b_1, b_2, \ldots, b_n\}$ and relevant certificates, the IP checks if it is `valid`, then uploads the public identity part to blockchain.
4) **IssueNymCred:** A user with public attributes $\mathbb{S}_{nym_U} = \{a_1, a_2, \ldots, a_m\}$ and relevant certificates, interacts with the IP to obtains a legal nym_U and the IP's signature of $\mathbb{S}_{nym_U}$, by using a zero-knowledge proof Ξ_1 that nym_U is linked to did_U and $\mathbb{S}_{nym_U}$ satisfies the statement ϕ. Then, IP uploads nym_U to the blockchain.
5) **ProveCred:** For a user's public identity part, a user sends a Schnorr signature of the service request Rq to the SP. For private identity part, a user computes a zero-knowledge proof Ξ_2 that a) he has a legal nym issued by the IP without exposing it; b) the provided private attributes satisfy the statement ϕ' required by the SP; c) the nym_U is encrypted by the auditors' public key. Note that proof a) can be implemented by using an accumulator and Σ-protocols.
6) **VerifyCred:** For public identity part, the IP verifies whether the Schnorr signature is `valid`; For private identity part, the IP verifies whether Ξ_2 is `valid`. Then, the SP provides services accordingly.
7) **Trace:** It is an interactive protocol between the SP and the auditors, then the auditors can find out the malicious user's g^{nym_U}.
8) **Revoke:** The IP can revoke a user's did_U or nym_U by invoking smart contracts.

Algorithm 1: `did_Initialzation()`.

```
1: Struct DID {
2:     $did$;
3:     $claim\{\ldots\}$;
4: }
5: DID[] public DID_List;
6: Constructor DID_List() {
7:     $owner = msg.sender$;
8:     $return$
9: }
```

V. Construction

In this section, we first present the smart contracts used in PBIDM. Then, we give the detailed construction for multiattributes. Note that PBIDM is easy to deploy in a one-attribute scenario.

A. Smart Contracts in PBIDM

In PBIDM, one of our design goals is to minimize the storage and computation costs on the blockchain. We deploy several simple yet effective smart contracts to support a set of algorithms, namely, `did_Initialzation()`, `nym_Initialzation()`, `add_Did_List()`, `revoke_Did_List()`, and `update_Nym_List()` in Algorithms 1–5. Note that Algorithms 1 and 2 are the concrete pseudocode of the public identity part and the private identity part in Fig. 3.

B. Detailed Construction

1) **Setup:** Given a security parameter n as an input, it generates the bilinear group pair $(\mathbb{G}, \mathbb{G}_T)$ with order q. The IP chooses generators $g, g', g_0, g_1, g_2, h, h_0, \rho_1, \ldots, \rho_m \in_R$

Algorithm 2: `nym_Initialzation()`.

```
1: Struct NYM {
2:     $nym$;
3:     $tag$; // $tag \in \{0, 1\}$ denotes the $nym$ is authorized/revoked
4:     $Acc$; //$Acc$ is the initialized accumulated value
5: }
6: NYM[] public NYM_List;
7: Constructor NYM_List($\delta$) {
8:     $owner = msg.sender$;
9:     $Acc = \delta$; // $\delta = h_0$, see KeyGen phase
10:    $return$ 1;
11: }
```

Algorithm 3: `add_Did_List` ($did, claim$).

```
1 if $onwer$ != $msg.sender$ then
2     $return$ 0;
3 else
4     $DID\_List[i].did = did$;
5     $DID\_List[i].claim = claim$;
6 end
```

Algorithm 4: `revoke_Did_List` (did).

```
1 if $onwer$ != $msg.sender$ then
2     $return$ 0;
3 else
4     $i = 0$;
5     while $DID\_List[i].did$ != $did$ do
6         $i$++;
7     end
8     $delete$ $DID\_List[i]$;
9 end
```

Algorithm 5: `update_Nym_List` (nym, tag, Acc).

```
1 if $onwer$ != $msg.sender$ then
2     $return$ 0;
3 else
4     $NYM\_List[i].nym = nym$;
5     $NYM\_List[i].tag = tag$;
6     $NYM\_List[i].Acc = Acc$;
7 end
```

$\mathbb{G}$, where m is the number of a user's private attributes, hash function $\mathcal{H} : \{0, 1\}^* \to \mathbb{Z}_q$. The main parameters in PBIDM are listed in Table I.

2) **KeyGen:**

a) UserKeyGen: A user generates its key pair (sk_U, pk_U), where $sk_U \in_R \mathbb{Z}_q^*$, $pk_U = h^{sk_U}$. We define pk_U as the user's public identity did_U.

b) IPKeyGen: The IP generates its key pair (sk_{IP}, pk_{IP}), where $sk_{IP} \in_R \mathbb{Z}_q^*, pk_{IP} = g'^{sk_{IP}}$ and initializes the accumulator as follows. Each IP random chooses $p \in_R \mathbb{Z}_q^*$ as a private key sk_{Acc}, sets the public key as $pk_{Acc} = h_0^p$, sets $\delta = h_0$, where δ is the initialized accumulated value.

TABLE I
PARAMETERS

Symbol	Definition
n	Security parameter
$\mathbb{G}$	Additive cyclic groups of order q
$\mathbb{G}_T$	Multiplicative cyclic group of order q
$\tilde{e}$	Bilinear map
$g, g', g_0, g_1, g_2,$ $h, h_0, \rho_1, ..., \rho_m$	Generators of $\mathbb{G}$
$\mathcal{H}$	Hash function: $\{0,1\}^* \to \mathbb{Z}_q$
(sk_U, pk_U)	The user's key pair
(sk_{IP}, pk_{IP})	The IP's key pair
(sk_i, pk_i)	The i_{th} auditor's key pair
did_U	The user's public identity
$\mathbb{S}_{did_U}$	The user's public attributes set
nym_U	The user's private identity
$\mathbb{S}_{nym_U}$	The user's private attributes set
$b_1, b_2, ..., b_n$	Public attributes
$a_1, ..., a_m$	Private attributes
(A, e, s)	BBS + signature value
d	The number of auditors
Acc_{new}, Acc_{old}	The accumulated value
sk_{Acc}, pk_{Acc}	The accumulator's key pair
w_U	The accumulator's witness
σ	Schnorr signature value
Ξ_1, Ξ_2	Non-interactive zero-knowledge proofs
Rq_1, Rq_2	Service requests
ϕ, ϕ'	The statement of users' attributes

c) AuditorKeyGen: We set the number of auditors to d. Each auditor generates its key pair (sk_i, pk_i), where $sk_i := x_i, pk_i := g^{x_i}, x_i \in_R \mathbb{Z}_q^*$. Then, they publish their public keys to each other, thereby computing the common public key $pk_{COM} = \prod_d^{i=1} pk_i$.

3) **IssueDidCred:** The public identity part is issued, through the following steps.

a) A user sends did_U, public attributes $\mathbb{S}_{did_U} = \{b_1, b_2, \ldots, b_n\}$ (for $i \in [1, n], b_i \in \mathbb{Z}_p^*$) and relevant certificates (i.e., ID card, passport) to the IP, where n is the number of a user's private attributes.

b) If the IP passes the verification, then the IP will send the public identity credentials to the blockchain by invoking `add_Did_List` ($did_U, claim$); if not, the IP rejects it. Note that all information is public and immutable, anyone can check its correctness.

4) **IssueNymCred:** A user gets the pseudonym nym_U and the private attributes $\mathbb{S}_{nym_U} = \{a_1, \ldots, a_m\}$ (for $i \in [1, m], a_i \in \mathbb{Z}_p^*$) through the following steps.

a) A user randomly chooses $s' \in_R \mathbb{Z}_p^*$, then computes $C = g_1^{s'} g_2^{sk_U}$. Then, the user sends C to the IP, along with the following noninteractive zero-knowledge proof $\Xi_1 = \mathcal{PK}\{(s', sk_U, a_1, a_2, \ldots, a_m) : C = g_1^{s'} g_2^{sk_U} \rho_1^{a_1} \rho_2^{a_2} \ldots \rho_m^{a_m} \wedge did_U = g^{sk_U} \wedge \phi(a_1, a_2, \ldots, a_m) = 1\}$.

b) The IP returns $\perp$, if Ξ_1 is not valid. Otherwise, the IP randomly chooses $s'' \in_R \mathbb{Z}_p^*$, computes $A = (g_0 C g_1^{s''})^{\frac{1}{e+sk_{IP}}}$, then sends (A, e, s'') to the user. Let e be the nym_U of a user.

c) The user computes $s = s' + s''$ and checks whether $\tilde{e}(A, g'^e \cdot pk_{IP}) = \tilde{e}(g_0 g_1^s g_2^{sk_U} \rho_1^{a_1} \rho_2^{a_2} \ldots \rho_m^{a_m}, g')$. The user then stores $(A, e, s, a_1, \ldots, a_m)$ locally.

d) The IP stores the entry $(did_U, nym_U, g^{nym_U})$ in the list Ψ locally, gets the latest Acc from the blockchain (denoted as Acc_{old}), computes a new accumulated value $Acc_{new} = Acc_{old}{}^{nym_U + p}$. The user keeps Acc_{old} as the witness w_U such that $Acc_{new} = w_U^{nym_U + p}$. The IP then sends the tuple $\{nym_U, 1, Acc_{new}\}$ to the blockchain by invoking `update_Nym_List(`$nym_U, 1, Acc_{new}$`)`.

e) When seeing the `Nym_List` changes on the blockchain, each user, for example Z, with his witness w_Z such that $w_Z^{nym_Z + p} = Acc_{old}$, updates his new witness $w_{Z'} = Acc_{old} \cdot w_Z^{nym_U - nym_Z}$. Thus, the user Z keeps $w_{Z'}$ as the new witness such that $w_{Z'}{}^{nym_Z + p} = Acc_{new}$.

5) **ProveCred:** We divide this algorithm into two parts as follows. Note that using public and private attributes depends on the application scenario, and we may use one or both.

a) Prove public attributes.

i) The user first sends to the SP the service request Rq_1, did_U, the public attributes $\mathbb{S}_1$ and the signature $\sigma = (k, s)$, where $k = \mathcal{H}(did_U || h^r || Rq_1)$, $s = r + k \cdot sk_U$, $r \in_R \mathbb{Z}_q$.

b) Prove private attributes.

i) The user sends the service request Rq_2 to the SP, an ElGamal ciphertext $m = (m_1, m_2)$ where $m_1 = g^{r'}, m_2 = pk_{COM}{}^{r'} \cdot g^e, r' \in_R \mathbb{Z}_q^*$, along with a zero-knowledge proof $\Xi_2 = \mathcal{SPK}\{(A, e, w_U, s, sk_U, r, a_1, a_2, \ldots, a_m) : A^{e + sk_{IP}} = g_0 g_1^s g_2^{sk_U} \rho_1^{a_1} \rho_2^{a_2} \ldots \rho_m^{a_m} \wedge m_1 = g^r \wedge m_2 = pk_{COM}{}^r \cdot g^e \wedge w_U{}^{e+p} = Acc \wedge \phi'(a_1, a_2, \ldots, a_m) = 1\}(Rq_2)$. Let $A_1 = g_1{}^{r_1} g_2{}^{r_2} g_3{}^{r_3}$, $A_2 = A g_2{}^{r_1}, A_3 = w_U g_3{}^{r_2}, \eta_1 = r_1 e, \eta_2 = r_2 e, \eta_3 = r_3 e$. It can be converted to a standard Σ-protocol $\Xi_2 = \mathcal{SPK}\{(e, s, sk_U, r, r_1, r_2, r_3, \eta_1, \eta_2, \eta_3, a_1, a_2, \ldots, a_m) : A_1 = g_1{}^{r_1} g_2{}^{r_2} g_3{}^{r_3} \wedge A_1{}^e = g_1{}^{\eta_1} g_2{}^{\eta_2} g_3{}^{\eta_3} \wedge \frac{\tilde{e}(A_2, pk_{IP})}{\tilde{e}(g_0, g')} = \tilde{e}(A_2, g')^{-e} \tilde{e}(g_2, pk_{IP})^{r_1} \tilde{e}(g_2, g')^{\eta_1} \tilde{e}(g_1, g')^s \tilde{e}(g_2, g')^{sk_U} \tilde{e}(\rho_1, g')^{a_1} \tilde{e}(\rho_2, g')^{a_2} \ldots \tilde{e}(\rho_m, g')^{a_m} \wedge m_1 = g^r \wedge m_2 = pk_{COM}{}^r \cdot g^e \wedge \frac{\tilde{e}(A_3, pk_{Acc})}{\tilde{e}(Acc, h_0)} = \tilde{e}(g_3, h_0)^{\eta_2} \tilde{e}(g_3, pk_{Acc})^{r_2} \tilde{e}(A_3, h_0)^{-e} \wedge \phi'(a_1, a_2, \ldots, a_m) = 1\}(Rq_2)$

6) **VerifyCred:** We divide this algorithm into two parts as follows.

a) Verify public attributes.

i) The SP checks whether the signature σ is `valid` by checking $k = \mathcal{H}(did_U || h^s \cdot did_U{}^{-k} || Rq_1)$. The IP also checks whether the public attributes $\mathbb{S}_1$ are consistent with the information stored on the blockchain and meet the requirements. If `valid`, the SP provides the corresponding services.

b) Verify private attributes.

i) The SP verifies if Ξ_2 is `valid`. If `valid`, the SP provides the corresponding services.

7) **Trace:**

a) For the private identity part, it is directly for IPs to obtain the did_U.

b) For the private identity part, the SP sends the tracing request $Re = \{m, tracing\}$ to auditors, then each auditor broadcasts their partial decryption key $m_1^{sk_i}$ and then recover $g^{nym_U} = c_2 / \prod_{i=1}^{d} (c_1^{sk_i})$. They then send g^{nym_U} to the IP, the IP searches the local list Ψ to get $\{did_U, nym_U, g^{nym_U}\}$.

8) **Revoke:**

a) For the public identity part, the IP invokes `revoke_Did_List(`did_U`)` on the blockchain.

b) For the private identity part, the IP gets the latest Acc from the blockchain (denoted as Acc_{old}), computes a new accumulated value $Acc_{new} = Acc_{old}{}^{\frac{1}{nym_U + p}}$ and invokes `update_Nym_List(`$nym_U, 0, Acc_{new}$`)` on the blockchain. Once a revocation operation is posted on the blockchain, each user, for example Z, with his witness w_Z such that $w_Z^{nym_Z + p} = Acc_{old}$, updates his new witness $w_{Z''} = (w_Z / Acc_{old})^{\frac{1}{nym_U - nym_Z}}$. Thus, the user Z keeps $w_{Z''}$ as the new witness such that $w_{Z''}{}^{nym_Z + p} = Acc_{new}$.

C. Details of Ξ_1 and Ξ_2

1) $\Xi_1 = \mathcal{PK}\{(s', sk_U, a_1, a_2, \ldots, a_m) : C = g_1^{s'} g_2^{sk_U} \rho_1^{a_1} \rho_2^{a_2} \ldots \rho_m^{a_m} \wedge did_U = g^{sk_U} \wedge \phi(a_1, a_2, \ldots, a_m) = 1\}$. We ignore the ϕ part. The specific process is as follows:

a) The IP sends a challenge $R \in_R \mathbb{Z}_p^*$ to the user.

b) The user chooses $r_{s'}, r_{sk_U}, r_{a_1}, r_{a_2}, \ldots, r_{a_m} \in_R \mathbb{Z}_p^*$.

c) The user computes $T_1 = g_1^{r_{s'}} g_2^{r_{sk_U}} \rho_1^{r_{a_1}} \rho_2^{r_{a_2}} \ldots \rho_m^{r_{a_m}}, T_2 = g^{r_{sk}}$.

d) The user computes $c = \mathcal{H}(T_1 || T_2 || R)$.

e) The user computes $z_{s'} = r_{s'} - c \cdot s', z_{sk_U} = r_{sk_U} - c \cdot sk_U, z_{a_1} = r_{a_1} - c \cdot a_1, z_{a_2} = r_{a_2} - c \cdot a_2, \ldots, z_{a_m} = r_{a_m} - c \cdot a_m$ and sends $(c, z_{s'}, z_{sk_U}, z_{a_1}, z_{a_2}, \ldots, z_{a_m})$ to the IP.

f) The IP outputs `accept` if $c = \mathcal{H}(C^c g_1^{z_{s'}} g_2^{z_{sk_U}} \rho_1^{z_{a_1}} \rho_2^{z_{a_2}} \ldots \rho_m^{z_{a_m}} || did_U{}^c g^{z_{sk_U}} || R)$ holds.

2) $\Xi_2 = \mathcal{SPK}\{(e, s, sk_U, r, r_1, r_2, r_3, \eta_1, \eta_2, \eta_3, a_1, a_2, \ldots, a_m) : A_1 = g_1{}^{r_1} g_2{}^{r_2} g_3{}^{r_3} \wedge A_1{}^e = g_1{}^{\eta_1} g_2{}^{\eta_2} g_3{}^{\eta_3} \wedge \frac{\tilde{e}(A_2, pk_{IP})}{\tilde{e}(g_0, g')} = \tilde{e}(A_2, g')^{-e} \tilde{e}(g_2, pk_{IP})^{r_1} \tilde{e}(g_2, g')^{\eta_1} \tilde{e}(g_1, g')^s \tilde{e}(g_2, g')^{sk_U} \tilde{e}(\rho_1, g')^{a_1} \tilde{e}(\rho_2, g')^{a_2} \ldots \tilde{e}(\rho_m, g')^{a_m} \wedge m_1 = g^r \wedge m_2 = pk_{COM}{}^r \cdot g^e \wedge \frac{\tilde{e}(A_3, pk_{Acc})}{\tilde{e}(Acc, h_0)} = \tilde{e}(g_3, h_0)^{\eta_2} \tilde{e}(g_3, pk_{Acc})^{r_2} \tilde{e}(A_3, h_0)^{-e} \wedge \phi'(a_1, a_2, \ldots, a_m) = 1\}(Rq_2)$, where $A_1 = g_1{}^{r_1} g_2{}^{r_2} g_3{}^{r_3}$, $A_2 = A g_2{}^{r_1}, A_3 = w_U g_3{}^{r_2}, \eta_1 = r_1 e, \eta_2 = r_2 e, \eta_3 = r_3 e$. We set $E_0 = \tilde{e}(g_0, g')$, $E_1 = \tilde{e}(g_2, pk_{IP})$,

$E_2 = \tilde{e}(g_2, g')$, $E_3 = \tilde{e}(g_1, g')$, $F_0 = \tilde{e}(g_3, h_0)$, $F_1 = \tilde{e}(g_3, pk_{Acc})$, $Z_1 = \tilde{e}(\rho_1, g')$, $Z_2 = \tilde{e}(\rho_2, g')$,..., $Z_m = \tilde{e}(\rho_m, g')$ as the system parameters to reduce computation cost. We ignore the ϕ' part. The specific process is as follows:

a) The user chooses $x_e, x_s, x_{sk_U}, x_r, x_{r_1}, x_{r_2}, x_{r_3}, x_{\eta_1}, x_{\eta_2}, x_{\eta_3}, x_{a_1}, x_{a_2},..., x_{a_m} \in_R \mathbb{Z}_p^*$.

b) The user computes $T_1 = {g_1}^{x_{r_1}} {g_2}^{x_{r_2}} {g_3}^{x_{r_3}}$, $T_2 = {A_1}^{-x_e} {g_1}^{x_{\eta_1}} {g_2}^{x_{\eta_2}} {g_3}^{x_{\eta_3}}$, $T_3 = \tilde{e}(A_2, g')^{-x_e} {E_1}^{x_{r_1}} {E_2}^{x_{\eta_1}} {E_3}^{x_s} {E_2}^{x_{sk_U}} {Z_1}^{x_{a_1}} {Z_2}^{x_{a_2}} \ldots {Z_m}^{x_{a_m}}$, $T_4 = g^{x_r}$, $T_5 = {pk_{COM}}^{x_r} \cdot g^{x_e}$, $T_6 = {F_0}^{x_{\eta_2}} {F_1}^{x_{r_2}} \tilde{e}(A_3, h_0)^{-x_e}$.

c) The user computes $c = \mathcal{H}(T_1||T_2||T_3||T_4||T_5||T_6||Rq_2)$.

d) The user computes $z_e = x_e - c \cdot e$, $z_s = x_s - c \cdot s$, $z_{sk_U} = x_{sk_U} - c \cdot sk_U$, $z_r = x_r - c \cdot r$, $z_{r_1} = x_{r_1} - c \cdot r_1$, $z_{r_2} = x_{r_2} - c \cdot r_2$, $z_{r_3} = x_{r_3} - c \cdot r_3$, $z_{\eta_1} = x_{\eta_1} - c \cdot \eta_1$, $z_{\eta_2} = x_{\eta_2} - c \cdot \eta_2$, $z_{\eta_3} = x_{\eta_3} - c \cdot \eta_3$, $z_{a_1} = x_{a_1} - c \cdot a_1$, $z_{a_2} = x_{a_2} - c \cdot a_2$,..., $z_{a_m} = x_{a_m} - c \cdot a_m$ and sends $(c, A_1, A_2, A_3, z_e, z_s, z_{sk_U}, z_r, z_{r_1}, z_{r_2}, z_{r_3}, z_{\eta_1}, z_{\eta_2}, z_{\eta_3}, z_{a_1}, z_{a_2},..., z_{a_m})$ to the SP.

e) The SP outputs `accept` if $c = \mathcal{H}({A_1}^c {g_1}^{z_{r_1}} {g_2}^{z_{r_2}} {g_3}^{z_{r_3}} || {A_1}^{-z_e} {g_1}^{z_{\eta_1}} {g_2}^{z_{\eta_2}} {g_3}^{z_{\eta_3}} || \left(\frac{\tilde{e}(A_2, pk_{IP})}{E_0}\right)^c \tilde{e}(A_2, g')^{-z_e} {E_1}^{z_{r_1}} {E_2}^{z_{\eta_1}} {E_3}^{z_s} {E_2}^{z_{sk_U}} {Z_1}^{z_{a_1}} {Z_2}^{z_{a_2}} \ldots {Z_m}^{z_{a_m}} || {m_1}^c g^{z_r} || {m_2}^c {pk_{COM}}^{z_r} g^{z_e} || \left(\frac{\tilde{e}(A_3, pk_{Acc})}{\tilde{e}(Acc, h_0)}\right)^c {F_0}^{x_{\eta_2}} {F_1}^{z_{r_2}} \tilde{e}(A_3, h_0)^{-z_e} || Rq_2)$ holds.

VI. Security Analysis and Comparison

PBIDM achieves our design goals including *unforgeability, blindness, unlinkability, traceability, revocability, public verifiability*. In general, similar to the security proof in [3], we reduce these properties to the security of *BBS+ signature*, *zero-knowledge proof*, *accumulator*, *distributed ElGamal encryption* and the security assumption of *blockchain*.

Lemma 1: PBIDM is unforgeable if *BBS+ signature* and the *accumulator* (see Section III-E) are secure under the q-SDH assumption an *Schnorr signature* [23] is secure in the random oracle model if the DL assumption holds.

Sketch: For public identity credentials, if an $\mathcal{A}$ can successfully forges a legal credential, it means it can forge a legal Schnorr signature, which is contradictory with the DL assumption. For private identity credentials, an $\mathcal{A}$ tries to forge a legal credential, there are two cases: 1) the user forges a BBS+ signature of the IP, which is contradictory with q-SDH assumption; 2) the user forges a non-existent nym, which can be accumulated in Acc. However, this is contradictory with the *collusion resistance* properties of an accumulator.

Lemma 2: PBIDM is blind if Ξ_1 is a ZKPOK protocol satisfying *zero-knowledge*.

Sketch: In *IssueNymCred* phase, the user sends the zero-knowledge proof $\Xi_1 = \mathcal{PK}\{(s,' sk_U, a_1, a_2, \ldots, a_m) : C = g_1^{s'} g_2^{sk_U} \rho_1^{a_1} \rho_2^{a_2} \ldots \rho_m^{a_m} \wedge did_U = g^{sk_U} \wedge \phi(a_1, a_2, \ldots, a_m) = 1\}$ of his private attributes $\mathbb{S}_{nym_U} = \{a_1, \ldots, a_m\}$. Due to Σ-protocol's properties, the zero-knowledge property of Ξ_1 ensures that commitment does not reveal information about the attribute set $\mathbb{S}_{nym_U}$. Then, the user interacts with the IP to obtain the final BBS+ signature (A, e, s).

Lemma 3: PBIDM is unlinkable if Ξ_2 is a noninteractive signature proof of knowledge protocol satisfying *zero-knowledge*.

Sketch: In *ProveCred* phase, the user sends the zero-knowledge proof $\Xi_2 = \mathcal{SPK}\{(A, e, w_U, s, sk_U, r, a_1, a_2, \ldots, a_m) : A^{e+sk_{IP}} = g_0 g_1^s g_2^{sk_U} \rho_1^{a_1} \rho_2^{a_2} \ldots \rho_m^{a_m} \wedge m_1 = g^r \wedge m_2 = {pk_{COM}}^r \cdot g^e \wedge {w_U}^{e+p} = Acc \wedge \phi'(a_1, a_2, \ldots, a_m) = 1\}(Rq_2)$ to the SP. This ensures that all privacy attributes are protected. At the same time, users are guaranteed to have a legal nym_U (note that $e = nym_U$), but the specific value of the nym_U is not disclosed.

Lemma 4: PBIDM is traceable and revocable if Ξ_2 is a noninteractive signature proof of knowledge protocol satisfying *soundness* and blockchain is *publicly accessible*.

Sketch: In *Trace* phase, for public identity part, it is easy to obtain did_U; for private identity part, as the Ξ_2 is sound, then we know that c is indeed the ciphertext of nym_U, and the auditors can decrypt it correctly. Then, IP can revoke did_U or nym_U as blockchain is *publicly accessible*.

Lemma 5: PBIDM achieves public verifiability if blockchain is *publicly verifiable*.

Sketch: Thanks to the *hash chain* structure and the decentralized P2P network of blockchain, without considering attacks against the blockchain itself, blockchain is *publicly verifiable*. And in PBIDM, we upload the public identity part and nym_U, Acc on the blockchain, anyone can easily obtain them and verify that the data on and off the chain is consistent.

Here, we give a comparison of recent works related to PBIDM in Table II. As for unforgeability, this is owned by all schemes, so we ignore it in the Table II. Yu et al.'s scheme [3] provides *traceability*, while *revocability* is missing. PBIDM can support W3C standard without any trusted hardware and support both the *traceability* and *revocability*.

VII. Performance

We analyze the performance of our proposed implementation in this section and compare it with Yu et al.'s scheme [3]. we use a bilinear pairings $\tilde{e} : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ with the order q to achieve 80 bits security level. The size of parameter q is 160 bits. To compare the computation of PBIDM, we compute the needed cryptographic operations on a personal computer (Dell with an i5-10400 processor, 8 G bytes RAM and Windows 10 operating system) using `MIRACLE` [24], which is a cryptography Library. The execution time of each operation is given in Table III.

We analyze the efficiency of PBIDM by counting different operations, i.e., T_{bp}, T_{mul}, T_{add}, T_h. Note that we use the single attribute application scenario, ignore the zero-knowledge proof ϕ, ϕ', set the number of auditors d to 3, and precompute several bilinear maps (e.g., E_0, E_1 in Section V-C) to reduce the computation cost. We give the complexity analysis of PBIDM in Table IV. There are two places that need our attention. First, in the *IssueDidCred* phase, the user needs to transfer relevant certificates to the IP, and the IP needs to upload data to the blockchain, we ignore the computation cost of the verification of certificates and the above upload operation, so the total computation cost in this phase is 0. Second, in the *Revoke* phase, the computation cost of the public identity part is also 0, as the

TABLE II
COMPARISON WITH RELATED WORK

	Building blocks	W3C Did Standard	Blindness	Unlinkability	Traceability	Revocability	Public Verifiability
Coconut [9] (2019)	PS signature, ZK proof Blockchain	×	✓	✓	×	×	✓
DIMS [14] (2020)	ZK proof, Blockchain	×	✓	✓	×	×	✓
Yu *et al.* [3] (2020)	BBS+ signautre, ZK proof Accumulator	×	✓	✓	×	✓	✓
DAC [11] (2014)	ZK proof, Blockchain	×	✓	✓	×	×	✓
PBIDM (Ours)	BBS+ signautre, Accumulator, ZK proof, ElGamal encryption, Blockchain	✓	✓	✓	✓	✓	✓

TABLE III
EXPERIMENTAL RESULTS

Notions	Description	Values
T_{bp}	Bilinear pairing	8.69 ms
T_{mul}	Multiplication operation in $\mathbb{G}$	2.60 ms
T_{add}	Addition operation in $\mathbb{G}$	0.01 ms
T_h	Hash operation	0.03 ms
$\lvert\mathbb{G}\rvert$	Bit length of an element in $\mathbb{G}$	1024 bits
$\lvert\mathbb{G}_T\rvert$	Bit length of an element in $\mathbb{G}_T$	1024 bits
$\lvert did\rvert$	Bit length of did in Fig. 3(b)	160 bits
$\lvert tag\rvert$	Bit length of tag in Algorithm 2	1 b

TABLE IV
COMPUTATION COST OF PBIDM

Algorithms		Computation cost
KeyGen	UserKeyGen	1 T_{mul}
	IPKeyGen	2 T_{mul}
	AuditorKeyGen	$d \cdot T_{mul}+(d-1)\cdot T_{add}$
IssueDidCred		0
IssueNymCred		15 T_{mul}+ 10 T_{add}+ 1 T_h
ProveCred	Public part	1 T_H+ 1 T_{mul}
	Private part	6 T_{bp}+20 T_{mul}+ 15 T_{add} + 1 T_h
VerifyCred	Public part	1 T_H+ 2 T_{mul} + 1 T_{add}
	Private part	6 T_{bp} + 15 T_{mul}+ 13 T_{add} + 1 T_h
Trace		d T_{mul}+ 1 T_{add}
Revoke	Public part	0
	Private part	2 T_{mul}+ 1 T_{add}

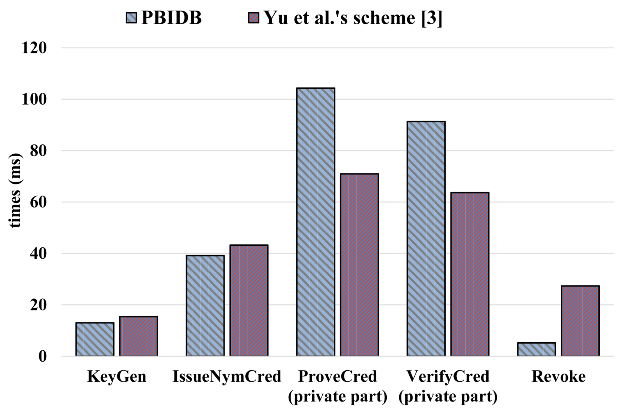


Fig. 4. Comparison of computation cost between PBIDM and Yu et al. 's scheme [3].

TABLE V
COMMUNICATION COST OF PBIDM

Algorithms		Communication cost	Specific values
IssueDidCred		3 $\lvert q\rvert$ + $\lvert claim\rvert$	480 bits + $\lvert claim\rvert$
IssueNymCred		7 $\lvert q\rvert$+3 $\lvert\mathbb{G}\rvert$+$\lvert tag\rvert$	4193 bits
ProveCred	Public part	2 $\lvert q\rvert$ + $\lvert Rq_1\rvert$	320 bits + $\lvert Rq_1\rvert$
	Private part	12 $\lvert q\rvert$ + 5 $\lvert\mathbb{G}\rvert$ + $\lvert tag\rvert$+ $\lvert Rq_2\rvert$	7041 bits + $\lvert Rq_2\rvert$
Trace		$(d-1)\lvert\mathbb{G}\rvert$	2048 bits $(d=3)$
Revoke	Public part	$\lvert q\rvert$	160 bits
	Private part	$\lvert q\rvert$ + $\lvert\mathbb{G}\rvert$ + $\lvert tag\rvert$	1185 bits

IP only needs to invoke `revoke_Did_List`(did_U) on the blockchain.

We summarize the detailed comparison between PBIDM and Yu et al.'s scheme [3] in Fig. 4. PBIDM and [3] have proximity the same overhead in the *KeyGen* phase. In the *IssueNymCred* phase, PBIDM takes 39.13 ms, while [3] takes 43.23 ms. The computation cost is slightly larger than Yu et al.'s scheme [3] in the *ProveCred* and *VerifyCred* phases, PBIDM needs 104.32 and 91.3 ms in the above two phases, while the scheme in [3] needs 70.95 ms and 63.6 ms, respectively. It is worth pointing out that our scheme has advantages over the *Revoke* phase, it only takes 5.21 ms in PBIDM, while Yu et al.'s scheme [3] needs 27.34 ms. We believe that the cost of PBIDM is acceptable for applications in IIoT.

For communication cost, we give the communication cost of PBIDM in Table V. In the *IssueDidCred* phase, the user needs to send did_U and the public attribute b_1 to the IP, and the IP invokes `add_Did_List`$(did_U, claim)$. Thus, the total cost is $3|q| + |claim|$ bits. In the *IssueNymCred* phase, the contents to be communicated are $(C, R, c, z_{s'}, z_{a_1}, A, e, s,'' nym_U, 1, Acc_{new})$, thus the cost is $7|q|$+$3|\mathbb{G}|$+$|tag|$ bits. In the *ProveCred* phase, the contents to be communicated for the public and private parts are (k, s, Rq_1) and $(m_1, m_2, c, A_1, A_2, A_3, z_e, z_s, z_{sk_U}, z_r, z_{r_1}, z_{r_2}, z_{r_3}, z_{\eta_1}, z_{\eta_2}, z_{\eta_3}, z_{a_1}, Rq_2)$, respectively, so the total costs are $(2|q| + |Rq_1|)$ bits and $(12|q| + 5|\mathbb{G}| + |tag| + |Rq_2|)$ bits, respectively. In the *Trace* phase, each auditors broadcasts their $c_1^{sk_i}$, so the cost is $(d-1)|\mathbb{G}|$ bits. In the *Revoke* phase, the contents to be communicated for the public and private parts are (did_U) and $(nym_U, 0, Acc_{new})$, respectively, so the costs are $|q|$ bits and $|q|$ + $|\mathbb{G}|$ + $|tag|$ bits, respectively.

TABLE VI
GAS COST OF PBIDM

Algorithms	Smart contract gas cost
Algorithm 1	137980
Algorithm 2	163946
Algorithm 3	25344
Algorithm 4	47823
Algorithm 5	45999

As for blockchain implementation, the prototype of our smart contract is implemented using Remix.[2] The concrete configuration is the compiler (0.8.6+commit.11564f7e), language (Solidity), EVM version (compiler default), Deployment Environment (JavaScript Virtual Machine). The gas cost of PBIDM is shown in Table VI, where gas cost is a measure of the cost of executing smart contracts on Ethereum. Therefore, it can be observed that PBIDM 's computation and communication costs are reasonable and acceptable in IIoT applications, while providing necessary properties.

VIII. CONCLUSION

In this article, we focus on the privacy-preserving identity management system for industrial Internet of Things. By using BBS+ signature, zero-knowledge proof, distributed EIGamal encryption and blockchain, we put forward a feasible and practical scheme for providing plentiful and necessary properties, i.e., *unforgeability, blindness, unlikability, traceability, revocability*, and *public verifiability*. Furthermore, it is naturally applicable to the W3C standard without any modification. We provide security analysis to ensure reasonable security assurance. Findings from the experiment indicated that PBIDM has reasonable computation and communication costs and is suitable for deployment in IIoT. In future work, we will consider deploying our designed scheme into the actual application scenario. Meanwhile, we will focus on other advanced anonymous signatures, such as short random signatures [4] and structure preserving signatures [25].

REFERENCES

[1] J. Sengupta, S. Ruj, and S. D. Bit, “A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT,” *J. Netw. Comput. Appl.*, vol. 149, 2020, Art. no. 102481.
[2] D. Chaum, “Security without identification: Transaction systems to make big brother obsolete,” *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.
[3] Y. Yu, Y. Zhao, Y. Li, X. Du, L. Wang, and M. Guizani, “Blockchain-based anonymous authentication with selective revocation for smart industrial applications,” *IEEE Trans. Ind. Inform.*, vol. 16, no. 5, pp. 3290–3300, May 2020.
[4] D. Pointcheval and O. Sanders, “Short randomizable signatures,” in *Proc. Cryptographers’ Track RSA Conf.*, 2016, pp. 111–126.
[5] W3C, “Verifiable credentials data model implementation report 1.0.,” Accessed: Jun., 2022. [Online]. Available: https://www.w3.org/TR/vc-data-model
[6] Microsoft, “ION: A public, permissionless, decentralized identifier network,” Accessed: Jun., 2022. [Online]. Available: https://www.github.com/decentralized-identity/ion
[7] WeBank, “Solution of entity identity and trusted data exchange based on consortium blockchain.” Accessed: Jun., 2022. [Online]. Available: https://fintech.webank.com/weidentity/
[8] J. Camenisch and A. Lysyanskaya, “Signature schemes and anonymous credentials from bilinear maps,” in *Proc. Annu. Int. Cryptol. Conf.*, 2004, pp. 56–72.
[9] A. Sonnino, M. Al-Bassam, S. Bano, S. Meiklejohn, and G. Danezis, “Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers,” in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–15.
[10] O. Sanders, “Efficient redactable signature and application to anonymous credentials,” in *Proc. IACR Int. Conf. Public Key Cryptogr.*, 2020, pp. 628–656.
[11] C. Garman, M. Green, and I. Miers, “Decentralized anonymous credentials,” in *Proc. Netw. Distrib. System Secur. Symp.*, 2014, pp. 1–15.
[12] WeBank, “FISCO BCOS.,” Accessed Jun., 2022. [Online]. Available: https://www.fisco.com.cn/
[13] D. Maram et al., “Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability,” in *Proc. IEEE Symp. Secur. Privacy*, 2021, pp. 1348–1366.
[14] X. Yang and W. Li, “A zero-knowledge-proof-based digital identity management scheme in blockchain,” *Comput. Secur.*, vol. 99, 2020, Art. no. 102050.
[15] J. Wang, L. Wu, K.-K. R. Choo, and D. He, “Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure,” *IEEE Trans. Ind. Inform.*, vol. 16, no. 3, pp. 1984–1992, Mar. 2020.
[16] I. Damgaard, C. Ganesh, H. Khoshakhlagh, C. Orlandi, and L. Siniscalchi, “Balancing privacy and accountability in blockchain identity management,” in *Proc. Cryptographers’ Track RSA Conf.*, 2021, pp. 552–576.
[17] M. H. Au, W. Susilo, and Y. Mu, “Constant-size dynamic k-TAA,” in *Proc. Int. Conf. Secur. Cryptogr. Netw.*, 2006, pp. 111–125.
[18] I. Damgård, “On Σ-protocols,” *Lecture Notes*, Dept. Comput. Sci., Univ. Aarhus, Aarhus, Denmark, 2002, p. 84.
[19] A. Fiat and A. Shamir, “How to prove yourself: Practical solutions to identification and signature problems,” in *Proc. Conf. Theory Application Cryptographic Techn.*, 1986, pp. 186–194.
[20] L. Nguyen, “Accumulators from bilinear pairings and applications,” in *Proc. Cryptographers’ Track RSA Conf.*, 2005, pp. 275–292.
[21] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.
[22] F. Brandt, “Efficient cryptographic protocol design based on distributed elgamal encryption,” in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2005, pp. 32–47.
[23] D. Pointcheval and J. Stern, “Security arguments for digital signatures and blind signatures,” *J. Cryptol.*, vol. 13, no. 3, pp. 361–396, 2000.
[24] S. Software Ltd, “Miracle,” Accessed: Jun., 2022. [Online]. Available: https://github.com/miracl/MIRACL
[25] D. Derler and D. Slamanig, “Highly-efficient fully-anonymous dynamic group signatures,” in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2018, pp. 551–565.

[2]Remix is an editor that can quickly write, debug, and deploy smart contracts online. Link: http://remix.ethereum.org.

Zijian Bao received the M.S. degree in computer application technology from the School of Computer Science and Engineering, Northeastern University, Shenyang, China, in 2019. He is currently working toward the Ph.D. degree in cyberspace security with the Key Laboratory of Aerospace Information Security and Trusted Computing Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China.

His research interests include cryptographic protocols.

Debiao He (Member, IEEE) received the Ph.D. degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, Wuhan, China in 2009.

He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. He has authored or coauthored more than 100 research papers in refereed international journals and conferences, such as IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION SECURITY AND FORENSIC, and Usenix Security Symposium. His work has been cited more than 10 000 times at Google Scholar. He is in the Editorial Board of several international journals, such as Journal of Information Security and Applications, Frontiers of Computer Science, and Human-centric Computing and Information Sciences. His main research interests include cryptography and information security, in particular, cryptographic protocols.

Dr. He was the recipient of the 2018 IEEE Sysems Journal Best Paper Award and the 2019 IET Information Security Best Paper Award.

Muhammad Khurram Khan (Senior Member IEEE) received the Ph.D. degree in security and privacy from Southwest Jiaotong University, Chengdu, China.

He is currently a Professor of Cybersecurity with the Center of Excellence in Information Assurance, King Saud University, Riyadh, Saudi Arabia. He is the founder and CEO of the Global Foundation for Cyber Studies and Research, Washington DC, USA. He has edited seven books and proceedings published by Springer-Verlag and IEEE. He has authored or coauthored more than 450 papers in international journals and conferences and he is an inventor of several U.S./PCT patents. His current research interests include cybersecurity, IoT security, biometrics, multimedia security, and digital authentication.

Prof. Khan is a Fellow of the Institution of Engineering and Technology (IET) U.K., a Fellow of the British Computer Society (BCS) U.K., a Fellow of the Future Technology Research Association International (FTRA) Korea, and a Member of the IEEE Technical Committee on Security and Privacy and IEEE Cybersecurity Community. He is the Editor-in-Chief of *Telecommunication Systems* (Springer-Nature). He is a full-time Editor/Associate Editor for several international journals/magazines, including IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, *IEEE Communications Magazine*, IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, *Journal of Network and Computer Applications* (Elsevier), IEEE ACCESS, and *IEEE Consumer Electronics Magazine*.

Min Luo received the Ph.D. degree in computer science from Wuhan University, Wuhan, China, in 2003.

He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. His research interests mainly include applied cryptography and blockchain technology.

Qi Xie received the Ph.D. degree in applied mathematics from Zhejiang University, China, in 2005.

He is a Professor in Hangzhou Normal University, Director of Key Laboratory of Cryptography of Zhejiang Province. He was a Visiting Scholar between 2009 and 2010 with the Department of Computer Science, University of Birmingham, U.K., and a Visiting Scholar with the Department of Computer Science, City University of Hong Kong in 2012. His research area is applied cryptography, including digital signatures, authentication and key agreement protocols, etc. He has published more than 80 research papers in international journals and conferences, and served as general co-chairs of ISPEC2012 and ACM ASIACCS2013, and a reviewer for over 30 international journals.