Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Blockchain-cloud privacy-enhanced distributed industrial data trading based on verifiable credentials

Junli Fang[1], Tao Feng[1*], Xian Guo[1], Rong Ma[1] and Ye Lu[1]

Abstract

Industrial data trading can considerably enhance the economic and social value of abundant data resources. However, traditional data trading models are plagued by critical flaws in fairness, security, privacy and regulation. To tackle the above issues, we first proposed a distributed industrial data trading architecture based on blockchain and cloud for multiple data owners. Subsequently, we realized implemented distributed identity management by the distributed verifiable credentials scheme that possesses the desirable properties, i.e., selective disclosure, multi-show unlinkability, threshold traceability, and public verifiability. Finally, we presented a fair trading mechanism without trusted third parties based on smart contracts, and we employed blockchain and multi-signature to ensure data integrity during data storage and trading. The security and performance analysis shows that our proposal is feasible for sensitive data trading for multiple data owners and provides a useful exploration for future industrial data trading and management.

Keywords Data trading, Verifiable credentials, Multiple owners, Data integrity, Industrial internet

Introduction

Industrial Internet logging rapid expansion in recent years will enhance productivity, and abundant industrial data is a paramount driving force to foster the integrated development of the digital economy and industry [1]. Data trading is conducive to utilize efficiently of data resources by endowing the value of data and bringing benefits to data owners and users [2].

However, a large amount of valuable industrial data has not been collected or merely exploited in a private storage environment, such as the data referring to train control, passenger flow or resource allocation in intelligent transportation systems [3–5]. Data trading is still in its infancy, and there are no industrial privacy-enhanced data trading architectures and practical solutions. Traditional data trading, which relies mainly on private deals and third-party data markets, has been constrained by critical flaws such as fairness and transparency, security of data and identities, as well as the difficulty of appeals and evidence collection [2, 6, 7]. To enhance the value of data, it is urgent to study the trusted sharing and trading mechanism of industrial datasets.

1) The management security of participants is the premises of trading and Industrial Internet systems, but the traditional identity management schemes are trapped by excessive information disclosure, inadequate regulation and are vulnerable to internal attacks or collusive attacks, which are easy to trigger concerns about privacy disclosure and data abuse. Credentials represent the qualification of users and the lack of efficient general credentials with enhanced privacy and regulation will impact blockchain that supports identifiers management smart contracts [1, 8]. Therefore, it

*Correspondence:
Tao Feng
fengt@lut.edu.cn
[1] School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China

is critical to design general credential management with selective disclosure and privacy-enhanced tracking for identity authentication and supervision during industrial data trading.

2) Authenticity and data confidentiality are the foundation of fair trading, however, Industrial Internet data stored in the cloud is under great threats such as unauthorized invaders, data leakage, and integrity damage. Therefore, it is non-trial to take into account the security of key access to protect data from misuse by unauthorized users and integrity to guarantee the accuracy and reliability of data [2].
3) Fair and secure data trading schemes ensure that the trading participants can securely trade and obtain data or profits, however, data tradings based on private and third parties are prone to suffer from rogue tradings. In addition, the data owner's control over the data and secondary reselling of the data need to be taken into account [2, 7].
4) The demand for multiple data owners and users is general in distributed industrial scenarios. The distributed industrial setting requires multiple participants to realize data sharing, data trading, and collaborative calculation, thus it is paramount and more practical to study the multiple data owners' industrial data trading scheme [6].

Cloud computing is deemed to be an adoptable alternative for data service [2, 9, 10] in industrial application-oriented scenarios through numerous cloud service models. Blockchain-cloud paradigm increases the confidence of both consumers and data providers by building a more trustworthy cloud ecosystem [11]. Blockchain is the ideal choice for enhancing both functionality and security/privacy of the cloud ecosystem in varied manners due to outstanding decentralization, immutability, traceability, anonymity, and transparency [11, 12]. The integration of blockchain technology and cloud computing has great potential in enhancing identity authentication and access control, data security and privacy, transaction fairness and efficiency of the decentralized data sharing and supply chain management applications [10, 11, 13, 14]. In particular, the smart contract in blockchain-enabled, applications is an expectable drivening factor to improve the efficiency and accuracy of data processing by automating payment once predefined conditions are satisfied [12].

Thus, we first designed a distributed data trading framework for multiple data owners based on blockchain and cloud aimed at identity, data, and trading security. In this framework, the encrypted data is stored on the cloud service platform, and the smart contract of certificate management and data trading is deployed on the blockchain for identity and trading management.

Secondly, we explored privacy-enhanced user identity management with properties including fine granularity, unlinkability, selective disclosure, and threshold traceability based on the verifiable credentials, which stems from the multi-message version of PS multi-signature.

Finally, we designed a distributed data trading scheme for multiple data owners without a trusted third party. The honest data owners are paid to ensure that the data user receives the correct industrial dataset by providing the correct decryption key, otherwise, the data user can initiate dispute resolution.

The rest of this paper is structured as follows. Some related definitions are recalled and the comparison is discussed in "Related works" section. Then " System model" section describes the system architecture, security model, and design objectives. "Preliminaries" section describes the relevant knowledge and "Proposal" section focuses on the details of the construction. In "Analysis" section, the security and performance are analyzed. Finally, "Conclusion" section concludes the article.

Related works

Researchers explored decentralized data sharing and trading architectures based on blockchain and cloud servers. Fan et al. provided one-to-many data sharing architectures in vehicular networks. Specifically, the data owner outsources the encrypted data to the cloud server and uses the blockchain as a broadcast channel to publish access policies based on attribute-based encryption [9]. Zhang et al. provided a data security sharing model based on privacy protection for IIoT and used blockchain logging technology to trace and account for illegal access [15]. Dai et al. and Zhang et al. designed a data trading ecosystem based on blockchain-cloud and Software Guard Extensions (SGX) architecture [13, 14]. Li et al. realized fair trading without a trusted third party by trading decrypting ciphertext on the blockchain and eliminating secondary sales on the chain [7]. Liu et al. proposed a decentralized transparent data trading scheme [10]. Koutsos et al. realized the privacy-enhanced decentralized data market based on blockchain, functional encryption, and zero-knowledge proof [16]. Liang et al. a blockchain-based fair and fine-grained data trading scheme with privacy preservation using the attribute-based anonymous credential, an authenticated data structure, and zeroknowledge proof [12].

However, most current studies on data trading schemes only focus on a single data owner [6]. Sensitive industrial data usually are traded after being approved by multiple departments. However, some challenges are remaining to be addressed before secure efficient group data trading is

widely applied. The previous works [17, 18] have studied group data sharing but failed to take into account data trading settings. Koutsos et al. proposed A decentralized data market scheme for multiple data owners, which has only focused on data computing privacy but overlooked identity privacy [16]. The scheme of Cao et al. proposed an iterative auction mechanism in the data market with multiple data owners but failed to take into account trading privacy [19].

Cloud servers probably tend to evade the obligations of notified users actively after data corruptions, which are caused by network or operation exceptions, cyberattacks software, and hardware failures [7, 10, 20]. Additionally, data owners are concerned about losing control over their data when using cloud services. Therefore, data integrity verification in the process of data storage and trading based on the cloud is an important security requirement [21–24]. The common techniques of data integrity verification are those based on hash values [7, 9, 18] or Third-Party Auditor (TPA) [22]. However, the methods used in the scheme [7, 9] have a bottleneck on scalability and public verifiability. Public verifiability means that any entity in the network can check the integrity of data stored in the cloud and the technology with TPA is capable of providing public verifiability [22]. However, TPA may dishonestly perform the challenge and response protocol and even deceive users by colluding with the cloud To address the above issues, blockchain is adopted to generate randomly challenging information against malicious TPA [12, 23, 24]. As stated in the previous paragraph, the data with public verifiability of multiple data owners especially the schemes is attractive in practical industrial environments. Fortunately, Wang et al. proposed an efficient public verifiable multi-owner data integrity verification solution based on multiple signatures [22], which contributes to our goal.

The precondition of secure data trading is privacy-enhanced identity management, especially identity privacy and traceability. The works in [10, 14, 17, 18] support identity privacy, while other schemes do not. The schemes [10, 17, 18] also provide the tracking of malicious users. However, most existing schemes employed ordinary identity authentication technologies that have the disadvantage of coarse granularity, privacy disclosure, being hard to trace, and lack of generalization and flexibility in the distributed setting. Verifiable Credentials (VCs) is the standardized digital credential with cryptographic security, privacy protection, and machine-readability and is one of the promising evolution supporting decentralized identity authentication [25].

As shown in Fig. 1, the credential holder requests a credential from the issuer, who provides trust endorsement for the holder by issuing credentials on relevant attributes. Then, the holder either saves the credential in the local wallet or hosts it on the blockchain. When trying to access a certain service, the credential holder presents the credential to the verifier who confirms whether the holder has met the requirements by verifying claims of the credential.

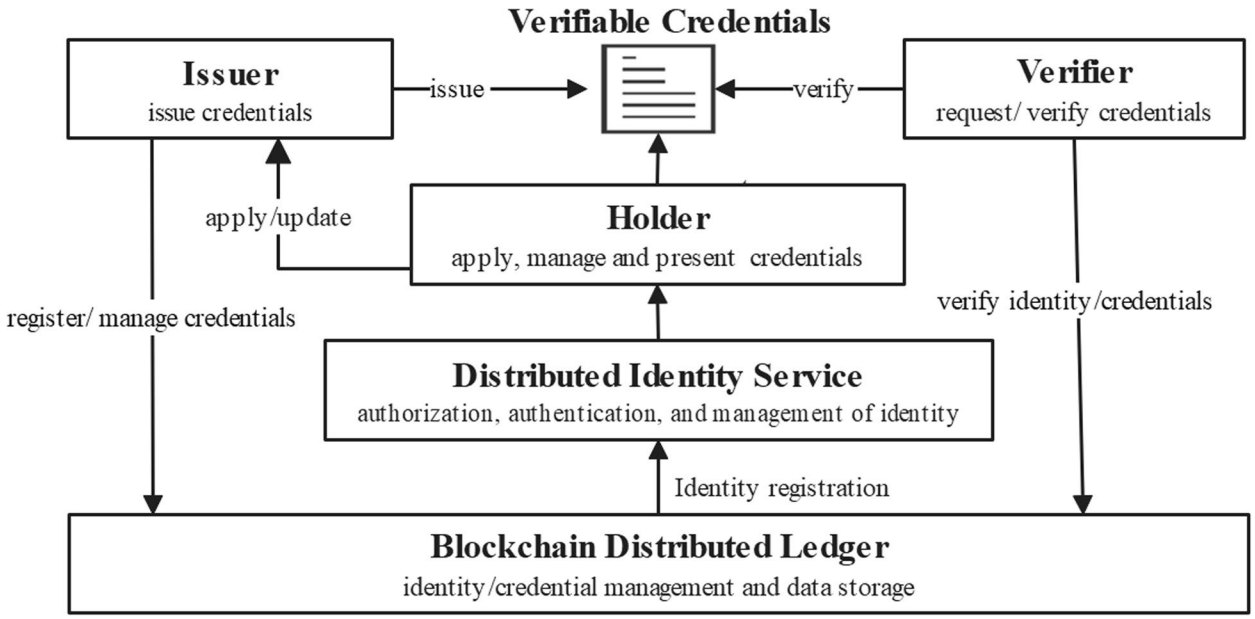

Fig. 1 Verifiable credentials model

Verifiable credentials have captured increasing interest and growing Support due to notable benefits [25]. Li proposed a verifiable credential scheme with selective disclosure based on the Bohen-Lynn-Shacham aggregate signature [26]. Yoon et al. proposed a personal data trading system based on blockchain, which uses verifiable credentials to prove the ownership of data [27]. Fotiou et al. proposed a personal data trading system based on verifiable credentials [28]. Moreover, the methods of [10] essentially support verifiable credentials. More works focus on the exploration of various fields [25, 29–31]. Therefore, we aim to construct the fair trading of multiple data owners based on cloud storage and blockchain architecture and privacy-enhanced identity management based on verifiable credentials.

Unfortunately, in order to enhance privacy and regulation, verifiable credentials are expected to prove the credentials' ownership in the multi-show unlinkable, blind issuance, fine-grained attribute privacy, minimized disclosure, and traceability. Multi- show unlinkability allows users to present credentials multiple times whereas verifiers are prevented from tracking users by link continuous session authentication services. Minimizing disclosures aims to void to leak of unnecessary information [25, 29]. However, most previous proposals failed to support all the above requirements. Essentially, the multi-show unlinkability of credentials stems from the re-randomizablity of the signature, which allows the signatures of the same message to be randomized into another unlinkable signature version without knowledge of the secret key [32, 33]. Minimized disclosure is realized by means of selective disclosure and zero-knowledge-proof technology [12]. Selective disclosure allows the credential holder to present corresponding subsets of credential attributes to verifiers who are still allowed to validate the whole credential. Zero-knowledge proof refers to the prover can prove the authenticity of the claim to the verifier, but does not reveal any further information about the claim except the authenticity of the claim.

Fortunately, the emerging Pointcheval-Sanders (PS) signature, which supports re-randomizable, efficient knowledge proof and multi-message signatures is expected to propel the evolution of privacy-enhanced verifiable credentials [32, 33]. Yu et al. proposed an anonymous authentication scheme called BASS supporting attribute privacy, selective revocation, credential soundness, and multi-showing-unlinkability, but the scheme is inefficient and inadequate in the corrupted distributed authorities setting [34]. García-Rodríguez et al. considered distributed credential issuance but overlooked corrupted authority [31]. Sonnino et al. proposed a scheme called Coconut which supports threshold issuance, selective disclosure, and multiple unlinkable selective [8].

However, unconditional identity privacy is not harmonious with the supervision of the regulator and may trigger illegal profits and data abuse in data trading. In the scheme with traceability, the issuing authority and the tracing authority are usually single or monolithic and thus are assumed to be trusted. However, corrupted issuing authorities may forge certificates, and corrupted tracking authorities can remove the anonymity of certificates. The proposals with multiple issuance authorities and tracking authorities are desirable and practical for distributed industrial applications. It is also very necessary to reveal the real identity of illegal users in a privacy-enhanced way. The scheme [35] separates the credential issuance and trace function, and also exposes the user's identity through threshold trace to prevent the corruption of a single trace authority from harming the privacy of legitimate users. Camenisch et al. fine-tuned an authentication mechanism for V2V and Liu et al. for the data market [10, 35].

The features of the above schemes are presented in Table 1. Following previous excellent works, we ameliorated the privacy-enhanced verifiable credentials to the needs of a distributed Industrial data trading scheme, which supports distributed blindly insurance, fine-grained, threshold traceability, multi-show unlinkability, and selective disclosure in the setting of multiple data owners and regulators.

System model

In this section, we briefly define the system model, security model, and security goals of our scheme (Table 2).

Architecture

The system model is depicted in Fig. 2, which involves five entities: cloud server (CS), blockchain (BC), data owners (DOs), data users (DUs), and regulators (RGs).

BC: Credential management smart contract (CMSC) and data trading smart contract (DTSC), and the credential revocation registry are deployed on BC. The CMSC and DTSC are in charge of handling and recording the operations of entities in the trading process. In addition, BC plays the role of the public verifier and executes data integrity verification protocols.

CS: The semi-trusted cloud server is responsible for storing data and providing data integrity proof to DOs and DUs.

DOs: DOs expect to gain benefits from selling data and perform data trading through CS and DTSC. In addition, DOs are in charge of issuing credentials to data users.

Table 1 Related works

Scheme	Cloud-based	Blockchain-based	Trade fairness	Multiple date owners	Integrity verification	Public verifiability	Identity privacy	Traceability
[7]	√	√	√	×	√	×	×	×
[9]	√	√	×	×	√	√	×	×
[15]	√	√	×	×	×	×	×	×
[13]	√	√	√	×	×	×	×	×
[14]	√	√	√	×	×	×	√	×
[10]	√	√	√	×	×	√	√	√
[16]	√	√	√	√	×	×	×	×
[17]	√	×	×	√	-	×	√	×
[18]	√	√	×	√	√	√	√	√
[19]	×	×	×	√	-	×	×	×
Ours	√	√	√	√	√	√	√	√

Table 2 Scheme comparison of verifiable credentials

Schemes	Distributed issurance	Blindly issurance	Fine-grained	Threshold traceability	Anonymity	Multi-show unlinkability	Selective disclosure
[34]	×	√	√	×	√	√	×
[35]	×	√	√	√	√	√	√
[8]	√	√	×	×	√	√	√
[26]	√	×	×	×	√	×	√
[31]	√	√	√	×	√	√	√
[10]	√	√	×	√	√	√	×
Ours	√	√	√	√	√	√	√

DUs: DUs will pay DOs after obtaining the correct data through CS and DTSC. DUs obtain multiple credentials from distributed DOs through CMSC. DUs selectively disclose the aggregated credential to CS for authentication.

RGs: RGs consist of multiple independent regulators departments and play the role of the dispute arbitration commission and the tracing authorities. In addition, RGs are responsible for generating global parameters and maintaining the blockchain.

Security model

We regard that RGs, CS, DOs, and DUs are semi-trusted and may deviate from the agreement for benefits. Thus, we introduce the adversaries $\mathcal{A}_{CS}$, $\mathcal{A}_{DOs}$, $\mathcal{A}_{DUs}$ and $\mathcal{A}_{RGs}$ in the model [7, 9, 23, 35].

$\mathcal{A}_{CS}$ is curious to infer sensitive information about cloud users and data. Additionally, CS usually carries out the protocols honestly but may conceal the data corruptions that are caused by network or operation exceptions, cyberattacks, as well as software or hardware failures.

$\mathcal{A}_{DOs}$ may launch attacks such as issuing untraceable illegal credentials for malicious users, uploading forged or wrong ciphertext to gain benefits, and denying or claiming ownership of the dataset.

$\mathcal{A}_{DUs}$ may launch attacks such as stealing data with illegal credentials, refusing payment by forging evidence (keys or signatures), and second reselling.

$\mathcal{A}_{RGs}$ tries to corrupt the regulators to reveal the user's identity.

The security assumptions are as follows: assume that there are secure authenticated broadcast channels between regulators and multiple data owners; assume that BC is a secure distributed infrastructure without vulnerabilities.

Security goals

Our goal is to create a distributed privacy-enhanced and supervisable industrial data trading scheme, and we proposed the security goal from the perspectives of identity, data, and trading security [7, 21, 35].

Identity security

Identity security should meet the following attributes:

Anonymity: Anonymity ensures that the identity information of DUs is not disclosed as long as the

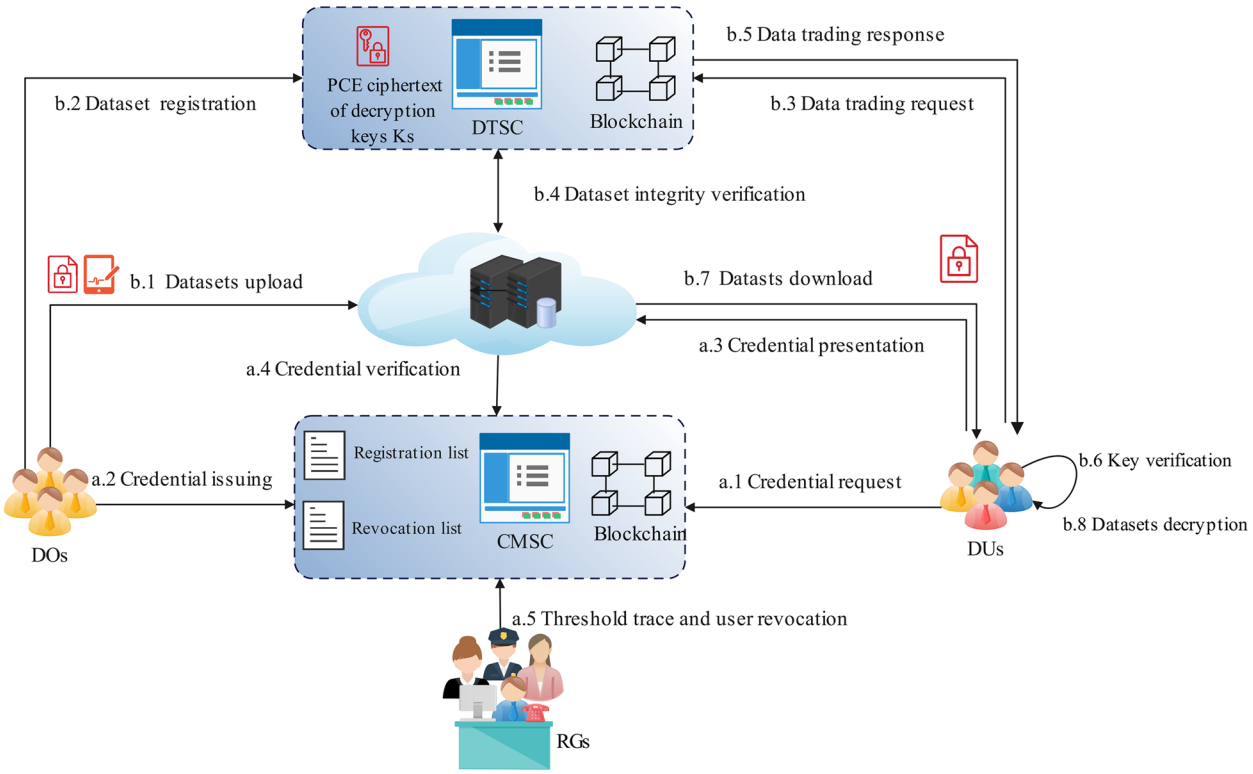

Fig. 2 System model

issuer is honest and the presenting credential is not opened.

Threshold traceability: A certain amount of RGs can jointly reveal the real identity of DUs.

Blind issuance: DOs do not obtain the user's secret identity information when issuing credentials.

Multi-show unlinkability: Multi-show allows DUs to present the same credentials more than once, whereas unlinkability means it is impossible to associate the presented credentials even if all authorities conspire. Multi-show unlinkability can prevent the verifier from tracking the user's activities through the authentication service of continuous sessions.

Selective disclosure: DUs only show a subset of the attributes required to prove the ownership of the credential, whereas the verifier only sees the disclosed attributes and still verifies the entire credentials. In other words, the credential verifier only obtained the cryptographic verification result of the attribute instead of the plaintext or ciphertext of the attribute.

Data security

Data security includes confidentiality, integrity, and data authenticity.

Data confidentiality: The original data can only be stored and sold through encryption, and only authorized users can obtain the decryption key.

Data integrity & public verifiability: Any verifier is able to audit the correctness of multi-owner data in the cloud with the public keys of all the owners.

Data authenticity: Data cannot be forged (i.e. unforgeability of trading data), and a malicious DO cannot deny or illegally claim the ownership of a dataset (i.e. Non-repudiation of data ownership).

Trading security

Trading security aims to ensure trading fairness and non-repudiation and to resist the second reselling attack.

Trade Fairness: Fair trading means DU must pay the corresponding fee once obtains the correct datasets and the DOs gain corresponding benefits from selling their datasets. Meanwhile, DU is allowed to refuse to pay after finding the data problem and to submit a complaint [7].

Trade non-repudiation: On the one hand, DOs or DUs may deny the trade for some reason. On the other hand, the proposal should resist the forged-evi-

dence-attack, namely, should prevent malicious DUs from rejecting payment using forged decryption keys or signature of data ciphertext during the dispute.
Anti-second-reselling-attack: The second resale attack refers that malicious DUs may resell the data to gain benefits. The proposal should avoid the second resale of data on the chain to protect the interests of DOs.

Preliminaries

Multi-signature

Multi-signature allows multiple signers to sign for an identical message. Wang et al. proposed a blockless verifiable multi-signature scheme and employed it in the public verification mechanism in the cloud environment. The multi-signature scheme consists of the following algorithms [22]:

MS.Setup$(1^\lambda) \rightarrow params$: input a security parameter λ, and outputs $params \leftarrow (p, \mathbb{G}, \mathbb{G}_T, e, g, \mathcal{H})$, where $g \in_R \mathbb{G}$ is a generator of $\mathbb{G}$, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map, $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{G}$ is a hash function.

MS.KGen$(params) \rightarrow (vsk_i, vpk_i)$: Let $\mathcal{I} = \{I_i\}_{\in[1,n]}$ be the set of n signers and given global parameters $params$, $I_i \in \mathcal{I}$ generates $vsk_i = v_i \in_R \mathbb{Z}_p^*$ and $vpk_i = g^{v_i} \in \mathbb{G}$.

MS.Sign$(m,mid,vsk_i) \rightarrow \psi_i$: given block $m \in \mathbb{Z}_p^*$ and its identifier mid, $I_i \in \mathcal{I}$ computes local signature respectively $\psi_i = [\mathcal{H}(mid)g^m]^{v_i}$ and broadcasts ψ_i.

MS.SAgg$(\{vpk_i\}_{i=1}^n, \{\psi_i\}_{i=1}^n) \rightarrow \psi$: after receiving n local signatures $\{\psi_i\}_{i=1}^n$ on block m, the designed signer outputs an aggregated signature $\psi = \prod_{i=1}^n \psi_i$.

MS.Verify$(m, mid, \psi, \{vpk_i\}_{i=1}^n) \rightarrow$ (1,0): given block $m \in \mathbb{Z}_p^*$ and its identifier mid, signature ψ and all the signers public keys $\{vpk_i\}_{i=1}^n$, the verifier computes $pk_s = \prod_{i=1}^n vpk_i$ and outputs 1 if $e(\psi, g) = e(\mathcal{H}(mid) \cdot g^m, pk_s)$ or 0 otherwise. The correctness of the above equation can be proved as

$$\begin{aligned} e\left(\psi, g\right) &= e\left(\prod_{i=1}^n \psi_i, g\right) \\ &= e\left(\prod_{i=1}^n (\mathcal{H}(mid)g^m)^{v_i}, g\right) \\ &= e\left(\mathcal{H}(mid)g^m, pk_s\right) \end{aligned}$$

Multi-message Pointcheval-Sanders multi-signature

Pointcheval-Sanders multi-signature is an interesting privacy-enhanced cryptographic primitive due to supporting public key aggregation, signature re-randomization, and efficient signature zero-knowledge proof. PS multi-signature consists of the following algorithms [32, 35]:

PSM.Setup$(1^\lambda) \rightarrow params$: input a security parameter λ, and output global parameters $params \leftarrow \left(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e, g, \tilde{g}, n, k\right)$, where $g \in_R \mathbb{G}$ and $\tilde{g} \in_R \tilde{\mathbb{G}}$ are the generators of $\mathbb{G}$ and $\tilde{\mathbb{G}}$ respectively. $e : \mathbb{G} \times \tilde{\mathbb{G}} \rightarrow \mathbb{G}_T$ is a bilinear map, n and k are the number of signers and messages respectively.

PSM.KGen$(params) \rightarrow (sk,pk)$: given $params$, returns key pair (sk,pk). This algorithm randomly selects $(x, y_1, \cdots, y_{k+1}) \in_R \mathbb{Z}_p^*$, computes $(\tilde{X}, \tilde{Y}_1, \cdots, \tilde{Y}_{k+1}) \leftarrow (\tilde{g}^x, \tilde{g}^{y_1}, \cdots, \tilde{g}^{y_{k+1}})$, and sets $sk \leftarrow (x, y_1, \cdots, y_{k+1})$, $pk \leftarrow (X, Y_1, \cdots, Y_{k+1})$.

PSM.KAgg$(pk_1, \cdots, pk_n) \rightarrow apk$: given $(pk_1, \cdots, pk_n)$, generates $(t_1, \cdots, t_n) \leftarrow \mathcal{H}_1(pk_1, \cdots, pk_n)$ and then returns $apk := (\tilde{X}', \tilde{Y}_1', \cdots, \tilde{Y}_{k+1}') = \prod_{i=1}^n ipk_i^{t_i} = (\tilde{g}^{\sum_{i=1}^n x_i t_i}, \tilde{g}^{\sum_{i=1}^n y_{i,0} t_i}, \tilde{g}^{\sum_{i=1}^n y_{i,1} t_i}, \cdots, \tilde{g}^{\sum_{i=1}^n y_{i,k+1} t_i})$.

PSM.Sign$(sk,\mathbf{m} = (m_1, \cdots, m_k)) \rightarrow \sigma$: given message $\mathbf{m} = (m_1, \cdots, m_k)$, private key sk, generates $(m', h) = \mathcal{H}_0(m_1, \cdots, m_k) \in \mathbb{Z}_p^* \times \mathbb{G}^*$, $\sigma_1 := h$, $\sigma_2 := h^{x + \sum_{j=1}^k y_j \cdot m_j + y_{k+1} \cdot m'}$ and returns $\sigma := (m', \sigma_1, \sigma_2) \leftarrow (m', h, h^{(x + \sum_{j=1}^k y_j \cdot m_j + y_{k+1} \cdot m')})$.

PSM.SAgg$((pk_i)_{i=1}^n, \boldsymbol{m} = (m_1, \cdots, m_k), (\sigma_i)_{i=1}^n) \rightarrow \sigma$: after receiving n local signatures $\sigma_i := (m'\sigma_{i,1}, \sigma_{i,2})_{i\in[n]}$ on message $\boldsymbol{m} = (m_1, \cdots, m_k)$, computes $\sigma_2 := \prod_{i=1}^n \sigma_{i,2}^{t_i} = h^{\xi + \sum_{i=1}^k u_\ell m_\ell + u_{k+1} m'}$, where $\xi := \Sigma_{i=1}^n x_i t_i$, $u_\ell = (\Sigma_{i=1}^n y_{i,\ell} t_i)_{\ell \in [1,k]}$, $u_{k+1} = \Sigma_{i=1}^n y_{i,(k+1)} t_i$, outputs aggregate signature $\sigma = (m', \sigma_1, \sigma_2)$.

PSM.Verf$(apk, \sigma, (m_1, \cdots, m_k)) \rightarrow 1/0$: given message $\boldsymbol{m} = (m_1, \cdots, m_k)$, signature σ and aggregated public key apk, set $\sigma = (m', \sigma_1, \sigma_2)$. This algorithm verifies if $\sigma_1 \neq 1_{\mathbb{G}}$ and $e\left(\sigma_1, \tilde{X}' \cdot \prod_{j=1}^k \tilde{Y}_j'^{m_j} \tilde{Y}_{k+1}'^{m'}\right) = \left(\sigma_2, \tilde{g}\right)$, outputs 1 if and 0 otherwise. The correctness of the above equation can be proved as

$$\begin{aligned} e\left(\sigma_2, \tilde{g}\right) &= e(h^{\xi + \sum_{j=1}^k u_j \cdot m_j + u_{k+1} \cdot m'}, \tilde{g}) \\ &= e(h, \tilde{g}^{\xi + \sum_{j=1}^k u_j \cdot m_j + u_{k+1} \cdot m'}) \\ &= e(h, \tilde{g}^{\sum_{i=1}^n x_i t_i} \tilde{g}^{\sum_{i=1}^n \sum_{j=1}^k y_{i,j} t_i \cdot m_j} \tilde{g}^{\sum_{i=1}^n y_{i,(k+1)} t_i m'}) \\ &= e(\sigma_1, \tilde{X}' \cdot \prod_{j=1}^k \tilde{Y}_j'^{mj} \tilde{Y}_{k+1}'^{m'}) \end{aligned}$$

ElGamal encryption

ElGamal encryption is a public key encryption primitive and consists of the following algorithms [36]:

ElG.KGen$(\mathbb{G}) \rightarrow (sk, pk)$: given $\mathbb{G}$, sets $sk = x \in_R \mathbb{Z}_p^*$, $pk = g^x$ and returns key pair (sk, pk).

ElG.Enc$(pk,m) \rightarrow c$: given public key pk, message $m \in \mathbb{Z}_p^*$, randomly selects $r \in_R \mathbb{Z}_p^*$ and returns corresponding ciphertext $c = (c_1, c_2) = (g^r, pk^r g^m)$.

ElG.Dec$(sk,c) \rightarrow g^m$: given private key sk and ciphertext c, returns $c_2 / c_1^{sk} = g^m$.

Plaintext checkable encryption

Plaintext checkable encryption (PCE) is a new public key encryption primitive, which provides a feasible way to check whether $c \in \mathcal{C}$ is the ciphertext of the plaintext message $m \in \mathcal{M}$ under the public key $ppk \in \mathbb{G}$ (where $\mathcal{M}$ and $\mathcal{C}$ respectively represent plaintext space and ciphertext space). PCE consists of the algorithms described below [37]:

PCE.Setup$(1^{\lambda}) \rightarrow$ *params*: *input a security parameter* λ*, and output* global parameters *params* $\leftarrow (p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e, g, \tilde{g})$*, where* $g \in_R \mathbb{G}$ and $\tilde{g} \in_R \tilde{\mathbb{G}}$ is the generator of $\mathbb{G}$ and $\tilde{\mathbb{G}}$, $e : \mathbb{G} \times \tilde{\mathbb{G}} \rightarrow \mathbb{G}_T$ is a bilinear map.

PCE.KGen(*params*) $\rightarrow$ (*psk*,*ppk*): given *params*, sets $psk := x \in_R \mathbb{Z}_p^*$, $ppk := g^x$, and return key pair (psk, ppk).

PCE.Enc(*ppk*,*m*) $\rightarrow c$: given public key *ppk*, message *m*, randomly selects α, $\beta \in_R \mathbb{Z}_p^*$,and returns corresponding ciphertex $c = (mg^{\alpha x}, g^{\alpha}, \tilde{g}^{\beta}, \tilde{g}^{\alpha\beta})$.

PCE.Dec$(psk, c) \rightarrow m$: given private key *psk* and ciphertext $c = (c_1,c_2,c_3,c_4)$, if c meets $e(g,c_4) = e(c_2,c_3)$ the algorithm returns corresponding plaintext $m = c_1/c_2^{psk}$ else return $\perp$.

PCE.Check(*m*,*ppk*,*c*) $\rightarrow (1, 0)$: given public key *ppk*, ciphertext $c = (c_1,c_2,c_3,c_4)$ and decrypted message *m*. This algorithm first verifies that the correctness of the ciphertext and return 0 if $e(g,c_4) \neq e(c_2,c_3)$, and then verifies if $e(c_1/m,c_3) = e(ppk,c_4)$, the algorithm outputs 1 means that c is an encryption result of *m* and 0 otherwise.

Zero-knowledge proofs

Zero-knowledge proof(ZKP) refers that the prover can prove the authenticity of the claim to the verifier, but does not reveal any further information about the claim except the authenticity of the claim. In general, it consists of algorithms described as follows:

NIZK.Setup$(1^{\lambda}) \rightarrow$ *params*: inputs a security parameter λ, and outputs global parameters *params*.

NIZK.Proof$\{x{:}R = (s, w)\} \rightarrow \pi$: inputs statements s and w, the prover constructs proof π to prove the secret value x satisfies relation $R{=}\ (s, w)$ in a zero-knowledge way.

NIZK.Verf(*params*,*s*,π) $\rightarrow$ (1,0): The verifier verifies the correctness of the proof π.

Verifiable secret sharing

The (τ, η) VSS scheme allows a dealer to share the secret s with η participants $\{S_1 \cdots S_\eta\}$ and $\tau - out - of - \eta$ participants can jointly combine shares and recover s. To prevent dishonest dealers from cheating on generating shares, VSS can support proof of encryption shares and public verifiability. Specifically, the VSS scheme consists of the following algorithms [38]:

VSS.KGen(*params*,$i \in [1, \eta]) \rightarrow (opk_i, osk_i)$: Each participant $S(i)$ runs **ElG.KGen** algorithm to obtain key pair (opk_i, osk_i), where $osk_i \leftarrow z_i \in_R \mathbb{Z}_p^*$, $opk_i \leftarrow \tilde{f}_i := \tilde{g}^{z_i} \in \tilde{\mathbb{G}}^*$.

VSS.Share(***opk***$, s, h, \tilde{g}) \rightarrow ((V_\ell)_{\ell\in[1,\tau]}, (\widetilde{C}_i, \pi_{Pi})_{i\in[1,\eta]})$: The dealer distributes the share of the secret s to $S(i)$. Firstly, the dealer randomly selects $(p_1, \cdots, p_\tau) \in_R \mathbb{Z}_p$, and computes the polynomial $\mathbf{P}(\chi) \leftarrow s + \Sigma_{\ell=1}^{\tau} p_\ell \chi^\ell \in \mathbb{Z}_p[\chi]$ and $H_s \leftarrow h^s$. Secondly, for $\forall i \in [1, \eta]$, the dealer sets $s_i \leftarrow \mathrm{P}(i)$ as the share of the secret s and distribute s_i to $S(i)$. Next, for $\forall \ell \in [1,\tau]$, the dealer computes and publishes $V_\ell \leftarrow h^{p_\ell}$ which is the verification value. Then, for $\forall i \in [1,\eta]$, the dealer randomly selects $r_i \in_R \mathbb{Z}_p$ and computes both $\widetilde{C}_i := (\widetilde{C}_{i,0}, \widetilde{C}_{i,1}) \leftarrow (\tilde{g}^{r_i}, \tilde{f}_i^{r_i}\tilde{g}^{s_i})$ and $\pi_{Pi} \leftarrow$ **NIZK.Prove**$\{r_i : \widetilde{C}_{i,0} = \tilde{g}^{r_i}, e(h, \widetilde{C}_{i,1}/\tilde{f}_i^{r_i}) = e(H_s \Sigma_{\ell=1}^{\tau} V_\ell^{i_\ell}, \tilde{g})\}$. Finally, the dealer broadcasts $((V_\ell)_{\ell\in[1,\tau]}, (\widetilde{C}_i, \pi_{Pi})_{i\in[1,\eta]})$ to all $S(i)$.

VSS.Verf$(\tilde{C}_i, \pi_{Pi}) \rightarrow (1_i, 0)$: Each participant $S(i)$ checks the correctness of the proof π_{Pi} in the way of Zero-knowledge Proof.

VSS.Recover$(\{\tilde{C}_i, opk_i\}_{i=1}^{\tau}) \rightarrow \tilde{g}^s$: Let $\mathcal{O}$ be the set of τ participants. Firstly, each $S_i \in \mathcal{O}$ computes $\tilde{C}_i^* = \tilde{C}_{i,1}/(\tilde{C}_{i,0}^{z_i}) = \tilde{g}^{s_i}$ with the secret key and broadcasts $\tilde{C}_i^*$ to other participants. If the recovered share copies from other participants $S(j)_{j\in\mathcal{O}\{i\}}$ and $\tilde{C}_i^* \neq \perp$ (namely all the recovered shares are correct), for all $j \in \mathcal{O}$, the openers calculate each Lagrange coefficient $w_j \leftarrow \prod_{j\in\mathcal{O}\{i\}} j/(j-i)$, and then calculate $\prod_{j\in\mathcal{O}}(\tilde{C}_i^*)^{w_j} = \tilde{g}^s$, the secret s is finally reconstructed over $\tilde{g}$.

Proposal

Overview

We aim to propose a verifiable distributed industrial dataset trading architecture model without trusted third parties. Furthermore, we designed a privacy-enhanced verifiable distributed identity management scheme and a verifiable data trading scheme.

In the identity management phase, DU requests verifiable credentials from DOs through CMSC and DOs employ PSM multi-signature to issue local credentials for validated DU. Significantly, we provided privacy-enhanced identity management for DU by employing the PSM multi-message signature to realize fine-grained attributes, selective attribute disclosure, and threshold traceability [35]. CMSC is in charge of aggregating and distributing credentials to DU. CMSC or DU randomize and selectively disclose the credentials to CS before data trading. CS verifies the validity of the credentials and provides corresponding services to DU with unrevoked and validated credentials. Although verifiable credentials protect users' privacy through anonymity and selective disclosure, DUs are not always honest. Therefore, RGs need to disclose or even revoke malicious credential holders in the way of the threshold. It is worth noting that threshold traceability contributes to avoiding the abuse of the power of RGs and further improving the privacy of data users.

The data trading phase aims to secure fair trading of the industrial dataset through the trading management smart contract DTSC. Before trading, DOs encrypt the dataset

using symmetric encryption technology and upload the ciphertext as well as the signature of the dataset to CS. Then CS verifies the signatures and registers the datasets to DTSC. Later, DU retrieves the interested dataset and submits the data trading request to DTSC. Once DTSC verifies the trading request of DU successfully and CS returns the correct data integrity verification result, DOs send the decryption key encrypted by the PCE algorithm to the authorized DU through DTSC. After receiving the PCE ciphertext, DU recovers the AES key and verifies the decryption result with his PCE private key. Next, DU downloads the ciphertext of the dataset and decrypts the original data. Finally, DU pays the fee before the specified time if the decryption result is correct. Otherwise, DU will refuse payment and submit evidence for dispute resolution. In the dataset integrity verification stems from a public verifiable multi-owner data scheme [22], and blockchain is employed to replace traditional TPA to reduce the problem of malicious third parties in the public audit.

Distributed identity management

Distributed identity management includes the stages of initialization, credential request, generation, aggregation, presentation, verification, trace and revocation (Fig. 3). Notations are shown in Table 3.

Initialization

In the initialization stage, public parameters and keys related to the subsequent processes are generated.

(1) RGs run function **CredSetup** to generate public parameters *params*, and then upload *params* to CMSC. It is assumed that *params* are implicit inputs of all other algorithms.
(2) DO $I_i \in \mathcal{I}$ runs function **IKGen** to generate the issuing key pair (isk_i, ipk_i) and initializes the registration list $\boldsymbol{RegList}_i$. Then, DOs upload $\boldsymbol{ipk} := \{ipk_i\}_{i\in[1,n]}$ to CMSC.
(3) RG $S_j \in \mathcal{S}$ runs function **OKGen** to generate the trace key pair (opk_j, osk_j) and initializes the trace list $\boldsymbol{TraceList}_j$. Then, RGs upload $\boldsymbol{opk} := \{opk_j\}_{j\in[1,\eta]}$ to CMSC.
(4) CMSC runs the **KAgg** algorithm to generate the aggregated public key *apk* and outputs $gpk := (\boldsymbol{ipk}, \boldsymbol{opk}, apk)$.

Credential request

In this stage, $U_{uid} \in \mathcal{U}$ with unique identification $uid \in_R \mathbb{Z}_p^*$ runs function **CredReq** to send the credential request $\boldsymbol{CredReq}_{uid}$ to $I_i \in \mathcal{I}$ for attribute sets $\boldsymbol{Attr}$ through CMSC. Firstly, U_{uid} generates additional scalar a' and public base h, then computes the user's secret identity H_{uid} and G_{uid} as well as zero-knowledge proof π_{uid}. Next, U_{uid} calls the algorithm **VSS.Share** to compute the share uid_j of the identity uid for $S_j \in \mathcal{S}$ and send them to all $\mathrm{RG_S}$ through CMSC. Finally, U_{uid} uploads $\boldsymbol{CredReq}_{uid} \leftarrow (G_{uid}, H_{uid}, \pi_{uid}, \{\tilde{C}_j, \pi_{Pj}\}_{j\in[1,\eta]}, \{V_\ell\}_{\ell\in[1,\tau]}, (a', h))$ to CMSC, where $\tilde{C}_j$ is the ciphertext of share uid_j and π_{Pj} is corresponding zero-knowledge proof, $\{V_\ell\}_{\ell\in[1,\tau]}$ is the verification value of the secret sharing algorithm.

Credential generation

After receiving $\boldsymbol{CredReq}_{uid}$, $I_i \in \mathcal{I}$ runs function **CredGen** to generate local credential $\boldsymbol{CredGen}_{i,uid}$ for valid users. Specifically, I_i will reject the credential request if there is duplicate registration, illegal identity, or incorrect share encryption. Otherwise, I_i will call the algorithm **PSM.Sign** to sign blindly on the user's identity uid and attributes $\boldsymbol{Attr}$. Finally I_i updates $\boldsymbol{RegList}_i$ and uploads $\boldsymbol{CredGen}_{i,uid}$ to CMSC.

Credential aggregation

After receiving all local credentials from $I_i \in \mathcal{I}$, CMSC runs the function **CredAgg** to generate aggregated credentials $\boldsymbol{CredAgg}_{uid}$. Specifically, CMSC first calls the algorithm **PSM.Verf** to verify whether the local credentials are correct. Next, Run the algorithm **PSM.SAgg** to aggregate $\{\boldsymbol{CredGen}_{i,uid}\}_{i=[1,n]}$ into complete credentials $\boldsymbol{CredAgg}_{uid}$. Then the algorithm **PSM.Verf** is called to verify the validity of $\boldsymbol{CredAgg}_{uid}$. If the validation is passed, $\boldsymbol{CredAgg}_{uid}$ will be issued to U_{uid} who will store his credential in the local wallet.

Credential presentation

When required to present trading qualifications, U_{uid} personally present credentials $\boldsymbol{Cred}_{uid,\rho}$ on the selective presentation attribute subset $\rho \subseteq \boldsymbol{Attr}$. The function **CredProve** will be called to randomize the signature of credential $\boldsymbol{CredAgg}_{uid}$ and generate the proof of signature.

Credential verification

During the trading, CS will check the validity of $\boldsymbol{Cred}_{uid,\rho}$ through function **CredVerify**, namely, verify the signature and its zero-knowledge proof in $\boldsymbol{Cred}_{uid,\rho}$.

Threshold trace and user revocation

When RGs receive complaints or inspect abnormal credentials, $\tau+1$ RGs $\mathcal{O}_i \in \mathcal{O}$ will run the function **Trace** to recover the identity of U_{uid} who computed the credential $\boldsymbol{Cred}_{uid,\rho}$. Firstly, $\mathcal{O}_i \in \mathcal{O}$ runs $\mathbf{CredVerify}(apk, \rho, \boldsymbol{Attr}, \boldsymbol{Cred}_{uid,\rho})$ to verify the validity of $\boldsymbol{Cred}_{uid,\rho}$. Then, $\mathcal{O}_j \in \mathcal{O}$ downloads $\boldsymbol{CredReq}_{uid} \leftarrow (G_{uid}, H_{uid}, \pi_{uid}, \{V_\ell\}_{\ell\in[1,\tau]}, \{\tilde{C}_j, \pi_{Pj})_{j\in[1,\eta]})$

$\mathbf{CredSetup}(1^\lambda, n, \eta, k) \to params = (p, \mathrm{G}, \tilde{\mathrm{G}}, \mathrm{G}_T, e, g, \tilde{g}, n, \eta, \tau, k, \mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2)$:

- $(p, \mathrm{G}, \tilde{\mathrm{G}}, \mathrm{G}_T, e, g, \tilde{g}, n, k) \leftarrow \mathbf{PSM.Setup}(1^\lambda)$.
- $\mathrm{H}_0 : \{0,1\}^* \to \mathrm{Z}_p^* \times \mathrm{G}^*$, $\mathrm{H}_1 : \tilde{\mathrm{G}}^{n(k+3)} \to \Theta^\eta \subseteq \mathrm{Z}_p^n$, $\mathrm{H}_2 : \{0,1\}^* \to \mathrm{Z}_p^*$.
- parse $params := (p, \mathrm{G}, \tilde{\mathrm{G}}, \mathrm{G}_T, e, g, \tilde{g}, n, \eta, \tau, k, \mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2)$.

$\mathbf{IKGen}(params, i \in [1,n]) \to (ipk_i, isk_i, \boldsymbol{RegList}_i)$

- $(isk_i, ipk_i) \leftarrow \mathbf{PSM.KGen}(params)$, where $isk_i := (x_i, y_{i,0}, y_{i,1}, \cdots, y_{i,k+1}) \in_R \mathrm{Z}_p^*$,
  $ipk_i := (\tilde{X}_i, \tilde{Y}_{i0}, \tilde{Y}_{i,1}, \cdots, \tilde{Y}_{i,k+1}) \leftarrow (\tilde{g}^{x_i}, \tilde{g}^{y_{i,0}}, \tilde{g}^{y_{i,1}}, \cdots, \tilde{g}^{y_{i,k+1}})$.
- parse $\boldsymbol{RegList}_i := \varnothing$, return $(ipk_i, isk_i, \boldsymbol{RegList}_i)$.

$\mathbf{OKGen}(params, j \in [1,\eta]) \to (opk_j, osk_j, \boldsymbol{TraceList}_j)$

- $(opk_j, osk_j) \leftarrow \mathbf{VSS.KGen}(params)$, where $(opk_j, osk_j) := (\tilde{f}_j \leftarrow \tilde{g}^{z_j}, z_j) \in \tilde{\mathrm{G}}^* \times \mathrm{Z}_p^*$
- parse $\boldsymbol{TraceList}_j := \varnothing$, return $(opk_j, osk_j, \boldsymbol{TraceList}_j)$.

$\mathbf{KAgg}(\boldsymbol{ipk} = \{ipk_i\}_{i\in[1,n]}) \to apk$

- $apk := (\tilde{X}', \tilde{Y}_0', \tilde{Y}_1', \cdots, \tilde{Y}_{k+1}') \leftarrow \mathbf{PSM.KAgg}(ipk_1, \cdots, ipk_n)$, where $\tilde{X}' \leftarrow \prod_{i=1}^n \tilde{X}_i^{t_i} = \tilde{g}^{\sum_{i=1}^n x_i t_i}$,
  $\tilde{Y}_0' \leftarrow \prod_{i=1}^n \tilde{Y}_{i0}^{t_i} = \tilde{g}^{\sum_{i=1}^n y_{i,0} t_i}$, $\tilde{Y}_j' \leftarrow \prod_{i=1}^n \tilde{Y}_{i,j}^{t_i} = \tilde{g}^{\sum_{i=1}^n y_{i,j} t_i}$, $\tilde{Y}_{k+1}' \leftarrow \prod_{i=1}^n \tilde{Y}_{i,k+1}^{t_i} = \tilde{g}^{\sum_{i=1}^n y_{i,k+1} t_i}$.
- parse $gpk := (\boldsymbol{ipk}, \boldsymbol{opk}, apk)$.

$\mathbf{CredReq}(\boldsymbol{gpk}, uid, \boldsymbol{Attr} = (a_1, \cdots, a_k), \tilde{Y}_0') \to \boldsymbol{CredReq}_{uid}$

- $(a', h) \leftarrow \mathrm{H}_0(uid, \boldsymbol{Attr})$, $H_{uid} := h^{uid}$, $G_{uid} := g^{uid}$, $\pi_{uid} \leftarrow \mathbf{NIZK.Prove}\{uid : H_{uid} \wedge G_{uid}\}$.
- $(\{\tilde{C}_j, \pi_{Pj}\}_{j\in[1,\eta]}, \{V_\ell\}_{\ell\in[1,\tau]}) \leftarrow \mathbf{VSS.Share}(\boldsymbol{opk}, uid, h, \tilde{Y}_0')$, where $\tilde{C}_j := (\tilde{C}_{j,0}, \tilde{C}_{j,1}) \leftarrow (\tilde{g}^{r_j}, \tilde{f}_j^{r_j} \tilde{Y}_0^{uid_j})$,
  $\pi_{Pj} \leftarrow \mathbf{NIZK.Prove}\{r_j : \tilde{C}_{j,0} = \tilde{g}^{r_j}, e(h, \tilde{C}_{j,1}/\tilde{f}_j^{r_j}) = e(H_{uid} \prod_{\ell=1}^t V_\ell^{\ell^j}, \tilde{Y}_0')\}$, $V_\ell := h^{p_\ell}$.
- parse $\boldsymbol{CredReq}_{uid} \leftarrow (G_{uid}, H_{uid}, \pi_{uid}, \{\tilde{C}_j, \pi_{Pj}\}_{j\in[1,\eta]}, \{V_\ell\}_{\ell\in[1,\tau]}, (a', h))$.

$\mathbf{CredGen}(isk_i, uid, \boldsymbol{Attr} = (a_1, \cdots, a_k), 1, \boldsymbol{gpk}, \boldsymbol{RegList}_i) \to (\boldsymbol{CredGen}_{i,uid}, \boldsymbol{RegList}_i)$

- abort if $(uid, \boldsymbol{Attr}, *) \in \boldsymbol{RegList}_i$ or $\mathbf{NIZK.Verf}(g, h, G_{uid}, H_{uid}, \pi_{uid}) = 0$ or
  $\mathbf{VSS.Verf}(h, \{V_\ell\}_{\ell\in[1,\tau]}, \{\tilde{C}_j, \pi_{Pj}\}_{j\in[1,\eta]}) = 0$.
- $\sigma_i := (a', \sigma_{i,1}, \sigma_{i,2})_{i\in[1,n]} = (a', h, h^{x_i + y_{i,0} \cdot uid + \sum_{j=1}^k y_{i,j} \cdot a_j + y_{i,(k+1)} \cdot a'})_{i\in[1,n]} \leftarrow \mathbf{PSM.Sign}(isk_i, (uid, \boldsymbol{Attr} = (a_1, \cdots, a_k)))$
- parse $\boldsymbol{CredGen}_{i,uid} := \sigma_i$, $\boldsymbol{RegList}_i \leftarrow \boldsymbol{RegList}_i \cup \{uid, \boldsymbol{Attr}, a'\}$.

$\mathbf{CredAgg}(apk, (uid, \boldsymbol{Attr}), \{\boldsymbol{CredGen}_{i,uid}\}_{i\in[1,n]}) \to \boldsymbol{CredAgg}_{uid}$

- $\mathbf{PSM.Verf}(ipk_i, (uid, \boldsymbol{Attr}), \boldsymbol{CredGen}_{i,uid}) \overset{?}{=} 1$
- $\sigma = (a', \sigma_1, \sigma_2) = (a', h, \prod_{i=1}^n \sigma_{i,2}^{t_i} = h^{\xi + \sum_{j=1}^k u_j a_j + u_{k+1} a'}) \leftarrow \mathbf{PSM.SAgg}(apk, (uid, \boldsymbol{Attr}), \{\sigma_i\}_{i\in[1,n]})$, where
  $\xi := \sum_{i=1}^n (x_i + y_{i,0} \cdot uid) t_i$, $u_j = (\sum_{i=1}^n y_{i,j} t_i)_{j\in[1,k]}$, $u_{k+1} = \sum_{i=1}^n y_{i(k+1)} t_i$.
- $\mathbf{PSM.Verf}(apk, (uid, \boldsymbol{Attr}), \boldsymbol{CredAgg}_{uid}) \overset{?}{=} 1$, namely checks whether $\sigma_1 \neq 1_{\mathrm{G}}$ and
  $e(\sigma_1, \tilde{X}' \tilde{Y}_0'^{uid} \prod_{i=1}^k \tilde{Y}_j'^{a_j} \tilde{Y}_{k+1}'^{a'}) = e(\sigma_2, \tilde{g})$.
- return $\boldsymbol{CredAgg}_{uid} := (uid, \boldsymbol{Attr}, \sigma, e(\sigma_1, \tilde{Y}_0'), e(\sigma_1, \tilde{Y}_{k+1}'))$.

$\mathbf{CredProve}(\boldsymbol{ipk}, cred_{uid}, \rho \subseteq \boldsymbol{Attr}) \to \boldsymbol{Cred}_{uid,\rho}$

- choose $r \in_R \mathrm{Z}_p^*$, compute $(\sigma_1', \sigma_2') \leftarrow (\sigma_1^r, \sigma_2^r)$, such that $e(\sigma_1', \tilde{X}' \tilde{Y}_0'^{uid} \prod_{i=1}^k \tilde{Y}_j'^{a_j} \tilde{Y}_{k+1}'^{a'}) = e(\sigma_2', \tilde{g})$.
- compute $\pi_\rho \leftarrow \mathbf{NIZK.Prove}\{(uid, a') : \mathbf{PSM.Verf}(apk, uid, (a', \sigma_1', \sigma_2') = 1)\}(\rho)$ .Detailed steps are
  following: choose $s_{uid}, s_{a'} \in_R \mathrm{Z}_p^*$, compute $u \leftarrow e(\sigma_1'^{s_{uid}}, \tilde{Y}_0') e(\sigma_1'^{s_{a'}}, \tilde{Y}_{k+1}')$, compute the challenge
  $c \leftarrow \mathrm{H}_2(u, \boldsymbol{Attr}, \rho, \sigma_1', \sigma_2', \boldsymbol{ipk}) \in \mathrm{Z}_p$ and response $\boldsymbol{v} := (v_{uid}, v_{a'}) \leftarrow (s_{uid} - c \cdot uid, s_{a'} - ca')$, set
  $\pi_\rho := (c, \boldsymbol{v}) \leftarrow (c, v_{uid}, v_{a'}) \in_R \mathrm{Z}_p^3$.
- parse $\boldsymbol{Cred}_{uid,\rho} \leftarrow (\sigma_1', \sigma_2', \pi_\rho)$.

$\mathbf{CredVerify}(apk, \rho, \boldsymbol{Attr}, \boldsymbol{Cred}_{uid,\rho}) \to b \in (0,1)$

- parse $\boldsymbol{Cred}_{uid,\rho} \leftarrow (\sigma_1, \sigma_2, \pi_\rho)$, $\pi_\rho := (c', v_{uid}, v_{a'})$
- check if $\mathbf{NIZK.Verf}(gpk, \sigma_1, \sigma_2, \rho, \boldsymbol{Attr}, \pi_\rho) = 1$
- compute $u' = e(\sigma_1'^{v_{uid}}, \tilde{Y}_0') e(\sigma_1'^{v_{a'}}, \tilde{Y}_{k+1}') e(\sigma_2'^{c}, \tilde{g}) e(\sigma_1'^{c}, \tilde{X}'^{-1} \prod_{j=1}^k \tilde{Y}_j'^{a_j})$
- reture1 if both $\sigma_1 \neq 1_{\mathrm{G}}$ and $c = \mathrm{H}_2(u', \boldsymbol{Attr}, \rho, \sigma_1, \sigma_2, apk)$

$\mathbf{Trace}((\boldsymbol{TraceList}_j, osk_j, \mathrm{O}, gpk, \rho, \boldsymbol{Attr}, \boldsymbol{Cred}_{uid,\rho})) \to \{uid / \perp\}$:

- returns $\perp$ if $\mathbf{CredVerify}(apk, \rho, \boldsymbol{Attr}, \boldsymbol{Cred}_{uid,\rho}) = 0$
- set $\boldsymbol{TraceList}_j[uid] \leftarrow \tilde{C}_j^* = \tilde{C}_{j,1} / (\tilde{C}_{j,0}^{z_j}) = \tilde{Y}_0^{uid_j}$
- for all $\boldsymbol{TraceList}_j[uid] \neq \perp$, compute $T_{uid,j} := e(\sigma_1, \tilde{C}_{j,1} / (\tilde{C}_{j,0}^{z_j})) = e(\sigma_1, \tilde{Y}_0^{uid_j})$
- broadcasts $T_j := \{(uid, T_{uid,j})\}_{uid:\ \boldsymbol{TraceList}_j[uid] \neq \perp}$ to all $\mathrm{O}_j \in \mathrm{O}$.
- upon receiving T_q from other RGs O_q （$\mathrm{O}_q \in \mathrm{O} \setminus \{\mathrm{O}_j\}$）

  -For all $\mathrm{O}_q \in \mathrm{O}$, compute Lagrange coefficient $w_q \leftarrow \prod_{q\in\mathrm{O}\setminus\{j\}} q/(q-j)$, and $\prod_{\mathrm{O}_j\in\mathrm{O}} (\tilde{C}_j^*)^{w_j} = \tilde{Y}_0^{uid}$.

  -For all $(uid, T_{uid,j}) \in T_j$ (in lexicographic order of user identities), if $\exists \mathrm{O}_q \in \mathrm{O} : (uid, *) \notin T_j$ proceed the following operations: firstly compute $(a', h) \leftarrow \mathrm{H}_0(uid, \boldsymbol{Attr})$, then for all $\mathrm{O}_q \in \mathrm{O} \setminus \{\mathrm{O}_j\}$, retrieve $(uid, T_{uid,j})$ from T_q , and finally return uid if $e(\sigma_1, \tilde{X}' \tilde{Y}_{k+1}'^{a'} \prod_{j=1}^k \tilde{Y}_j'^{a_j}) \prod_{q\in\mathrm{O}} T_{uid,q}^{w_q} = e(\sigma_2, \tilde{g})$

Fig. 3 The details of distributed identity management

Table 3 Notations

Symbols	Descriptions
λ	Security parameter
$\mathcal{H}$	Hash function : $\{0,1\}^* \rightarrow \mathbb{G}$
$\mathcal{H}_0$	Hash function : $\mathbb{Z}_p^k \rightarrow \mathbb{Z}_p^* \times \mathbb{G}^*$
$\mathcal{H}_1$	Hash function : $\tilde{\mathbb{G}}^{n(k+3)} \rightarrow \Theta^n \subseteq \mathbb{Z}_p^n$
e	Bilinear groups
$\mathbb{G},\tilde{\mathbb{G}},\mathbb{G}_T$	Cyclic groups of order p
$g,\hat{g},\tilde{g}$	Generators:$g,\hat{g} \in_R \mathbb{G}^*,\tilde{g} \in_R \tilde{\mathbb{G}}$
$\mathcal{I}\Theta = \{I_i\}_{i\in[1,n]}$	The set of n DOs,$n = \lvert\mathcal{I}\rvert$
$\mathcal{S}\Theta = \{S_j\}_{j\in[1,\eta]}$	The set of η RGs,$\eta = \lvert\mathcal{S}\rvert$
$\mathcal{U}\Theta = \{U_j\}_{j\in[1,q]}$	The set of q DUs,$q = \lvert\mathcal{U}\rvert$
$\mathcal{O}\Theta = \{\mathcal{O}\Theta_i\}_{i\in[1,\tau+1]}$	The set of $\tau+1$ RGs,$\tau+1 = \lvert\mathcal{O}\rvert$

from CMSC, and updates the trace list $\boldsymbol{TraceList}_j[uid] \leftarrow \widetilde{C}_j^* = \widetilde{C}_{j,1}/(\widetilde{C}_{j,0}^{z_j}) = \widetilde{Y}_0^{uid_j}$, where uid_j is the share of $\mathcal{O}_j \in \mathcal{O}$ on secret identity uid. Next, $\mathcal{O}_i \in \mathcal{O}$ broadcasts recovered share of his own and collects shares of other RGs $\mathcal{O}_q \in \mathcal{O}$ and finally calls the algorithm **VSS.Recover** to recover uid. If no such identity is found, open the algorithm to return $\perp$.

When RGs judged that U_{uid} needs to be revoked, RGs will submit the revocation update information to CMSC which is in charge of maintaining the revocation registry and revoking corresponding credentials.

Verifiable data trading

The verifiable data trading scheme mainly includes initialization, dataset encryption and upload, dataset validation and registration, and the trading stage. Furthermore, the trading stage consists of data trading request, data trading response, key verification and decryption, as well as dispute.

Initialization

In this phase, the public parameters $\Gamma := (p,\mathbb{G},\mathbb{G}_T,e_1,\hat{g},\mathcal{H}) \leftarrow \textbf{MS.Setup}(1^\lambda)$ and $pp := (p,\mathbb{G},\tilde{\mathbb{G}},\mathbb{G}_T,e_2,g,\tilde{g}) \leftarrow \textbf{PCE.Setup}(1^\lambda)$ are first generated. Then, DO $I_i \in \mathcal{I}$ generates the key pair $(vsk_i,vpk_i) \leftarrow \textbf{MS.KGen}(\Gamma)$ (where $(vsk_i,vpk_i) \leftarrow (v_i \in_R \mathbb{Z}_p^*, g^{v_i} \in \mathbb{G})$) for dataset upload and data integrity verification and $Ks \leftarrow \textbf{AES.KGen}(1^\lambda)$ for the AES encryption algorithm(the data owner can generate the AES encryption key through negotiation or by selecting a representative). Meanwhile,U_{uid} generates the key pair $(psk,ppk) := (x \in_R \mathbb{Z}_p^*, g^x \in \mathbb{G}) \leftarrow \textbf{PCE.KGen}(pp)$ for the PCE algorithm.

Dataset upload and registration

In the data upload phase, DOs encrypt and upload the dataset, and submit the data upload smart contract **UploadData** to DTSC. Firstly, DOs select the dataset $DATA = \{m_1 \cdots m_\ell\}$ $(m_j \in \mathbb{Z}_p^* j \in [1,\ell])$ and encrypt $DATA$ using the AES encryption algorithm to generate $DATA^* = \{\textbf{Enc}_{Ks}(m_j)\}_{j\in[1,\ell]}$. Then, each $I_i \in \mathcal{I}$ runs function $\psi_i \leftarrow \textbf{MS.Sign}(m_j,mid_j,vsk_i)$ to generate the local signature $\psi_i = \{\psi_{j,i}\}_{j\in[1,\ell]} = \{[\mathcal{H}(mid_j)\hat{g}^{m_j}]^{v_i}\}_{j\in[1,\ell]}$ (where mid_j is the identifier of block m_j) and broadcasts it in $\mathcal{I}$. Next, DOs generate properties of the dataset, including the unique identifier did, the index hash $Index := Hash(DATA \parallel did)$, the dataset description $ProData$ (including file name, file size, keywords, data samples, underlying data model, and other attributes), and access policy $Policy$. Finally, DOs upload $File = \{DATA^*, \psi_i, Index, ProData, Policy, did\}$ to CS and submit the contract **UploadData** (as shown in Fig. 4) to DTSC.

After receiving $File$, CS verifies the local signatures $\{\psi_{j,i}\}_{j\in[1,\ell]\, i\in[1,n]}$ in turn and computes the aggregate signature of the data block $\psi_j \leftarrow \textbf{MS.SAgg}(\{vpk_i\}_{i=1}^n, \{\psi_{j,i}\}_{i=1}^n)$. Then, CS stores data and generates the file storage address Loc_f. Finally, CS submits the data registered smart contract **RegistrData** (as shown in Fig. 5) to DTSC. The

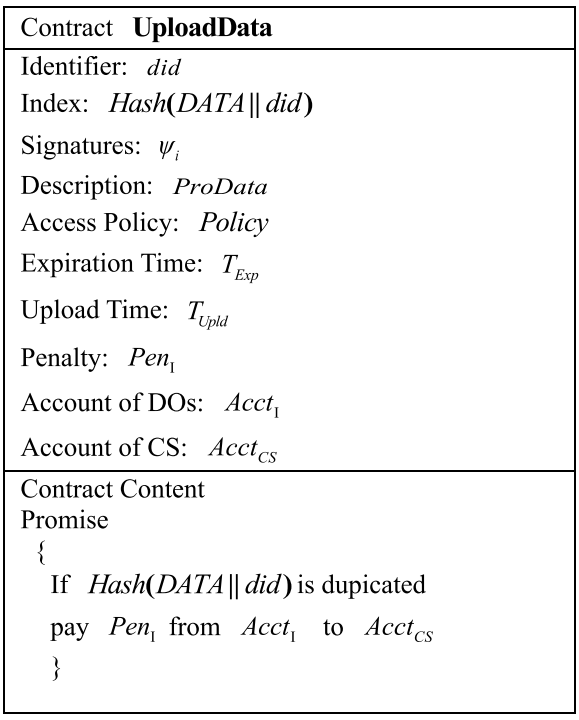

Fig. 4 Data upload smart contract

Contract **RegistrData**
Identifier: did Index: $Hash(DATA \| did)$ Description: $ProData$ Signatures: ψ_j Location of Dataset: Loc_f Receiving Time: T_{Rec} Expiration Time: T_{Exp} Access Policy: $Policy$ Penalty: Pen_{CS} Account of DOs: $Acct_{\mathrm{I}}$ Account of CS: $Acct_{CS}$

Contract Content
Promise
{
If data integrity verification is false
pay Pen_{CS} from $Acct_{CS}$ to $Acct_{\mathrm{I}}$
}

Fig. 5 Data registered smart contract

Contract **RequestData**
Identifier: did Index: $Hash(DATA \| did)$ Description: $ProData$ Request Time: T_{Req} Payment due date: T_{pay} Trading Charges: Fee_{Trd} Account of DOs: $Acct_{\mathrm{I}}$ Account of DU: $Acct_{uid}$ Public Key of PCE: ppk_{uid} Credential: $\boldsymbol{Cred}_{uid,\rho}$ Access Information: $AccInfo$

Contract Content
Promise
{
If （ $\mathbf{PCE.Dec}(psk, c_{Ks}) \neq \perp$ &
$\mathbf{PCE.Check}(Ks', ppk_{uid}, c_{Ks}) == 1$ & data integrity verification is true ）
pay Fee_{Trd} from $Acct_{uid}$ to $Acct_{\mathrm{I}}$ by T_{pay}
}

Fig. 6 Trading request smart contract

contract **RegistrData** means that *File* has been received by CS.

Trading

The data trading stage mainly includes data trading request, data trading response, data integrity verification, key verification and decryption as well as disputes,

1) Data trading request
   U_{uid} retrieves the needed data through the description *ProData* and submits the smart contract **RequestData** to DTSC. As shown in Fig. 6, **RequestData** contains basic data information, trade information and contract code. In addition, U_{uid} needs to provide his PCE public key ppk_{uid}, credentials $\boldsymbol{Cred}_{uid,\rho}$ and access information *AccInfo*. The contract code stipulates that the data owner must provide correct PCE ciphertext c_{Ks} and data integrity verification results. Apparently, **RequestData** guarantees that U_{uid} will pay the data trade fee Fee_{Trd} to DOs before the time T_{pay} if U_{uid} can correctly decrypt the dataset.
2) Data integrity verification
   In the data integrity verification process, blockchain sends challenge information to CS. CS calculates evidence based on the challenge information and returns the evidence.
   In the challenge phase, blockchain generates ℓ random numbers $\delta_1 \cdots \delta_\ell \in_R \mathbb{Z}_p^*$ and sends the challenge request $\mathrm{chal} = (j, \delta_j)_{j\in[1,\ell]}$ to CS. In the evidence generation phase, CS calculates $\mu = \sum_{j=1}^{\ell} \delta_j m_j \in \mathbb{Z}_p^*$ and $\upsilon = \prod_{j=1}^{\ell} \psi_j^{\delta_j} \in \mathbb{G}$ after receiving the challenge message, and returns the verification evidence $(\mu, \upsilon, \{mid_1 \cdots mid_\ell\})$ to blockchain. In the data verification phase, after blockchain receives the verification evidence $(\mu, \upsilon, \{mid_1 \cdots mid_\ell\})$ and the public key sequence $\{vpk_1 \cdots vpk_n\}$ of all data owners, calculates $pk_s = \prod_{i=1}^{n} vpk_i$ and verifies whether the equation $e(\upsilon, \hat{g}) = e(\prod_{j=1}^{\ell} \mathcal{H}(mid_j)^{\delta_j} \cdot \hat{g}^{\mu}, pk_s)$ is true. Finally, DTSC sends data integrity verification result to DOs and DU.
3) Data trading response
   Before the data trading response, the validity of dataset and data user identity need to be checked. Firstly, DTSC checks whether there is the same hash index *Index* of the dataset on the chain and checks the data validity period. Then, DTSC initi-

Contract **SellData**
Identifier: did Index: $Hash(DATA \| did)$ Description: $ProData$ Sell Time: T_{Sell} Payment due date: T_{pay} Dispute closing time: T_{close} Trading charges: Fee_{Trd} Ciphertext of PCE: c_{Ks} Account of DOs: $Acct_1$ Account of DU: $Acct_{uid}$

Contract Content
Promise
{
If (**DEC**$(Ks', DATA^*) == \perp$ ‖
PCE.Check$(Ks', ppk_{uid}, c_{Ks}) == 0$ ‖
pay 0 from $Acct_{uid}$ to $Acct_1$ by T_{close}
else pay Fee_{Trd} from $Acct_{uid}$ to $Acct_1$ by T_{pay}
}

Fig. 7 Trading response smart contract

ates the data integrity verification request and CS returns the data integrity verification result. If there are other duplicate indexes or the dataset expires or data integrity verification failed, DTSC will reject the request and notify DOs. Meanwhile, CMSC will check the revocation registry and will call the function **CredVerify** to verify the validity of the credentials $\boldsymbol{Cred}_{uid,\rho}$, and reject the data request of U_{uid} if the credential is revoked or illegal. Then DTSC will check the access control policy, and will reject the request if it does not match.

After all checks are completed, DOs will respond to the request and submit the data sell contract **SellData** to DTSC. As shown in Fig. 7, **SellData** provides the correct ciphertext of Ks, i.e. $c_{Ks} \leftarrow$ **PCE.Enc**(ppk_{uid}, Ks), and meanwhile promises that U_{uid} is allowed to deny the validity of c_{Ks} and initiate dispute handling before the dispute closing time T_{close}. The contracts **RequestData** and **SellData** ensure that the fair trading between DOs and U_{uid}, that is, honest DOs get paid, and honest U_{uid} gets the correct data.

4) Key verification and decryption
After receiving the PCE ciphertext, U_{uid} first recovers the AES key Ks' through $K_S' \leftarrow$ **PCE.Dec**(psk_{uid}, c_{Ks}), and checks whether Ks' is the correct decryption of c_{Ks}(i.e. **PCE.Check**$(Ks', ppk_{uid}, c_{Ks}) \stackrel{?}{=} 1$). Then, U_{uid} downloads and decrypts the ciphertext of the dataset $DATA^*$ from CS to obtain the original data (i.e. $DATA' \leftarrow$ **DEC**$(Ks' DATA^*)$). Next, U_{uid} calculates the index value of $DATA'$ and checks whether it is consistent with the index value stored on the platform (i.e. $Hash(DATA' \parallel did) \stackrel{?}{=} Hash(DATA \parallel did)$).
If the above checks pass, U_{uid} will pay Fee_{Trd} before D_{pay}. Otherwise, if Ks is incorrect decryption (i.e. **PCE.Check**$(Ks', ppk_{uid}, c_{Ks}) == 0$) or $DATA^*$ is incorrect decryption (i.e.**DEC**$(Ks' DATA^* == \perp)$), U_{uid} will refuse to pay within T_{pay} and initiate the trading dispute before the trading closed time T_{close}.
5) Dispute
In the dispute stage, Fee_{Trd} will continue to be locked until the evidence is confirmed. U_{uid} invokes the **Dispute** contract and submits the evidence, including $Hash(DATA' \parallel did)$,$Ks'$ and signature ψ_j. If DTSC verifies Ks successfully but fails to decrypt the ciphertext correctly (i.e. **PCE.Check**$(Ks', ppk_{uid}, c_{Ks}) == 1$ and $Hash(DATA' \parallel did) \neq Hash(DATA \parallel did)$), meanwhile the signature verification is passed **MS.Verify**$(m, mid, \psi_j, \{vpk_1 \cdots vpk_n\}) = 1$, the trading fails and the locked Fee_{Trd} will not be paid to the data owner. If anyone is not satisfied, it means the trading is successful and the locked Fee_{Trd} will continue to be paid to the data owner (Fig. 8).

Analysis

Security analysis

Identity security

In this scheme, user authentication is realized by fine-grained verifiable credentials with selective disclosure, which meet the anonymity, threshold traceability, and multi-show unlinkability. Our definition follows the security concept of dynamic group signature in [35] and the security definition of anonymous credentials in [8].

1) Anonymity

Anonymity ensures that U_{uid} prove valid credential without revealing its true identity.
In the credential request stage, U_{uid} generates the secret identity $H_{uid} := h^{uid}$ as well as zero-knowledge proof π_{uid}. Apparently, it is unfeasible for an adversarial to extract uid from H_{uid}. In the credential generation stage, each $I_i \in \mathcal{I}$

Contract **Dispute**
Identifier: did
Index: $Hash(DATA' \| did)$
Signatures: ψ_j
Restored decryption key: Ks'
Ciphertext of PCE: c_{Ks}
Public Key of PCE: ppk_{uid}
Trading Charges: Fee_{Trd}
Complaint Time: $T_{Complaint}$
Dispute closing time: T_{close}
Account of dos: $Acct_1$
Account of DU: $Acct_{uid}$

Contract Content
Promise
{
If （ **PCE.Check**$(Ks', ppk_{uid}, c_{Ks}) == 1$ &
$Hash(DATA' \| did) \neq Hash(DATA \| did)$ &
MS.Verify$(m, mid, \psi_j, \{vpk_1, \cdots, vpk_n\}) = 1$ ）
pay 0 from $Acct_{uid}$ to $Acct_1$ by D_{close}
else
pay Fee_{Trd} from $Acct_{uid}$ to $Acct_1$ by T_{close}
return $\perp$
}

Fig. 8 Dispute smart contract

verifies the proof π_{uid} and then blindly signs H_{uid} with isk_i, and generates PS multi-message signature $\sigma_{i,2} = h^{x_i + y_{i,0} \cdot uid + \sum_{j=1}^{k} y_{i,j} \cdot a_j + y_{i,(k+1)} \cdot a'}$. Due to zero-knowledge proof and blind issuance, I_i is unfeasible to learn the true identity of π_{uid}. In the credential presentation stage, U_{uid} generates $\boldsymbol{Cred}_{uid,\rho} \leftarrow (\sigma_1', \sigma_2', \pi_\rho)$ randomize $(\sigma_1', \sigma_2') \leftarrow (\sigma_1^r, \sigma_2^r)$ and computes zero-knowledge proof π_ρ. Due to zero-knowledge proof and re-randomness of PS signature, CS can convince the validation of $\boldsymbol{Cred}_{uid,\rho}$ through **CredVerify**, but is unfeasible to infer any information about uid. Only $\tau+1$ RGs are capable of recovering identity of U_{uid} through threshold trace operation. Therefore, anonymity is guaranteed and the identity information U_{uid} is unfeasible to be disclosed as long as $I_i \in \mathcal{I}$ is honest and the presenting credential is not opened.

2) Threshold traceability
Threshold traceability refers to that $\tau + 1$ RGs can recover the identity of valid credential $\boldsymbol{Cred}_{uid,\rho}$ but any τ corrupt RGs are incapable. Essentially, the tinformation of threshold traceability originates from the secret share of identity during the issuance process. Specially, U_{uid} shared $\widetilde{C}_j := \left(\widetilde{C}_{j,0}, \widetilde{C}_{j,1}\right) \leftarrow \left(\tilde{g}^{r_j} \tilde{f}_j^{r_j} \tilde{Y}_0^{'uid_j}\right)$(i.e. the ciphertext of the share uid_j about his secret identity uid) to all $S_j \in \mathcal{S}$, where $\tilde{f}_j^{r_j} \tilde{Y}_0^{'uid_j}$ is the ElGamal ciphertext of uid_j under the trace public key $opk_j = \tilde{f}_j$ of S_j. $\tilde{C}_j$ means that the key $\tilde{Y}_0^{'uid}$ is secretly shared between all RGs. In the issuance, I_i blindly signs to secret identity H_{uid} after $\tilde{C}_j$ is validated through **VSS.Verf**. The aggregated credential contains the signature $\sigma_2 = h^{x' + y_0' \cdot uid + \sum_{j=1}^{k} y_j' \cdot a_j + y_{k+1}' \cdot a'}$ that includes the information of $\tilde{Y}_0^{'uid}$. In the trace stage, $\tau + 1$ RGs compute the ciphertext of share about the secret identity of uid using trace secret key $osk_j = z_j$(i.e.$\tilde{C}_j^* = \tilde{C}_{j,1} / (\tilde{C}_{j,0}^{z_j}) = \tilde{Y}_0^{'uid_j}$) and recover the identity uid of U_{uid}.

3) Blind issuance
In the credential generation stage, each $I_i \in \mathcal{I}$ firstly verifies the proof π_{uid} and then blindly signs the user's secret identity $H_{uid} := h^{uid}$ with isk_i, and finally generates signature $\sigma_{i,2} = h^{x_i + y_{i,0} \cdot uid + \sum_{j=1}^{k} y_{i,j} \cdot a_j + y_{i,k+1} \cdot a'}$. Due to zero-knowledge proof and blind issuance, I_i is unfeasible to learn the true identity of π_{uid}

4) Multi-show unlinkability
It mainly stems from the re-randomness of the PS signature (namely, when no one knows the secret key knowledge, the signature of the same message can be randomized into another unlinkable signature version). As shown in the presented credential $\boldsymbol{Cred}_{uid,\rho} = \left(\sigma_1^r, \sigma_2^r, \pi_\rho\right)$ (where $r \in_R \mathbb{Z}_p^*$), the newly $\boldsymbol{Cred}_{uid,\rho}$ varies with r and is unlinkable with previous ones.

5) Selective disclosure
When credential $\boldsymbol{Cred}_{uid,\rho} \leftarrow (\sigma_1', \sigma_2', \pi_\rho)$ is presented, CS needs to check the validation of (σ_1', σ_2') through **PSM.Verf**$(apk, uid, (a', \sigma_1', \sigma_2')$ and the validation of π_ρ through **NIZK.Verf**$(gpk, \sigma_1, \sigma_2, \rho, \boldsymbol{Attr}, \pi_\rho)$. Actually, the nature of the process **NIZK.Verf** is checking $c \leftarrow \mathcal{H}_2(u, \boldsymbol{Attr}, \rho, \sigma_1', \sigma_2', \boldsymbol{ipk}) \in \mathbb{Z}_p$, which is the ciphertext of the selective presentation attribute subset $\rho \subseteq \boldsymbol{Attr}$. Ultimately, CS only obtained the cryptographic verification result of the attribute ρ instead of the plaintext of the attribute.

Data security

1) Data confidentiality
The dataset is encrypted by symmetric encryption (i.e. $DATA^* = \{\mathbf{Enc}_{Ks}(m_j)\}_{j \in [1,k]}$) and the decryption key Ks is encrypted by PCE encryption (i.e.$c_{Ks} \leftarrow$ **PCE.Enc**(ppk_{uid}, Ks). Meanwhile, only DUs who hold the unrevoked valid credential and

match the specified access policy are allowed to obtain *Ks*.

2) Data integrity & public verifiability
We employed DTSC to replace traditional TPA to prevent collusion attacks with malicious third parties due to the public verifiability of the multi-owner data integrity verification scheme. Therefore, the proposal inherits the merits of the multi-owner data integrity verification scheme.
3) Data authenticity
Data unforgeability is guaranteed because the original data ciphertext adopts multi-signature technology, which can prevent attackers from forging the owner's data. Meanwhile, both multi-signature technology and smart contract also ensures the undeniable ownership of data.

Trading security

1) Trade fairness
The smart contract **RequestData**, **SellData** and **Dispute** guarantees the fair trading between the DOs and DU. Specially, **RequestData** guarantees that U_{uid} will pay Fee_{Trd} to DOs before D_{pay} if U_{uid} correctly decrypts $DATA^*$, **SellData** provides the correct ciphertext c_{Ks} of the data decryption key *Ks*, and promises that U_{uid} is allowed to deny the validity of c_{Ks} and initiate a dispute before dispute closing time D_{close}. **Dispute** guarantee that U_{uid} will refuse to pay within D_{pay} and initiate the trading dispute before D_{close} when *Ks* is incorrectly decrypted (i.e. $\mathbf{PCE.Check}(Ks', ppk_{uid}, c_{Ks}) == 0$) or $DATA^*$ is incorrectly decrypted (i.e.$\mathbf{DEC}(Ks' DATA^*) == \perp$).
2) Trade non-repudiation
The operations of entities in the trading process are recorded on the blockchain and the immutability of blockchain prevents the repudiation of participating entities. Moreover, the proposal is capable of resisting the forged decryption key attack and forged signature attack during the dispute, thus preventing malicious DUs from skipping out on the bill. Firstly, the decryption key *Ks* is encrypted by the PCE algorithm, so the forged key $\tilde{Ks}$ cannot be verified (i.e. $\mathbf{PCE.Check}(\tilde{Ks}, ppk_{uid}, c_{Ks}) == 0$), namely, the forged key $\tilde{Ks}$ cannot be used as evidence of dispute. Secondly, it is impossible to forge the signature as evidence due to the unforgery of the multi-signature [7, 22].
3) Anti-second-reselling-attack
DTSC records the index hash $Index := Hash(DATA \parallel did)$ and the datasets with duplicate indexes are deemed illegal. Meanwhile, RGs are liable to tracing malicious DUs or DOs and thus prevent privately trading data off the chain to a certain extent due to the immutability and traceability of blockchain [7].

Table 4 The complexity analysis of distributed identity management

Algorithm	Computational cost	Time cost
IKGen	$(k+3)n\boldsymbol{E}_2$	55.60 ms
OKGen	$\eta\boldsymbol{E}_2$	13.90 ms
KAgg	$(k+2)(n\boldsymbol{E}_2+(n-1)\boldsymbol{M}_2)$	41.70 ms
CredReq	$(\tau+2)\boldsymbol{E}_1+3\eta\boldsymbol{E}_2+\eta\boldsymbol{M}_2$	49.25 ms
CredGen	$n((k+3)\boldsymbol{E}_1+(k+2)\boldsymbol{M}_1)$	30.20 ms
CredAgg	$(n+1)(2\boldsymbol{P}+(k+2)\boldsymbol{E}_2+(k+2)\boldsymbol{M}_2)+n\boldsymbol{E}_1$	714.83 ms
CredProve	$1\boldsymbol{F}+2\boldsymbol{P}+4\boldsymbol{E}_1$	115.73 ms
CredVerify	$3\boldsymbol{F}+4\boldsymbol{P}+4\boldsymbol{E}_1+(k+2)\boldsymbol{E}_2+(k+2)\boldsymbol{M}_1$	233.91 ms
Trace	$\tau\boldsymbol{E}_2+(\tau-1)\boldsymbol{M}_2$	8.34 ms

Performance evaluation

We implemented the off-chain cryptographic operations overheads using the JPBC cryptography Library on the computer with a 2.80 GHz processor and 16 GB memory. The computational cost and the execution time of distributed identity management (set k=1, η=5, n=5, τ=3) are shown in Table 4, where the exponentiation operation and multiplication operation in $\mathbb{G}$ and $\tilde{\mathbb{G}}$ are denoted by $\boldsymbol{E}_1$,$\boldsymbol{E}_2$,$\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ respectively. In the $\mathbb{G}_T$ group, the pairing operation, exponentiation operation and multiplication operation are denoted by $\boldsymbol{P}, \boldsymbol{F}$ and $\boldsymbol{T}$ respectively. The execution time of each operation of Type F in JPBC is $T_{\boldsymbol{E}_1}$=1.51ms, $T_{\boldsymbol{E}_2}$= 2.78ms, $T_{\boldsymbol{M}_1}$ = 0.01ms, $T_{\boldsymbol{M}_2}$ = 0.02ms, $T_{\boldsymbol{P}}$ = 54.77ms, $T_{\boldsymbol{F}}$ = 12.01ms, $T_{\boldsymbol{T}}$ = 0.15ms. We omitted the cost of hash. The cost of **IKGen** and **CredGen** for each DO is $(k+3)\boldsymbol{E}_2$ and $(k+3)\boldsymbol{E}_1+(k+2)\boldsymbol{M}_1$ respectively, which both grow linearly with k. The cost of **OKGen** and Trace for each RG is $\boldsymbol{E}_2$ and $\tau\boldsymbol{E}_2+(\tau-1)\boldsymbol{M}_2$ respectively. The cost of **KAgg** and **CredAgg** for each CMSC is $(k+2)(n\boldsymbol{E}_2+(n-1)\boldsymbol{M}_2)$ and $(n+1)(2\boldsymbol{P}+(k+2)\boldsymbol{E}_2+(k+2)\boldsymbol{M}_2)+n\boldsymbol{E}_1$ respectively, which are both grows linearly with k and η. The cost of aggregation operation is $n\boldsymbol{E}_1$ which also grows linearly with n and the verification operation for each partial signature and aggregate signature is $2\boldsymbol{P}+(k+2)\boldsymbol{E}_2+(k+2)\boldsymbol{M}_2$ which is grows linearly with k. The cost of **CredReq** for DU is $(\tau+2)\boldsymbol{E}_1+3\eta\boldsymbol{E}_2+\eta\boldsymbol{M}_2$, which mainly stems from **VSS.Share** algorithm. The cost of **CredProve** for DU is constant $1\boldsymbol{F}+2\boldsymbol{P}+4\boldsymbol{E}_1$ and the cost of **CredVerify** is $3\boldsymbol{F}+4\boldsymbol{P}+4\boldsymbol{E}_1+(k+2)\boldsymbol{E}_2+(k+2)\boldsymbol{M}_1$, which is grows linearly with k. It is interesting to note that both the cost of attribute display and verification are $O(1)$ due to only one aggregate credential being involved in the display and validation of attributes. Moreover, the partial credentials and aggregate credentials are both composed of two group elements. Therefore, distributed identity management has an

advantage in computation efficiency and is suited for practical applications.

Regarding the PCE algorithm, the ciphertext length contains 4 group elements in $\mathbb{G}$, and the cost of key generation, encryption, decryption and verification is $\boldsymbol{E}_1$,$2\boldsymbol{E}_1+2\boldsymbol{E}_2$, $2P+\boldsymbol{E}_1+\boldsymbol{M}_1$ and $4\mathrm{P}+\boldsymbol{E}_1+\boldsymbol{M}_1$. Accordingly, the execution time is 1.51 ms, 8.58 ms,111.06 ms, and 220.60 ms, respectively. In the data integrity verification phase, the cost of signature, evidence generation, and verification is about $2n\mathbf{E}$, $\ell\mathrm{E}$ and $\ell\mathrm{E}+2P$ respectively. Accordingly, the the execution time is 15.1 ms, 34.16 ms, and 143.7msrespectively ($T_{\boldsymbol{E}}$=8.54ms, $T_{\boldsymbol{P}}=5.29$ms and set n=5, $\ell=4$).

The smart contract needs to perform cryptographic operations, including the credentials key aggregation, credentials aggregation, integrity verification, and PCE Check. Every OPcode in the Ethereum Virtual Machine (EVM) specification has an associated gas fee [12]. The gas cost can be estimated by the precompiled contract that implements ate pairing check on the elliptic curve alt-bn128, namely, the gas cost of ECAdd, ECMul, and ECParingare 40000gas, 500gas, and 80,000*k+100,000, respecivly.

In the data trading stage, large files are stored off-chain, and the hash index is stored on the blockchain, which results in a total 512-bit storage cost. That is, the storage cost is very low.

Conclusion

In this article, we presented a distributed industrial data trading architecture model for multiple data owners scenarios based on blockchain and cloud. In the framework, a fair trading mechanism for industrial datasets without a trusted third party is designed based on smart contracts and a public verifiable data integrity scheme for multi-data ownership is used to ensure the integrity of data. Additional, verifiable credentials with selective disclosure and threshold tracking provide privacy-enhanced authentication and malicious user tracking. Analysis results show that our proposal has successfully achieved the security goals.

Abbreviations

CS	Cloud server
BC	Blockchain
DOs	Data owners
DUs	Data users
RGs	Regulators
TPA	Third-Party Auditor
CMSC	Credential management smart contract
DTSC	Data trading smart contract
PCE	Plaintext checkable encryption

Acknowledgements

We sincerely thank the Reviewers and the Editor for their valuable suggestions.

Authors' contributions

Fang and Feng proposed the scheme and wrote the main manuscript text. Guo, Ma and Lu gave important advices with the respect to the construction, analysis, and writing. All authors reviewed and approved the final manuscript.

Funding

This work is supported by National Natural Science Foundation of China (Grant No. 61762039), Natural Science Foundation of Gansu Province, China (23YFGA0060, Grant No.20JR5RA467), Science and Technology project of Shaanxi Province, China (Grant No 2020GY-041).

Availability of data and materials

The data used to support the findings of this study is available from the corresponding author upon request.

Declarations

Competing interests

The authors declare no competing interests.

Received: 26 March 2023 Accepted: 11 October 2023
Published online: 02 February 2024

References

1. Li JQ, Yu FR, Deng GQ, Luo CW, Ming Z, Yan Q (2017) Industrial internet: a survey on the enabling technologies, applications, and challenges. IEEE Commun Surv Tutor 19(3):1504–1526. https://doi.org/10.1109/comst.2017.2691349
2. Garrido GM, Sedlmeir J, Uludag O, Alaoui IS, Luckow A, Matthes F (2022) Revealing the landscape of privacy-enhancing technologies in the context of data markets for the IoT: a systematic literature review. J Netw Comput Appl 207:29. https://doi.org/10.1016/j.jnca.2022.103465
3. Li Y, Zhu L, Wang H, Yu FR, Tang T, Zhang D (2023) Joint security and resources allocation scheme design in edge intelligence enabled CBTCs: a two-level game theoretic approach. IEEE Trans Intell Transp Syst 1–14. https://doi.org/10.1109/TITS.2023.3294546
4. Zhu L, Shen C, Wang X, Liang H, Wang H, Tang T (2023) A learning based intelligent train regulation method with dynamic prediction for the metro passenger flow. IEEE Trans Intell Transp Syst 24(4):3935–3948. https://doi.org/10.1109/TITS.2022.3231838
5. Feng J, Liu L, Hou X, Pei Q, Wu C. QoE Fairness Resource Allocation in Digital Twin-Enabled Wireless Virtual Reality Systems," in IEEE Journal on Selected Areas in Communications, vol. 41, no. 11. 2023. p. 3355–3368. https://doi.org/10.1109/JSAC.2023.3313195.
6. Agahari W, Ofe H, de Reuver M (2022) It is not (only) about privacy: how multi-party computation redefines control, trust, and risk in data sharing. Electron Mark 32(3):1577–1602
7. Li YN, Feng XT, Xie J, Feng HW, Guan ZY, Wu QH (2020) A decentralized and secure blockchain platform for open fair data trading. Concurr Comput-Pract Exp 32(7):11. https://doi.org/10.1002/cpe.5578
8. Sonnino A, Al-Bassam M, Bano S, Meiklejohn S, Danezis G, Internet S (2019) Coconut: threshold issuance selective disclosure credentials with applications to distributed ledgers. 26th Annual Network and Distributed System Security Symposium (NDSS). Internet Soc, San Diego
9. Fan K, Pan Q, Zhang K, Bai YH, Sun SL, Li H, Yang YT (2020) A secure and verifiable data sharing scheme based on blockchain in vehicular social networks. IEEE Trans Veh Technol 69(6):5826–5835. https://doi.org/10.1109/tvt.2020.2968094
10. Liu DX, Huang C, Ni JB, Lin XD, Shen XS (2022) Blockchain-cloud transparent data marketing: consortium management and fairness. IEEE Trans Comput 71(12):3322–3335. https://doi.org/10.1109/tc.2022.3150724
11. Gai KK, Guo JN, Zhu LH, Yu S (2020) Blockchain meets cloud computing: a survey. IEEE Commun Surv Tutor 22(3):2009–2030. https://doi.org/10.1109/Comst.2020.2989392
12. Xue L, Ni J, Liu D, Lin X, Shen X (2023) Blockchain-based fair and fine-grained data trading with privacy preservation. IEEE Trans Comput 72(9):2440–2453. https://doi.org/10.1109/TC.2023.3251846
13. Dai WQ, Dai CK, Choo KKR, Cui CZ, Zou DQ, Jin H (2020) SDTE: a secure blockchain-based data trading ecosystem. IEEE Trans Inf Forensic Secur 15:725–737. https://doi.org/10.1109/tifs.2019.2928256
14. Zhang XH, Li XH, Miao YB, Luo XZ, Wang YW, Ma SQ, Weng J (2022) A data trading scheme with efficient data usage control for industrial IoT. IEEE Trans Ind Inform 18(7):4456–4465. https://doi.org/10.1109/tii.2021.3123312

15. Zhang QK, Li YJ, Wang RF, Liu L, Tan YA, Hu JJ (2021) Data security sharing model based on privacy protection for blockchain-enabled industrial Internet of Things. Int J Intell Syst 36(1):94–111. https://doi.org/10.1002/int.22293
16. Koutsos V, Papadopoulos D, Chatzopoulos D, Tarkoma S, Hui P. Agora: A Privacy-Aware Data Marketplace. IEEE Trans Dependable Secur Comput. 2022;19(6):3728–40. https://doi.org/10.1109/TDSC.2021.3105099.
17. Liu XF, Zhang YQ, Wang BY, Yan JB (2013) Mona: secure multi-owner data sharing for dynamic groups in the cloud. IEEE Trans Parallel Distrib Syst 24(6):1182–1191. https://doi.org/10.1109/tpds.2012.331
18. Huang H, Chen XF, Wang JF (2020) Blockchain-based multiple groups data sharing with anonymity and traceability. Sci China Inf Sci 63(3):13. https://doi.org/10.1007/s11432-018-9781-0
19. Cao XY, Chen Y, Liu KJR (2017) Data trading with multiple owners, collectors, and users: an iterative auction mechanism. IEEE Trans Signal Inf Proc Netw 3(2):268–281. https://doi.org/10.1109/tsipn.2017.2668144
20. Liu L, Feng J, Mu X, Pei Q, Lan D, Xiao M (2023) Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing. IEEE Trans Intell Trans Syst. 1–14. https://doi.org/10.1109/TITS.2023.3249745
21. Wang H, Qin H, Zhao MH, Wei XC, Shen H, Susilo W (2020) Blockchain-based fair payment smart contract for public cloud storage auditing. Inf Sci 519:348–362. https://doi.org/10.1016/j.ins.2020.01.051
22. Wang B, Li H, Liu X, Li F, Li X (2014) Efficient public verification on the integrity of multi-owner data in the cloud. J Commun Netw 16(6):592–599. https://doi.org/10.1109/JCN.2014.000105
23. Li S, Liu J, Yang GN, Han JG (2020) a blockchain-based public auditing scheme for cloud storage environment without trusted auditors. Wirel Commun Mob Comput 2020:13. https://doi.org/10.1155/2020/8841711
24. Miao Y, Huang Q, Xiao MY, Li HB (2020) Decentralized and privacy-preserving public auditing for cloud storage based on blockchain. IEEE Access 8:139813–139826. https://doi.org/10.1109/access.2020.3013153
25. Sedlmeir J, Smethurst R, Rieger A, Fridgen G (2021) Digital identities and verifiable credentials. Bus Inf Syst Eng 63(5):603–613. https://doi.org/10.1007/s12599-021-00722-y
26. Li Z (2022) A verifiable credentials system with privacy-preserving based on blockchain. J Inf Secur 13(2):43–65
27. Yoon D, Moon S, Park K, Noh S, Ieee (2021) Blockchain-based personal data trading system using decentralized identifiers and verifiable credentials. 12th International Conference on ICT Convergence (ICTC) - beyond the pandemic era with ICT convergence innovation. Ieee, Jeju Island, pp 150–4
28. Fotiou N, Pittaras I, Chadoulos S, Siris VA, Polyzos GC, Ipiotis N, Keranidis S (2023) Authentication, authorization, and selective disclosure for IoT data sharing using verifiable credentials and zero-knowledge proofs. Emerging technologies for authorization and authentication: 5th International workshop, ETAA 2022, Copenhagen, Denmark, September 30, 2022, revised selected papers. Springer, pp 88–101
29. Mukta R, Martens J, Paik HY, Lu QH, Kanhere SS (2020) Blockchain-based verifiable credential sharing with selective disclosure. 19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom). Ieee Computer Soc, Guangzhou, pp 960–7
30. Mahalle PN, Shinde G, Shafi PM (2020) Rethinking decentralised identifiers and verifiable credentials for the internet of things. Internet of things, smart computing and technology: a roadmap ahead. pp 361–74
31. Garcia-Rodriguez J, Moreno RT, Bernabe JB, Skarmeta A (2021) Implementation and evaluation of a privacy-preserving distributed ABC scheme based on multi-signatures. J Inf Secur Appl 62:15. https://doi.org/10.1016/j.jisa.2021.102971
32. Pointcheval D, Sanders O (2016) Short randomizable signatures. Cryptographers Track at the RSA Conference (CT-RSA). Springer International Publishing Ag, San Francisco, pp 111–26
33. Pointcheval D, Sanders O (2018) Reassessing security of randomizable signatures. Cryptographers'Track at the RSA Conference (CT-RSA). Springer International Publishing Ag, San Francisco, pp 319–38
34. Yu Y, Zhao YQ, Li YN, Du XJ, Wang LH, Guizani M (2020) Blockchain-based anonymous authentication with selective revocation for smart industrial applications. IEEE Trans Ind Inform 16(5):3290–3300. https://doi.org/10.1109/tii.2019.2944678
35. Camenisch J, Drijvers M, Lehmann A, Neven G, Towa P (2020) Short threshold dynamic group signatures. Security and cryptography for networks: 12th International Conference, SCN 2020, Amalfi, Italy, September 14–16, 2020, proceedings. Springer, pp 401–23
36. ElGamal T (1985) A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory 31(4):469–472
37. Canard S, Fuchsbauer G, Gouget A, Laguillaumie F (2012) Plaintext-checkable encryption. Topics in cryptology–CT-RSA 2012: The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27–March 2, 2012 proceedings. Springer, pp 332–48
38. Feldman P (1987) A practical scheme for non-interactive verifiable secret sharing. 28th Annual Symposium on Foundations of Computer Science (sfcs 1987). IEEE, Los Angeles, pp 427–38

Publisher's Note

Junli Fang (1985-) received her master's degree from Beijing Jiaotong University in 2009. She is currently a lecturer and a Ph.D. student in Lanzhou University of Technology. Her interests include cyber security and privacy preservation, Industrial Internet security and blockchain.

Tao Feng (1970-) received his Ph.D. degree from Xidian Univerisity in 2008. He is now a professor and Ph.D. supervisor in Lanzhou University of Technology. His research focuses on network and information security, Blockchain, and Industrial Internet security.

Xian Guo (1971-) received his Ph.D. degree from Lanzhou University of Technology in 2012. He is now a professor in Lanzhou University of Technology. His research focuses on blockchain and Industrial Internet security.

Rong Ma (1992-) received her master's degree from Fujian Normal University in 2019. Currently, she is a Ph.D. student in the School of Computer and Communication at Lanzhou University of Technology. Her interests include data security and privacy protection for Industrial Internet.

Ye Lu (1986-) received his Ph.D. degree from Lanzhou University of Technology in 2018. He is currently a lecturer in Lanzhou University of Technology. His research interests include cyber security and privacy preservation, industrial network security and blockchain.