Abstract Data plays a paramount role in today's business landscape as a significant driving force. Data sharing accelerates a company's data value exploration. However, one major obstacle hindering data sharing is the identity privacy breaches. Although various privacy-preserving data sharing approaches have been proposed, excessive protection measures could impede regulation and inadvertently offer protection to wrongdoers. In this work, we propose a privacy and regulatory approach for data sharing, ensuring the protection of data providers' identities while enabling the traceability of malicious providers without relying on centralized trust. We introduce an SM2-based traceable ring signature algorithm to achieve conditional privacy protection. By integrating this algorithm with blockchain technology, we devise an innovative data sharing scheme capable of identifying malicious data providers without exposing the original data. We present the security analysis to prove our scheme's correctness and security. Furthermore, we conduct experiments to assess the efficiency of our scheme, and the results demonstrate its effectiveness.

Keywords Anonymity, Blockchain, Data sharing, SM2, Traceable ring signature

1 Introduction

1.1 Background and Motivation

Data plays a crucial role in driving productivity in the globalized digital economy [1, 2]. Sharing data is a vital commercial necessity, as it enhances production value and unleashes the full potential of companies [3]. Essentially, data sharing enables the provision of data resources to diverse applications while ensuring its availability for entities worldwide. By addressing business challenges and enabling effective data-driven decision-making, data sharing optimizes the utilization of relevant data, generating valuable insights.

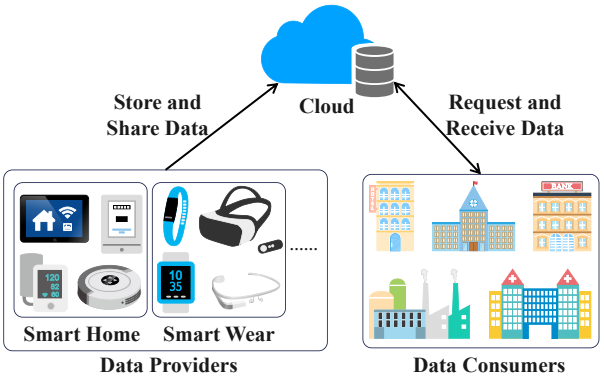


Fig. 1 A typical cloud-based data sharing system

In a typical cloud-based data sharing scenario, the system model comprises data consumers, data providers, and cloud storage providers (CSP). Data providers, with limited storage resources, outsource data to CSPs for sharing. With ample storage capacity, CSPs offer data sharing services to consumers requesting data. As shown in Figure 1, the stored data within the CSP, derived from smart home devices and wearables, holds the potential to reveal the provider's identity. Accordingly, when consumers such as banks and factories request data from the CSP, precautions must be taken to secure the privacy of the data and providers' identities.

Problems. The privacy issue poses a significant barrier to data sharing, raising substantial concerns regarding the potential risks to both economic assets and personal security [4]. Additionally, privacy-preserving technology can inadvertently offer refuge to illicit activities. For example, Monero, a cryptocurrency with advanced privacy features such as RingCT [5–8], has been exploited for money laun-

dering [9, 10]. The privacy safeguards provided by such technologies pose challenges for regulators attempting to trace the true identities of criminals involved in such activities.

Limitations. Leveraging its public, transparent, and tamper-resistant properties, blockchain facilitates traceability [11–13].Some researches have explored the storage of crucial parameters within the blockchain to enable traceability [14–19]. Nevertheless, it is crucial to recognize that the pseudonym mechanism employed by the blockchain falls short in protecting user identities via data analysis [20, 21]. Several studies have proposed traceable data sharing schemes leveraging the concept of traceable ring signatures (TRS) [22] across domains [23–27]. TRS provides conditional anonymity by allowing group users to sign data using a ring signature that conceals the identity of the signer within the group. Simultaneously, all TRS members collaborate to trace the true signer. By leveraging the trustworthiness of the blockchain as a storage platform for TRS parameters, the combination of TRS and blockchain holds promise in achieving trusted traceability [28, 29].

1.2 Proposed Scheme

We present a privacy and regulatory approach to data sharing that preserves the data provider's identity while tracing the malicious provider without the requirement for centralized trust. Based on SM2, the Chinese cryptographic public key algorithm standard [30], we propose an SM2-based traceable ring signature (STRS), ensuring both data authenticity and provider anonymity. The providers can collaboratively trace the signer *without a centralized trusted* authority. Subsequently, we design an STRS and blockchain-based data sharing scheme (SBDS). Within this scheme, critical parameters such as public keys and ring signatures are recorded in the blockchain to guarantee data authenticity and minimize communication overhead among entities. Additionally, we incorporate an STRS element into the blockchain, enabling traceability without the actual raw data.

Our key contributions in this work are as follow:

- We propose a traceable ring signature scheme utilizing the SM2 signature algorithm, facilitating traceability without the need for centralized trust.
- We design a data sharing scheme that incorporates our algorithm and blockchain technology. To aid in verification and tracing, we introduce a traceable data structure.
- We conduct comprehensive security analysis and test the overhead of our scheme. The results demonstrate effectiveness and efficiency.

1.3 Layout

The rest of this paper is organized as follows. Section 2 lists the blockchain privacy-enhancing schemes, related work of TRS, and TRS applications in data sharing. Section 3 gives the system model, threat model and design goals. Section 4 lists the preliminaries. Section 5 gives the construction of the proposed STRS. Section 6 proposes a data sharing scheme based on STRS. Section 7 analyzes the properties of our scheme. Section 8 evaluates the performance of our scheme. Section 9 presents the conclusion.

2 Related Work

This section presents privacy-enhancing schemes for blockchain technology, a comprehensive literature review on traceable ring signatures and the data sharing schemes built upon them.

2.1 Privacy-enhancing Schemes for Blockchain

Scholars have made strides in de-pseudonymisation, analyzing blockchain transaction graphs and other methods [20, 21]. Current privacy-enhancing technologies for blockchain systems can be broadly classified into two categories.

Mixing schemes. Prominent approaches [31–34] have been devised primarily for cryptocurrencies. They employ a mixer, either centralized or decentralized, to transfer indistinguishable tokens from various users to their intended destinations. By mixing a substantial number of participants' tokens, each with an equal amount, these methods effectively protect users' privacy by impeding attackers' ability to identify individuals through token flow analysis, thus preserving user anonymity.

Cryptography-based schemes. Cryptography protects identities and transaction values in privacy-centric blockchains. Monero [5] utilizes ring signatures, allowing users to obfuscate their transactions by amalgamating multiple inputs from various sources into a single transaction amount. These inputs are mixed together, forming a ring structure that masks the true origin of funds. This paper focuses on Monero as the leading solution employing ring signatures to enhance blockchain privacy. In contrast, Zcash [35] employs zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) for privacy preservation, enabling statement validation without exposing additional information. It supports shielded transactions, hiding users identities and transaction values.

Unfortunately, these technologies hide unlawful behaviours, making blockchain digital currency a popular payment medium for illegal activities like as money laundering and extortion [9, 10].

2.2 Traceable Ring Signatures

Ring signatures [36] enable a participant to anonymously sign a message on behalf of a group known as a "ring". While the verifier can authenticate the signature's validity, they cannot distinguish the actual identity of the signer within the ring. However, the inherent anonymity of ring signatures can be subject to abuse by ring members. To solve it, Fujisaki and Suzuki proposed traceable ring signatures in [22] aiming to mitigate the unregulated anonymity. Each message is accompanied by a unique random number referred to as ISSUE. The Trace algorithm in TRS allows for the identification of the signer within the ring who shares the same ISSUE. Fujisaki introduced a sub-linear traceable ring signature scheme, relying on a trusted common reference string (CRS) [37]. Ho Au et al. developed traceable ring signatures based on bilinear maps [38]. Teng et al. proposed an SM2-based traceable ring signature scheme for the smart grid, employing a stealth address to protect the identities of signers within a designated set of ring members [39]. However, their approach relies on a centralized and trusted monitoring center to trace signer identities. In our scheme, we eliminate the reliance on a centralized authority, ensuring that the traceability process aligns with the principles outlined in [22]. Scafuro et al. proposed one-time traceable ring signatures, where each member can anonymously sign a single message [40]. Fan et al. presented a ring signature utilizing the SM2 algorithm, which was introduced by the State Password Administration of China [41]. Peng et al. constructed a ring signature scheme using the SM9 algorithm [42].

2.3 Data Sharing Schemes with Traceability

TRS enables anonymity within a trustless environment while ensuring traceability, inspiring researchers to explore its applicability in various domains, including the power grid, healthcare, and more [23–29]. Samra and Fouzi introduced a certificateless aggregate scheme based on TRS to facilitate authentication [23]. Han et al. developed a lattice-based TRS that exhibits robust resistance against quantum attacks by leveraging the Short Integer Solution (SIS) problem [24]. Peng et al. proposed a traceable identity-based ring signature to secure the privacy of mobile IoT devices while enabling the identification of corrupted entities [25]. Lu et al. devised a traceable attribute-based ring signature system, allowing for precise control over access to electronic health records [26]. Fraser and Quaglia introduced a reporting system employing trace ring signatures, addressing the need for both signer anonymity and de-anonymity [27].

Blockchain technology facilitates the tracing of anonymous identities [28, 29, 43]. Lai et al. developed a certificateless traceable ring signature algorithm to ensure data integrity while enabling traceability in medical data sharing scenarios [28]. Tang et al. designed a multi-authority traceable ring signature scheme to address privacy concerns while utilizing blockchain technology to ensure consistency in storing electricity data [29]. These solutions enhance data sharing schemes based on TRS, improving both efficiency and security. However, they fail to consider the risk of data loss, which we address by storing intermediate parameters using blockchain, thus enabling traceability.

3 Overview

In this section, we introduce the system model, basic workflow, threat model, assumptions and design goals.

3.1 System Model

Figure 2 shows the system model of our scheme, which involves four entities:

- **Cloud storage provider(CSP)**. Commercial entities primarily fulfill this role, boasting ample storage resources and offering data storage services to both providers and consumers.
- **Consumers**. They request data from CSP for data analysis and smart decision-making.
- **Providers**. Individuals who use smart home or smart wear devices generate data, but their storage capacity is often limited. As a result, they form a group (i.e., a community) and collaborate with the CSP to share their data with data consumers.
- **Blockchain**. Blockchain operates on a global network of nodes, embodying the qualities of decentralization, tamper-proofing, and support for smart contract execution. Accessible to all, it assumes the responsibility of storing shared data information within the CSP.

Additionally, we design an STRS-based traceable data structure, named SDS, for data sharing. It contains the shared data and the smart contract in the blockchain. The details are shown in Section 6.1.

3.2 Basic Workflow

Our protocol is defined by a collection of phases as follow:

Setup Phase. In this phase, n data providers generate their cryptographic parameters. First, the

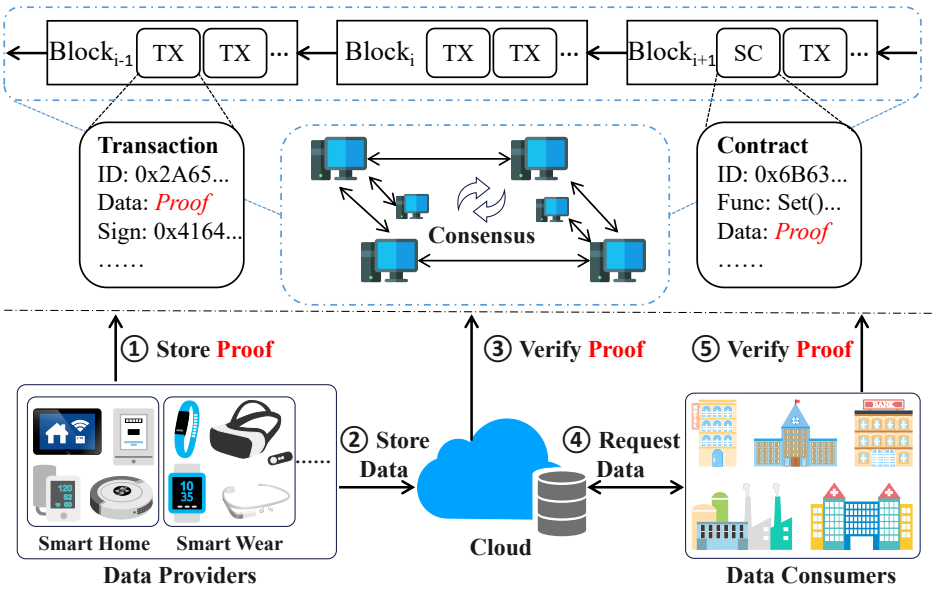


Fig. 2 System Model

providers generate their STRS parameters, i.e., the public key P_i and the private key d_i. Next, they should generate blockchain account parameters, i.e., the key pair (pk_i, sk_i).

Creating Phase. They need to construct SDS for data m before storing. First, they use a one-way function to produce a random number as the `DataID` with the data. Second, they should generate the traceable ring signature `TRSig`. Third, providers should send a transaction to deploy smart contract for the data. Finally, they fill the `Addr` to complete SDS.

Uploading Phase. After the provider completes the SDS, the data and its proof can be uploaded to the CSP. CSP will check the data with proof. First, CSP finds the blockchain smart contract according to the `Addr`. Next, CSP checks the `DataID` and `TRSig` in smart contract whether these data are equal to those in SDS. Last, CSP uses the `rpks` to verify the `TRSig`.

Using Phase. The consumer who counters the desired data will send a request to the CSP. After obtaining the data, the consumer can also check the data as the CSP does in **Uploading Phase**.

Tracing Phase. If some data are reported illegal, CSP should delete the data while keeping and sending the trace evidence (i.e., `DataID`, `Addr` and `TRSig`) to providers for accountability. Providers should execute the *Trace* algorithm with parameters in blockchain (i.e., `issue`) to disclose the malicious.

3.3 Threat Model and Assumption

Our design objective is to ensure the exposure of dishonest behavior while preserving privacy. Within each group, there are dozens of provider members who possess efficient key management systems, such as AWS KMS and Google KMS, thereby minimizing the risk of key compromise. The majority of providers honestly share accurate and lawful data. However, a minority of providers may update false or illegal data, which can have detrimental social and economic consequences. Additionally, CSP acts as an intermediary, storing data from providers and delivering it to consumers. We view it as a typical curious but honest role that provides data-

sharing services but desires to learn the data privacy [44, 45]. The adversary $\mathcal{A}$ with probabilistic polynomial time (PPT) computing ability is interested in the privacy of data and providers.

It is assumed that consumers are equipped with professional analysis and anomaly detection modules capable of identifying illegal data. We emphasize that our approach does not aim to supersede the CSP's existing anomaly detection capabilities. Building upon this assumption, consumers can prompt the CSP to halt the sharing of abnormal data.

We assume that the existing communication technologies ensure reliable transfer. The entities within the system can use anonymous communication tool, such as Tor [46], to conceal the network traffic.

We make the assumption that the secure append-only blockchain possesses the following features [11, 12]:

- **Immutability**. Once data has been confirmed by the blockchain, it becomes resistant to manipulation.
- **Openness**. All nodes within the blockchain network enjoy unrestricted freedom to join or leave the network.
- **Data Transparency** Data in the blockchain are openly accessible to all network nodes. any modifications or creations of data can be observed by all.
- **Identity Privacy-preserving**. The blockchain pseudonym mechanism indicates that the possibility that the adversary $\mathcal{A}$ infers the entity's real identity through a pseudonym is negligible.
- **Smart Contract**. It is a program that runs on the blockchain. Any node can send transactions to create a smart contract and alter the states of contract.

3.4 Design Goals

Our scheme aims at the following goals.

- **Data Confidentiality**. Due to the overwhelming amount of data, providers are compelled to store it in remote cloud centers. The shared data is of a highly sensitive nature and must be protected from unauthorized access.
- **Providers Anonymity**. Data providers share data within the group and seek to preserve the anonymity from entities outside the group. While others can determine that the shared data originates from a group, they cannot distinguish the specific individual responsible for sharing it.
- **Integrity**. Integrity guarantees that the CSP faithfully stores the shared data, and reliably delivers it to the consumers.
- **Traceability**. In cases where the shared data is incorrect or unlawful, data providers should possess the means to identify the responsible party. Furthermore, we maintain that there should be no centralized trust among these providers.

4 Preliminaries

4.1 Bilinear Pairing

Given two cyclic groups of large prime order q, G_1 and G_T. Let g_1 and g_2 be the generators of G_1 and G_T, respectively. A cryptographic bilinear map is a map e: $G_1 \times G_1 \rightarrow G_T$ satisfying the three properties as follows.

Bilinear: for $\forall P, Q \in G_1$ and $\forall x, y \in Z_q^*$, $e(P^x, Q^y) = e(P, Q)^{xy}$;

Non-degenerate: $\exists g_1 \in G_1$, then $e(g_1, g_1) \neq 1$;

Computable: the map e can be computed efficiently.

4.2 Elliptic Curve Discrete Logarithm Problem (ECDLP)

Let E be an elliptic curve over a finite field $\mathbb{F}$. Suppose that there are two points $P, Q \in E(\mathbb{F})$, satisfying $Q = k \cdot P$. It is hard to calculate k with Q and P.

4.3 SM2 Signature Algorithm

SM2 is a public key encryption standard adopted by the People's Republic of China. In this section, we briefly review the SM2 digital signature algorithm, which includes a set of algorithms: Setup, Key Generation, Signature Generation and Verification, defined below:

1. Setup. Given the security parameter 1^λ, the algorithm outputs an elliptic curve point group $\mathbb{G}$ of order q, where G is the generator of $\mathbb{G}$ and a hash function $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$.
2. Key Generation. A user U randomly chooses $d_A \in \mathbb{Z}_q^*$ as the private key, sets the public key $P_A = d_A \cdot G$. The algorithm outputs the key pair (pk, sk).
3. Signature Generation. Given a user's private key d_A, a message m, the user A first computes $e = H_1(m)$, then randomly chooses $k \in \mathbb{Z}_q^*$, then calculates $(x_1, y_1) = k \cdot G$, computes $r = (x_1 + e) \bmod q$ and $s = ((1 + d_A)^{-1} \cdot (k - r \cdot d_A)) \bmod q$. Finally, the algorithm outputs the signature $\sigma = (r, s)$.
4. Verification. Given a user's public key P_A, a message m, a signature (r, s), the verifier first checks whether $r, s \notin \mathbb{Z}_q^*$, then computes the hash value $e = H_1(m)$, lets $t = (r + s) \bmod q$, calculates $(x_1, y_1) = s \cdot G + t \cdot P_A$, sets $R = (e + x_1) \bmod q$. If $R = r$, the algorithm outputs 1, which indicates the signature is valid; otherwise outputs 0.

5 The Proposed SM2 Traceable Ring Signature

In this section, we put forward the SM2 traceable ring signature (STRS). we first give a formal definition of STRS. Then, we describe the algorithms in detail. Finally, we discuss the STRS.

5.1 Definition

A traceable ring signature scheme [22] Σ is a quartet of algorithms {KeyGen, Sign, Verify, Trace} defined as follows.

- KeyGen. It is a probabilistic polynomial-time algorithm, which takes as input the security parameter 1^λ, outputs a key pair (pk, sk).
- Sign. It is a probabilistic polynomial-time algorithm, which takes as input a private key sk_i, a tag TAG = {ISSUE, LIST} where LIST = $\{pk_1, pk_2, ..., pk_n\}$, where $i \in [1, n]$, and a message m, outputs a signature σ.
- Verify. It is a deterministic polynomial-time algorithm, which takes as input a tag TAG = {ISSUE, LIST} where LIST = $\{pk_1, ..., pk_n\}$, a message m, and a signature σ, outputs a bit b. If $b = 1$, the signature is valid; otherwise, not.
- Trace. It is a deterministic polynomial-time algorithm, which takes as input a tag TAG = {ISSUE, LIST}, two message/signature pairs $\{(m, \sigma), (m', \sigma')\}$, outputs the following string: "indep", "linked" or pk_i, where $pk_i \in$ LIST.

Public Traceability. The output of the Trace algorithm looks confusing, which we discuss in detail here. For any ISSUE, any message m, m', any $i, i' \in [1, n]$, we have $(pk, sk) \leftarrow$ KeyGen, $\sigma \leftarrow$ Sign(sk, TAG, m), $\sigma' \leftarrow$ Sign(sk, TAG, m), it holds

with an overwhelming probability that the following statement holds.

$$\mathsf{Trace}(\mathsf{TAG}, m, \sigma, m', \sigma') = \begin{cases} \text{“indep”}, & i \neq i' \\ \text{“linked”}, & i = i',\ m = m' \\ pk_i, & otherwise. \end{cases}$$

Correctness. We define the correctness of a traceable ring signature scheme as follows. For any ISSUE, any message m, any $i \in [1, n]$, we have $(pk, sk) \leftarrow \mathsf{KeyGen}$, $\sigma \leftarrow \mathsf{Sign}(sk, \mathsf{TAG}, m)$, it holds with an overwhelming probability Verify(TAG, m, σ)=1.

5.2 Constructions

In this section, we first appoint the notions that will be used. $H_1 : \{0,1\}^* \rightarrow G$, $H_2 : \{0,1\}^* \rightarrow G$ and $H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ denote hash functions, $\|$ denotes the concatenation of bit string, $\perp$ denotes abort, $\emptyset$ denotes an empty set. # denotes the number of members in a set. We use the same elliptic curve point group $\mathbb{G}$ of order q as the SM2 digital signature standard, where G is the generator of $\mathbb{G}$.

Here, we give the detailed constructions of our SM2-based traceable ring signature.

1. KeyGen. Each user runs this algorithm, gets their key pairs.
   (a) Each user U_i randomly chooses $d_{U_i} \in \mathbb{Z}_q^*$ as the private key, sets the public key $P_i = d_i \cdot G$.
   (b) The algorithm outputs the key pair (pk_i, sk_i), $pk_i = P_i$, $sk_i = d_i$.
2. Sign. Given the private key sk_i, the tag TAG = {ISSUE, LIST}, and the message m, the user runs this algorithm to output a signature σ.
   (a) The user U_i randomly chooses $(n-1)$ users' public keys, adds his own public key, constructs the list LIST = {pk_1, pk_2, ..., pk_n}, let TAG = {ISSUE, LIST}, the ISSUE is generated by a pseudo-random number generator.
   (b) Calculate the hash value $W = H_1(\mathsf{TAG})$.
   (c) Compute $\sigma_i = d_i \cdot W$.
   (d) Compute $A_0 = H_2(\mathsf{TAG}, m)$ and $A_1 = (\sigma_i / A_0)^{1/i}$.
   (e) For all $j \neq i$, compute $\sigma_j = A_0 {A_1}^j \in \mathbb{G}$.
   (f) Randomly choose $k_i \in Z_q^*$, set $c_{i+1} = H_3(\mathsf{TAG}, A_0, A_1, k_i \cdot G, k_i \cdot W)$.
   (g) For $j = i+1, \ldots, n, 1, \ldots, i-1$, the user U_i chooses $s_j \in Z_q^*$, computes $T_j = s_j \cdot G + \left(s_j + c_j\right) \cdot P_j$, $Y_j = s_j \cdot W + \left(s_j + c_j\right) \cdot \sigma_i$, then computes $c_{j+1} = H_3(\mathsf{TAG}, A_0, A_1, T_j, Y_j)$, lets $c_1 = c_{n+1}$.
   (h) Compute $s_i = \left((1 + d_i)^{-1} \cdot (k_i - c_i \cdot d_i)\right) \ mod\ q$.
   (i) Output the traceable ring signature value $\sigma = (A_1, c_1, s_1, s_2, \ldots, s_n)$ on message m.
3. Verify. Given the tag TAG = {ISSUE, LIST}, the message m, and the signature σ, the verifier runs this algorithm to output a bit.
   (a) If $c_1, s_1, s_2, \ldots, s_n \notin Z_q^*$, return $\perp$.
   (b) For $i = 1, 2, \ldots, n$, compute $A_0 = H_2(\mathsf{TAG}, m)$ and $\sigma_i = A_0 {A_1}^i$.
   (c) Calculate the hash value $W = H_1(\mathsf{TAG})$.
   (d) For $j = i+1, \ldots, n, 1, \ldots, i-1$, compute $T_j = s_j \cdot G + \left(s_j + c_j\right) \cdot P_j$, $Y_j = s_j \cdot W + \left(s_j + c_j\right) \cdot \sigma_i$, then compute $c_{j+1} = H_3\left(\mathsf{TAG}, A_0, A_1, T_j, Y_j\right)$.
   (e) If $c_1 = c_{n+1}$, then outputs $b = 1$; otherwise $b = 0$.
4. Trace. Given the tag TAG = {ISSUE, LIST}, two message/signature pairs $\{(m, \sigma), (m', \sigma')\}$, everyone can run this algorithm.
   (a) Calculate the hash value $W = H_1(\mathsf{TAG})$.
   (b) For $i = 1, 2, \ldots, n$, compute $A_0 = H_2(\mathsf{TAG}, m)$ and $\phi_i = A_0 {A_1}^i$.

(c) Similarly, for $i = 1, 2, \ldots, n$, compute $A_0' = H_2(\mathsf{TAG}, m')$ and $\phi_i' = A_0' {A_1'}^i$.

(d) For $i = 1, 2, \ldots, n$, if $\phi_i = \phi_i'$, store P_i on TLIST, TLIST is an empty list initially.

(e) Finally, perform the following steps:

- If the public key P is the only entry in TLIST, output P;
- If TLIST = LIST, output "link";
- If TLIST = $\emptyset$ or $1 <$ #TLIST $< n$, output "indep".

5.3 Discussion

This scheme realizes the generation of traceable ring signature based on the SM2 digital signature algorithm. The signer hides his identity in the signature group by collecting the users' public keys. The signature label generated at the same time alleviates the abuse of signature, and realizes the tracking of the signer by means of a secondary signature. The invention ensures the integrity, unforgeability, anonymity and traceability of the signature.

- **Integrity**. In signature phase, the signature $\sigma = (A_1, c_1, s_1, s_2, \ldots, s_n)$, $A_1 = (\sigma_i / A_0)^{1/i}$ and $A_0 = H_2(\mathsf{TAG}, m)$. In verification phase, the verifier calculates the $A_0 = H_2(\mathsf{TAG}, m)$ to verify the signature. Consequently, the signature σ is generated via original data m. Once the data is tampered with, it cannot pass the verification phase, so as to ensure the integrity of data.
- **Unforgeability**. In our construction, the signature $\sigma = (A_1, c_1, s_1, s_2, \ldots, s_n)$, where $A_1 = (\sigma_i / A_0)^{1/i}$ and $\sigma_i = d_i \cdot W$. The private key d_i is only known by the signer U_i, so it is impossible for others to forge the signature without private key.
- **Anonymity**. In verification phase, the verifier uses $\mathsf{LIST} = \{pk_1, pk_2, ..., pk_n\}$ to verify the signature σ. Besides, the auxiliary parameters $c_1, ...c_n$ are generated with users' public keys rather than the signer's public key. As a result, the probability that adversary can identify the real signer is less than $1/n$, where n is the number of users.
- **Traceability**. In trace phase, all users in the ring should resign data m' with the same tag TAG. Recall that $\phi_i = A_0 {A_1}^i = d_i \cdot W = d_i \cdot H_1(\mathsf{TAG})$. If the old signature σ is equal to new σ', it means that they are generated by the single private key d_i. Hence, we catch the signer.

6 STRS and Blockchain-based Data Sharing System

Our scheme ensures the traceability of sharing data leveraging STRS and blockchain. We first design a traceable data structure (SDS) to assist in data sharing in Section 6.1. Next, we give the formal definition in Section 6.2 to briefly describe the process. Then, we elaborate on it in Section 6.3. For the sake of readability, let's give the notations in Table 1.

6.1 STRS-based Traceable Data Structure

The STRS-based traceable data structure (SDS) contains data head $\mathrm{sds_h}$ and data content $\mathrm{sds_c}$. The former contains auxiliary information. The *DataID* is the unique identification of this data. The *Addr* is the smart contract address of this data. The *TR-Sig* is the traceable ring signature. The $\mathrm{obj_c}$ is the encrypted content for sharing.

Correspondingly, we use smart contract to record information so that consumers and the CSP can

Table 1 Summary of notations

Notations	Meanings
G	The base point of elliptic curve with order o.
bpk/bsk	The blockchain public/private key pairs.
$addr$	The address of smart contract.
K	The symmetric encryption (AES-256) key
λ	The security parameters.
$\mathbb{G}$	The multiplicative group.
q	The prime order of $\mathbb{G}$.
g	The generator of $\mathbb{G}$.
H_1, H_2, H_3	The distinct one-way functions.
n	The number of providers in one group.
h	The hash value.
pk/sk	The STRS public/private key pair of provider.
pks	The STRS public key set of all providers.
σ	The STRS signature generated by provider.
TAG	The tag used in STRS.
ISSUE	The random number used in STRS.
A_0, A_1, W	The elements for STRS.
T_i, Y_i	The intermediate parameter for signature.
s_i, c_i	The random number for STRS.
ϕ	The intermediate parameter for tracing.
sds	The STRS-based traceable data structure.
sds_H	The structure head of sds.
sds_C	The structure body of sds.

verify the sds. The entries of the contract contains $Addr$, $TRSig$, ISSUE, $Element$ and PKs. The $Addr$ is a built-in attribute that is the address of smart contract. The $TRSig$ is the traceable ring signature of this sds. The ISSUE is the random number used in the $TRSig$. The $Element$ records the elements used in the $TRSig$, i.e., A_0. The $TPKs$ records all traceable ring signature public keys of this group for the sds. The ISSUE and PKs are used to verify the $TRSig$. The ISSUE and A_0 are useful to trace the malicious provider. The former is the core of TRS to trace the provider who uses the same ISSUE to generate traceable ring signature. The latter is recorded in the blockchain. When the original data is broken, providers can use the A_0 to execute the tracing algorithm. Figure 3 gives the class diagram of SDS.

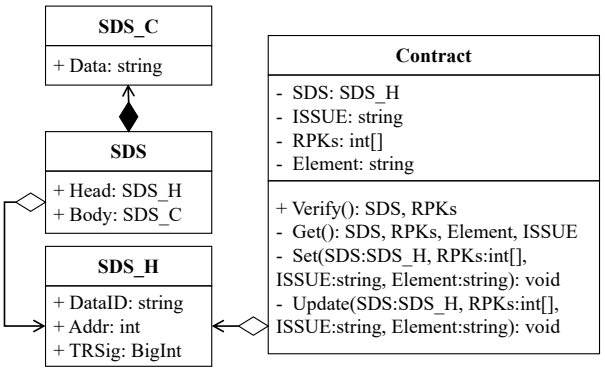


Fig. 3 The class diagram of STRS-based traceable data structure

6.2 Formal Definition

The formal definition is given as follows:

- **Setup**(1^λ). Take as input security parameters λ, all entities generate their cryptographic parameters for blockchain keys (bpk, bsk), TRS keys (pk, sk).
- **Creating**(bsk, pk, sk). The providers generate STRS signature σ for the encrypted data m, create smart contract SC in blockchain, and construct SDS.
- **Uploading**(m, σ). The CSP verifies the σ of m and parameters in SC sent by a provider.
- **Using**(m, σ). The consumer verifies the σ of m from the CSP and SC.
- **Tracing**(σ, pk, sk). Given the signature σ of illegal data, the providers jointly execute the trace algorithm of STRS and output the malicious provider.

6.3 A Concrete Scheme

Setup(1^λ):

1. We adopt elliptic curves cryptography (ECC) algorithm to generate blockchain public/private

key pair as Bitcoin and Ethereum. In a finite field $\mathbb{F}_R$ with prime p, we choose a curve $y^2 = x^3 + ax + b$, let G be a base point of this curve with order $o, o > 2^{160}$. Let provider p_i randomly choose a number bsk in $[1, o-1]$ and calculates $bpk = bsk \cdot G$. The BC public key of p_i is bpk and the private key is bsk. We let $bpks = (bpk_1, \ldots, bpk_n)$ be an ordered public-key list for n providers. We also use the public key bpk represent the address $addr$ in blockchain.

2. Let $\mathbb{G}$ be a multiplicative group of prime order q and let g be a generator of $\mathbb{G}$, $H_1 : \{0,1\}^* \rightarrow G$, $H_2 : \{0,1\}^* \rightarrow G$ and $H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ be distinct one-way functions, $\|$ denotes the concatenation of bit string, $\perp$ denotes abort, $\emptyset$ denotes an empty set. Provider p_i picks up random element $sk_i \in \mathbb{Z}_q^*$ and computes $pk_i = sk_i \cdot G$. The TRS public key of p_i is pk_i and the corresponding private key is sk_i. We let $pks = (pk_1, \ldots, pk_n)$ be an ordered public-key list for n providers.
3. The provider p_i generates a symmetric encryption (AES-256) key K and sends it to other members in this ring.

Creating(bsk, pk, sk):

1. The providers have created the data $data$ and encrypt it with key K (i.e., sds_c) and leveraged CSP for sharing. Then, they need to generate the sds_h, including $DataID$, $Address$ and $TRSig$.
2. They use the one-way function to generate the $DataID \leftarrow H_3(\text{sds}_\text{c})$.
3. One provider p_i generate tag $\mathsf{TAG} = \{\mathsf{ISSUE}, \mathsf{LIST}\}$, where ISSUE is a random number, $\mathsf{LIST} = pks$. The provider p_i calculates the hash value $W = H(\mathsf{TAG})$, computes signature $\sigma_i = sk_i \cdot W$, sets $A_0 = H_2(\mathsf{TAG}, \text{sds}_\text{c})$ and $A_1 = \left(\frac{\sigma_i}{A_0}\right)^{\frac{1}{i}}$. For all $j \neq i$, p_i computes $\sigma_j = A_0 A_1^j$. Then, the provider p_i randomly chooses $k_i \in Z_q^*$, set $c_{i+1} = H_3(\mathsf{TAG}, A_0, A_1, k_i \cdot G, k_i \cdot W)$. For $j = i+1, \ldots, n, 1, \ldots, i-1$, the provider p_i chooses $s_j \in Z_q^*$, computes $T_j = s_j \cdot G + \left(s_j + c_j\right) \cdot pk_j$, $Y_j = s_j \cdot W + \left(s_j + c_j\right) \cdot \sigma_i$, then computes $c_{j+1} = H_3(\mathsf{TAG}, A_0, A_1, T_j, Y_j)$, lets $c_1 = c_{n+1}$. Then the p_i computes $s_i = \left((1 + d_i)^{-1} \cdot (k_i - c_i \cdot d_i)\right) \; mod \; q$. The traceable ring signature is $TRSig = \sigma = (A_1, c_1, s_1, s_2, \ldots, s_n)$ on message sds_c.
4. The provider p_i constructs a transaction to create the smart contract with parameters (pks, $\sigma, \mathsf{ISSUE}, A_0$) to get the address of contract $addr_{SC}$ to fill sds_h. We set $Addr = addr_{SC}$.

Uploading(sds, σ):

1. The provider sends the (sds, σ) to the CSP with the description information that is a descriptive document to describe the content of the data, such as `readme` file in GitHub.
2. The CSP extracts the address of smart contract from sds_h.
3. The CSP accesses the blockchain to fetch the parameters $(pks, \sigma^{BC}, \mathsf{ISSUE})$ according to $Addr$. Then, the DSP compares the σ of sds_h and the σ^{BC} of smart contract.
4. If they are equal, the CSP parses the $\sigma = (A_1, c_1, s_1, s_2, \ldots, s_n)$. The CSP parses $\mathsf{TAG} = (\mathsf{ISSUE}, pks)$, computes $W = H(\mathsf{TAG})$, $A_0 = H_2(\mathsf{TAG}, \text{sds}_\text{c})$. For $j = i+1, \ldots, n, 1, \ldots, i-1$, CSP computes $T_j = s_j \cdot G + \left(s_j + c_j\right) \cdot pk_j$, $Y_j = s_j \cdot W + \left(s_j + c_j\right) \cdot \sigma_i$, then computes $c_{j+1} = H_3\left(\mathsf{TAG}, A_0, A_1, T_j, Y_j\right)$. If $c_1 = c_{n+1}$, then the verification is successful.
5. The CSP stores the data and makes the description information public.

Using(sds, σ):

1. According to the description, the consumer can find and request the desired data from the CSP.
2. The CSP sends sds to the consumer.
3. The consumer accesses the blockchain to get the parameters $(pks, \sigma^{BC}, \mathsf{ISSUE})$ according to $Addr$ of $\mathrm{sds_h}$. Then, the consumer compares the σ of $\mathrm{sds_h}$ and the σ^{BC} of smart contract.
4. If they are equal, If they are equal, the consumer parses the $\sigma = (A_1, c_1, s_1, s_2, \ldots, s_n)$. The consumer parses $\mathsf{TAG} = (\mathsf{ISSUE}, pks)$, computes $W = H(\mathsf{TAG}), A_0 = H_2(\mathsf{TAG}, \mathrm{sds_c})$. For $i = 1, 2, \ldots, n$, CSP computes $T_j = s_j \cdot G + \left(s_j + c_j\right) \cdot pk_j$, $Y_j = s_j \cdot W + \left(s_j + c_j\right) \cdot \sigma_i$, then computes $c_{j+1} = H_3\left(\mathsf{TAG}, A_0, A_1, T_j, Y_j\right)$. If $c_1 = c_{n+1}$, then the verification is successful.
5. The consumer sends the public key bpk_c to the CSP. The CSP sends it to the providers. Every providers can encrypt K with bpk_c to get $\{K\}^{enc}_{bpk_c}$, and sends it to the CSP. The CSP sends $\{K\}^{enc}_{pk_c}$ to the consumer. The consumer decrypts it with private key bsk_c to obtain K. Finally, the consumer can decrypt $\mathrm{sds_c}$ to get the plaintext.

Tracing(σ, pk, sk):

1. As the assumption in Section 3.3, the consumer deployed an analysis module and an anomaly detection module can catch the illegal data. Then, the consumer constructs a warning transaction with $DataID$ and the corresponding σ, sends it to the blockchain.
2. The CSP should keep eyes on the blockchain, once it finds the warning transaction, it should suspend the accessibility of illegal data and sends the $DataID$ to the providers.
3. The providers access the blockchain to obtain the parameters $(pks, \sigma, \mathsf{ISSUE}, A_0)$. Compare the ISSUE in the blockchain and it in the $\mathrm{sds_h}$.
4. If they are equal, the providers should check whether the shard data are illegal or incorrect.
5. If the data are clean, the providers should inform the CSP, and CSP will recover the accessibility.
6. If not, provider p_i generates a new data m', calculates the hash value $W = H_1(\mathsf{TAG})$. For $i = 1, 2, \ldots, n$, p_i computes $\phi_i = A_0 {A_1}^i$. Similarly, for $i = 1, 2, \ldots, n$, p_i computes $A'_0 = H_2(\mathsf{TAG}, m')$ and $\phi'_i = A'_0 {A'_1}^i$. For $i = 1, \ldots, n$, if $\phi_i = \phi_i{}'$, the i is the order number of the malicious provider.
7. Get rid of the malicious provider, others need to rebuild the ring and smart contract, generate new data and send them to the CSP, following the process explained in **setup** and **creating** phases.

7 Correctness and Security

In this section, we analyze the correctness and security features of our scheme.

Theorem 1 (Correctness)**.** The correctness of STRS signature is twofold. On the one hand, the honestly generated signature should always be valid. On the other hand, the Trace algorithm should always output the signer identity of a valid signature.

Proof. (1) Assume that the provider p_i has generated a signature σ for data m, following the algorithm Sign. We can parse that $\sigma = (A_1, c_1, s_1, \ldots, s_n)$, where

$$c_1 = c_{n+1} = H_3(\mathsf{TAG}, A_0, A_1, T_n, Y_n)$$

$$T_n = s_n \cdot G + (s_n + c_n) \cdot P_n$$

$$Y_n = s_n \cdot W + (s_n + c_n) \cdot \sigma$$

$$c_{i+1} = H_3(\mathsf{TAG}, A_0, A_1, k_i \cdot G, k_i \cdot W)$$

The verification follows the similar process except $c_{i+1} = H_3(\mathsf{TAG}, A_0, A_1, T_i, Y_i)$, where

$$\begin{aligned} T_i &= s_i \cdot G + (s_i + c_i) \cdot P_i \\ &= s_i \cdot G + (s_i + c_i) \cdot d_i \cdot G \\ &= (1 + d_i) \cdot s_i \cdot G + c_i d_i \cdot G \\ &= (1 + d_i) \cdot \Big((1 + d_i)^{-1} \cdot (k_i - c_i d_i)\Big) \cdot G + c_i d_i \cdot G \\ &= k_i \cdot G, \end{aligned}$$

$$\begin{aligned} Y_i &= s_i \cdot W + (s_i + c_i) \cdot \sigma \\ &= s_i \cdot W + (s_i + c_i) \cdot d_i \cdot W \\ &= (1 + d_i) \cdot s_i \cdot W + c_i d_i \cdot W \\ &= (1 + d_i) \cdot \Big((1 + d_i)^{-1} \cdot (k_i - c_i d_i)\Big) \cdot W + c_i d_i \cdot G \\ &= k_i \cdot W \end{aligned}$$

. Consequently, we obtain

$$c_{i+1} = H_3(\mathsf{TAG}, A_0, A_1, k_i \cdot G, k_i \cdot W),$$

therefore, we can get $c_1 = c_{n+1}$.

(2) For the Trace algorithm, given the tag $\mathsf{TAG} = \{\mathsf{ISSUE}, pks\}$, two message/signature pairs $\{(m, \sigma), (m', \sigma')\}$, all providers calculate the hash value $W = H_1(\mathsf{TAG})$. For $i = 1, 2, \ldots, n$, compute $A_0 = H_2(\mathsf{TAG}, m)$ and $\phi_i = A_0 {A_1}^i$. Similarly, for $i = 1, 2, \ldots, n$, compute $A'_0 = H_2(\mathsf{TAG}, m')$ and $\phi'_i = A'_0 {A'_1}^i$. We can observe that

$$\begin{aligned} \phi'_i &= A'_0 {A'_1}^i \\ &= A'_0 (\sigma'_i / A'_0)^{1/i^i} \\ &= \sigma'_i = d_i \cdot W = \sigma_i \\ &= A_0 (\sigma_i / A_0)^{1/i^i} \\ &= A_0 {A_1}^i = \phi_i. \end{aligned}$$

As a result, we can conclude that if $\phi_i = \phi_i{}'$, the signer is the provider p_i. Hence, the correctness of our scheme is hold. □

Theorem 2 (Unforgeability)**.** The STRS scheme is unforgeability if the Elliptic Curve Discrete Logarithm Problem (ECDLP) holds for the PPT adversary $\mathcal{A}$.

Proof. Assuming that there exists a PPT adversary $\mathcal{A}$ which may be the data provider or any other, it holds the system public parameters $(q, n, g, \mathbb{G}, G, H_1, H_2, H_3)$. Then $\mathcal{A}$ tries to forge a ring signature which can pass the verification. At the beginning, we assume that $\mathcal{A}$ can interact with the signer U_i, therefore $\mathsf{LIST} = \{P_1, P_2, ...P_n\}$, ISSUE and message m can be viewed by $\mathcal{A}$ in the STRS signing phase. If $\mathcal{A}$ forges the signature σ without private key d_i. We say the signer holds:

$$(m, T_1, T_2, ..., T_n, c_1, c_2, ...c_n, s_1, s_2, ..., s_n),$$

and the $\mathcal{A}$ holds

$$(m, T'_1, T'_2, ..., T'_n, c'_1, c'_2, ...c'_n, s'_1, s'_2, ..., s'_n).$$

For $i \in [1, n]$, there are $T_i = T'_i$, but $c_i \neq c'_i$, $s_i \neq s'_i$. According to the signing phase, we get

$$T_i = s_i \cdot G + (s_i + c_i) \cdot P_i$$

$$T'_i = s'_i \cdot G + (s'_i + c'_i) \cdot P_i.$$

Since $T_i = T'_i$, we get

$$P_i = \frac{(s'_i - s_i) \cdot G}{(s'_i - s_i) + (c'_i - c_i)}.$$

Since $P_i = d_i \cdot G$, we get

$$d_i = \frac{(s'_i - s_i)}{(s'_i - s_i) + (c'_i - c_i)}.$$

The adversary $\mathcal{A}$ can calculate d_i. In other words, the PPT adversary $\mathcal{A}$ can solve ECDLP, which contradicts the assumption. Hence, the $\mathcal{A}$ cannot forge the signature σ without private key. □

Theorem 3 (Anonymity)**.** Our scheme is anonymous for the PPT adversary $\mathcal{A}$.

Proof. Assuming the adversary $\mathcal{A}$ get a valid signature $\sigma = (A_1, c_1, s_1, s_2, \ldots, s_n)$ on message m. And assuming that $\mathcal{A}$ can distinguish the private key d of σ with non-negligible probability. For A_1, $A_1 = (\sigma_i/A_0)^{1/i}$, $A_0 = H_2(\mathsf{TAG}, m)$, $\sigma_i = d_i \cdot W$. Since all the elements are known by the $\mathcal{A}$, we say $\mathcal{A}$ obtains A_1. For c_1, $c_1 = c_{n+1} = H_3(\mathsf{TAG}, A_0, A_1, k_n \cdot G, k_n \cdot W)$, k_n is the integer in Z_q^* and is statistically indistinguishable. On the other hand, the c_1 is the hash value, which is indistinguishable due to the collision-resistance of hash function. That means the $\mathcal{A}$ cannot find a value k' to generate a hash value that equel to c_1. For $(s_1, s_2, \ldots, s_n)$, they are integers in Z_q^* and are statistically indistinguishable. In summary, the $\mathcal{A}$ cannot distinguish the identity of signer based on the valid signature. Hence, STRS scheme is anonymous. □

Theorem 4 (Integrity)**.** STRS scheme is able to ensure the data integrity for PPT adversary.

Proof. Assuming the signer U_i signs the message m to get signature $\sigma = (A_1, c_1, s_1, s_2, \ldots, s_n)$. Assuming the adversary $\mathcal{A}$ get a valid signature $\sigma' = \left(A_1', c_1', s_1', s_2', \ldots, s_n'\right)$ on message m' without private key d_i, where $m \neq m'$.

The consumer parses the two signatures $\sigma = (A_1, c_1, s_1, s_2, \ldots, s_n)$, and $\sigma' = \left(A_1', c_1', s_1', s_2', \ldots, s_n'\right)$. If $\sigma = \sigma'$, we say at least $A_1 = A_1'$. For A_1, $A_1 = (\sigma_i/A_0)^{1/i}$, $A_0 = H_2(\mathsf{TAG}, m)$. For A_1', $A_1' = (\sigma_i/A_0')^{1/i}$, $A_0' = H_2(\mathsf{TAG}, m')$. If $A_1 = A_1'$, we say at least $A_0 = A_0'$. Then the adversary can calculate the A_0' without pre-image m. We say the adversary can destroy the collision resistance of hash functions, which contradicts our assumption. Any modification of the message m will be detected by the consumers. As a result, STRS supports integrity. □

Theorem 5 (Data Authenticity)**.** STRS scheme can achieve data authenticity for PPT adversary.

Proof. The consumers can check the messages validity by verifying the signature σ. Based on *unforgeability* (Theorem 2) and *integrity* (Theorem 4) that are on the strength of the elliptic curve discrete logarithm problem and the collision resistance of hash functions, the PPT adversary cannot forge a valid signature for the message m without the private key. Therefore, STRS supports authenticity of data. □

Theorem 6 (Traceability)**.** In STRS, any signer who violates the protocol would be exposed even the original data is broken.

Proof. As the explained in Theorem 1, we can say that if the original data m is accessible, the providers can re-generate the signatures, and they can trace the signer. However, if the original data m is broken, the providers cannot re-generate the signatures. To this end, we require the providers to store the element A_0 in the blockchain. For the Trace algorithm, given the tag $\mathsf{TAG} = \{\mathsf{ISSUE}, pks\}$, only the new message/signature pair $\{(m', \sigma')\}$, all providers calculate the hash value $W = H_1(\mathsf{TAG})$. For $i = 1, 2, \ldots, n$, compute $\phi_i = A_0 {A_1}^i$. Since the A_0 is stored in the blockchain, we can also compute the $\phi_i = A_0 {A_1}^i$ without the original data m. For $i = 1, 2, \ldots, n$, compute $A_0' = H_2(\mathsf{TAG}, m')$ and $\phi_i' = A_0' {A_1'}^i$. We can observe that

$$
\begin{aligned}
\phi_i' &= A_0' {A_1'}^i \\
&= A_0' (\sigma_i'/A_0')^{1/i^i} \\
&= \sigma_i' = d_i \cdot W = \sigma_i \\
&= A_0 (\sigma_i/A_0)^{1/i^i} \\
&= A_0 {A_1}^i = \phi_i.
\end{aligned}
$$

Recall that, based on the collision resistance of hash functions and the elliptic curve discrete logarithm problem, the PPT adversary cannot forge a valid signature. As a result, we can conclude that STRS can trace the signer without the original data. □

8 Performance Evaluation

In this section, we analyze the communication overhead and computation overhead of STRS from the theory, and then we perform the evaluation using Hyperledger Fabric as the blockchain backend.

8.1 Theoretical Analysis

8.1.1 Communication Overhead

Let $|q|$ be the bits of order q, $|\mathbb{G}|$ be the bits of group $\mathbb{G}$. In STRS, the Sign generates signatures $\sigma = (A_1, c_1, s_1, s_2, \ldots, s_n)$, where $A_1 \in \mathbb{G}$ and $c_1, s_1, s_2, \ldots, s_n \in \mathbb{Z}_q^*$. Consequently, the communication cost of STRS are $(n+1)|q| + |\mathbb{G}|$.

8.1.2 Computation Overhead

This paper focus the computation cost in algorithms includes Sign, Verify and Trace. Suppose that the number of ring member is n, T_{add} is the additive operation on group $\mathbb{G}$, T_{mul} denotes the multiplication operation on group $\mathbb{G}$, T_{exp} is the exponentiation operation on $\mathbb{G}$, T_H is the hash function for $\{0,1\}^* \rightarrow G$ (*e.g.*, the H_1 and H_2), and $T_{H'}$ represents the hash function for $\{0,1\}^* \rightarrow \mathbb{Z}_q^*$ (*e.g.*, the H_3). In addition, T_{ZA} denotes the inverse operation on group $\mathbb{Z}_q^*$. T_E denotes the pair operation. We analyze and compare STRS and other related ring signature work, the results are shown in Table 2.

8.2 Experimental Evaluation

The experiment is carried out on a laptop with an Ubuntu 20.04 operating system, an Intel Core i7-10510U processor and 8GB of memory. We tested the overhead of STRS with the Miracle Library[1)]. We used Hyperledger Fabric v2.3.2 as blockchain to run smart contract coded via IBM blockchain platform. Based on the `fabric-sdk-go`, we built the web interfaces to interact with blockchain. We also evaluated the performance with web load testing tools `http_load` and `siege` v4.1.2.

8.2.1 Microbenchmarks

In the STRS experiments, we use the SM2 curve `sm2p256v1`, the number of ring members is 10, we give the computation overhead of the algorithms in Figure 4, the lengths of the messages used are 4K bytes. The results prove that the overhead are millisecond-level, so it is acceptable for users.

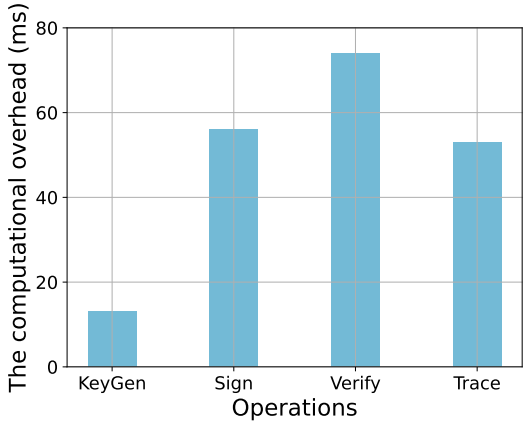


Fig. 4 The computational overhead of STRS

8.2.2 Macrobenchmarks

From a macro-level perspective, we evaluated the time costs of steps in a local Hyperledger Fabric network. This network consists of one order node and two peer nodes from two organizations. The smart contract (chain-code in Hyperledger Fabric) that records critical data elements $DataID$, $TRSig$,

[1)] Miracle Library

Table 2 Computation overhead of ring signature schemes

Schemes	Sign	Verify	Trace
STRS	$2T_H + nT_{H'} + T_{ZA}$ $+(2n-1)T_{add} + (4n+1)T_{mul}$	$2T_H + (n-1)T_{H'}$ $+(2n-2)T_{add} + (6n-4)T_{mul}$	$3T_H + (4n)T_{mul}$
TRS [22]	$2T_H + T_{H'}$ $+(3n-1)T_{mul} + (7n+1)T_{exp}$	$2T_H + T_{H'}$ $+(3n)T_{mul} + (5n)T_{exp}$	$4T_H$ $+(2n)T_{mul} + (2n)T_{exp}$
MATRS [29] [1]	$nT_{H'}$ $+(7n+2)T_{mul}$	$2T_E$ $+(2n-2)T_{mul}$	$4nT_E$ $+(n+\ell-1)T_{mul}$
SATRS [39] [2]	$(n+2)T_{H'} + T_{ZA}$ $+(4n-4)T_{add} + KeyEn + Enc$	$(n)T_{H'} + (3n)T_{add}$	$Dec + 2T_{add}$
OTRS [40] [3]	$(n[\lambda]+1)T_{H'}$	$(n[\lambda]+1)T_{H'}$	$(n^2[\lambda]^2)T_{H'}$
SRS [41]	$nT_H + T_{ZA}$ $+(n-1)T_{add} + (2n-1)T_{mul}$	nT_H $+nT_{add} + (2n)T_{mul}$	-
SLRS [41]	$T_H + nT_{H'} + T_{ZA}$ $+(2n-2)T_{add} + (4n-1)T_{mul}$	$nT_{H'} + T_{ZA}$ $+(2n)T_{add} + (4n)T_{mul}$	-

[1] ℓ denotes the number of records sent by the signer.

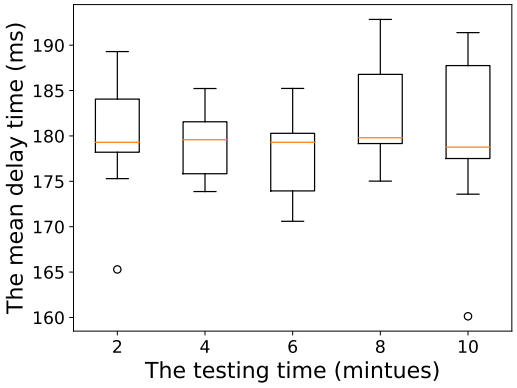
[2] *KeyEn* denotes SM9 key generation algorithm. *Enc* represents SM2 encryption algorithm. *Dec* denotes SM2 decryption algorithm.

[3] $[\lambda]$ denotes the length of oracle value.

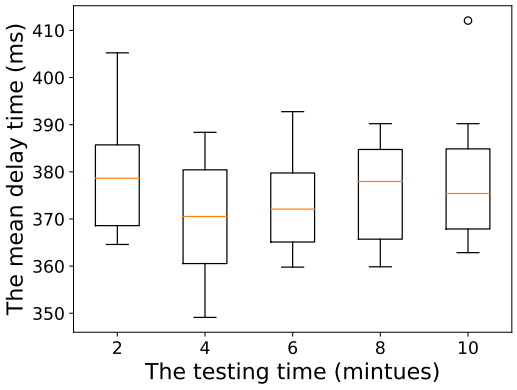
$ISSUE$, $rpks$ and A_0, consists of functions `query` and `update`. The former allows the CSP to retrieve above parameters. The latter enables the provider to set the aforementioned parameters. We simulated the interaction of users and consumers with blockchain and built two web interfaces for each function based on the Fabric SDK `fabric-sdk-go`. To assess performance, we employed `http_load` and `siege`. We configured the system with 100 concurrent consumers and conducted test runs of 2, 4, 6, 8, and 10 minutes respectively. The results of `http_load` are shown in Figure 5, which displays the average transaction delay at the millisecond (ms) level. The mean waiting time is less than 200 ms for `query`, while for `update` it is approximately 400 ms. Since the `query` operation involves reading the blockchain state and does not require consensus, it exhibits faster processing compared to `update`.

Figure 6 gives the results of `siege`, providing insights into the transaction rate and the number of successful transactions. Figure 6(a) represents the transaction rate for both `query`and `update` operations. It is notable that the transaction rate for the `query` operation reaches approximately 430 transactions per second, while for `update` it is around 140 transactions per second. This gap means that blockchain is more suitable for processing read-heavy work. The `update` operation, involving modifications to the blockchain state and requiring consensus among nodes, exhibits a lower throughput. Figure 6(b) indicates the number of successful transactions, we focus on the count of successful transactions. Remarkably, the trade execution demonstrates relative smoothness.

We should note that these experiments were con-

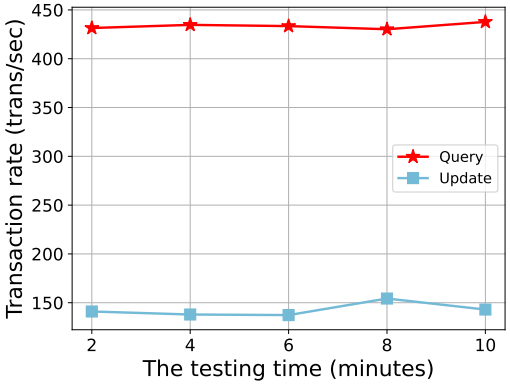


(a) Query

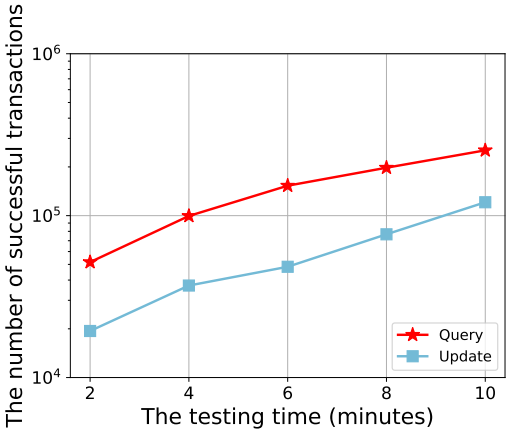


(b) Update

Fig. 5 The results of `http_load`



(a) The transaction rate



(b) The successful transactions

Fig. 6 The results of `siege`. The red solid line with a star denotes the transaction rate, and the blue solid line with a square represents the transaction number

ducted on a laptop with common performance capabilities. Considering the hardware environment, achieving a mean waiting time of less than 400 ms and a transaction throughput exceeding 100 per second is considered satisfactory. From this perspective, we can infer that our scheme is capable of supporting a large number of consumers when deployed in a server cluster equipped with adequate computational resources.

9 Conclusion

To address the privacy and regulatory challenges of existing data sharing schemes, we propose traceable ring signatures based on the SM2 signature algorithm, providing conditional anonymity. Moreover, we introduce traceable data structures and employ blockchain technology for data sharing information storage. Our scheme enables identification of malicious data provider while protecting innocent ones, without requiring access to the original data. Comprehensive security analysis validates the correctness, while evaluation shows feasibility.

Future work. The subsequent studies will focus on addressing the following inquiries: (1) Privacy concerns pertaining to data consumers. Present data privacy compliance laws and regulations, primarily emphasize the protection of data providers' privacy, often overlooking the privacy requirements of data consumers. Consequently, our future research will delve deeper into methods for safeguarding the

privacy of data consumers. (2) Ensuring high efficiency for large-scale providers. Our current solution exhibits a slower tracing process as the size of the ring increases. Consequently, it proves less efficient for sizable groups. However, this limitation can be mitigated by partitioning the extensive group into multiple smaller ones.

This paper serves as an extension of our previous conference paper [47].

Acknowledgements This work was supported in part by the National Key R&D Program of China (No.2021YFB2700600); in part by the Finance Science and Technology Project of Hainan Province (No.ZDKJ2020009); in part by the Hainan Province Science and Technology Special Fund (No.GHYF2022010); in part by the National Natural Science Foundation of China (Nos.62163011, 62072092, 62072093 and U1708262); in part by the Fundamental Research Funds for the Central Universities (No.N2023020); in part by the Natural Science Foundation of Hebei Province (No.F2020501013); in part by the China Postdoctoral Science Foundation (No.2019M653568); and in part by the Key Research and Development Project of Hebei Province (No.20310702D).

References

1. Zheng X, Cai Z. Privacy-preserved data sharing towards multiple parties in industrial iots. IEEE Journal of Selected Areas in Communications, 2020, 38(5): 968–979
2. Kong Q, Su L, Ma M. Achieving privacy-preserving and verifiable data sharing in vehicular fog with blockchain. IEEE Transactions on Intelligent Transportation Systems, 2021, 22(8): 4889–4898
3. Fu A, Yu S, Zhang Y, Wang H, Huang C. NPP: A new privacy-aware public auditing scheme for cloud data sharing with group users. IEEE Transactions on Big Data, 2022, 8(1): 14–24
4. Liu Y, Yu J, Yang M, Hou W, Wang H. Towards fully verifiable forward secure privacy preserving keyword search for IoT outsourced data. Future Generation Computer Systems, 2022, 128: 178–191
5. Monero . About monero. https://getmonero.org/knowledge-base/about, 2023. Accessed 4 Mar. 2023.
6. Sun S, Au M H, Liu J K, Yuen T H. RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency Monero. In: Proceeding of the 22nd European Symposium on Research in Computer Security (ESORICS). 2017, 456–474
7. Yuen T H, Sun S, Liu J K, Au M H, Esgin M F, Zhang Q, Gu D. RingCT 3.0 for blockchain confidential transaction: Shorter size and stronger security. In: Proceeding of the 2020 Financial Cryptography and Data Security (FC). 2020, 464–483
8. Zhang F, Huang N, Gao S. Privacy data authentication schemes based on borromean ring signature. Journal of Cryptologic Research, 2018, 5: 529–537
9. Kolachala K, Simsek E, Ababneh M, Vishwanathan R. SoK: Money laundering in cryptocurrencies. In: Proceeding of the 16th International Conference on Availability, Reliability and Security (ARES). 2021, 5:1–5:10
10. Rathore M M, Chaurasia S S, Shukla D. Mixers detection in bitcoin network: a step towards detecting money laundering in crypto-currencies. In: IEEE International Conference on Big Data, (Big Data). 2022, 5775–5782
11. Feng Q, He D, Zeadally S, Khan M K, Kumar N. A survey on privacy protection in blockchain system. Journal of Network and Computer Applications, 2019, 126: 45–58
12. Huang H, Kong W, Zhou S, Zheng Z, Guo S. A survey of state-of-the-art on blockchains: Theories, modelings, and tools. ACM Computing Surveys, 2021, 54(2): 44:1–44:42
13. Lin G, Wang H, Wan J, Zhang L, Huang J. A blockchain-based fine-grained data sharing scheme for e-healthcare system. Journal of Systems Architecture, 2022, 132: 102731
14. Su Z, Wang Y, Xu Q, Zhang N. LVBS: lightweight vehicular blockchain for secure data sharing in disaster rescue. IEEE Transactions on Dependable and Secure Computing, 2022, 19(1): 19–32
15. Li T, Wang H, He D, Yu J. Synchronized provable data possession based on blockchain for digital twin. IEEE Transactions on Information Forensics and Security, 2022, 17: 472–485
16. Cui L, Chen Z, Yang S, Ming Z, Li Q, Zhou Y, Chen S, Lu Q. A blockchain-based containerized edge computing platform for the internet of vehicles. IEEE Internet of Things Journal, 2021, 8(4): 2395–2408
17. Song J, Zhang P, Alkubati M, Bao Y, Yu G. Research advances on blockchain-as-a-service: architectures, applications and challenges. Digital Communications and Networks, 2022, 8(4): 466–475
18. Miao Q, Lin H, Hu J, Wang X. An intelligent and privacy-enhanced data sharing strategy for blockchain-empowered Internet of Things. Digital Communica-

tions and Networks, 2022, 8(5): 636–643
19. Guo R, Yang G, Shi H, Zhang Y, Zheng D. O^3-R-CP-ABE: An efficient and revocable attribute-based encryption scheme in the cloud-assisted IoMT system. IEEE Internet Things of Journal, 2021, 8(11): 8949–8963
20. Biryukov A, Khovratovich D, Pustogarov I. Deanonymisation of clients in bitcoin P2P network. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS). 2014, 15–29
21. Kappos G, Yousaf H, Maller M, Meiklejohn S. An empirical analysis of anonymity in zcash. In: Proceeding of the 27th USENIX Security Symposium. 2018, 463–477
22. Fujisaki E, Suzuki K. Traceable ring signature. In: Proceeding of the 10th International Conference on Practice and Theory in Public-Key Cryptography (PKC). 2007, 181–200
23. Bouakkaz S, Semchedine F. New efficient certificateless scheme-based conditional privacy preservation authentication for applications in VANET. Vehicular Communications, 2022, 34: 100414
24. Han L, Cao S, Yang X, Zhang Z. Privacy protection of VANET based on traceable ring signature on ideal lattice. IEEE Access, 2020, 8: 206581–206591
25. Peng X, Gu K, Liu Z, Zhang W. Traceable identity-based ring signature for protecting mobile iot devices. In: Proceeding of the 6th International Conference Data Mining and Big Data (DMBD). 2021, 158–166
26. Lu A, Li W, Yao Y, Yu N. TCABRS: an efficient traceable constant-size attribute-based ring signature scheme for electronic health record system. In: Proceeding of the 6th IEEE International Conference on Data Science in Cyberspace (DSC). 2021, 106–113
27. Fraser A, Quaglia E A. Report and trace ring signatures. In: Proceeding of the 20th International Conference Cryptology and Network Security (CANS). 2021, 179–199
28. Lai C, Ma Z, Guo R, Zheng D. Secure medical data sharing scheme based on traceable ring signature and blockchain. Peer-to-Peer Networking and Applications, 2022, 15(3): 1562–1576
29. Tang F, Pang J, Cheng K, Gong Q. Multiauthority traceable ring signature scheme for smart grid based on blockchain. Wireless Communications and Mobile Computing, 2021, 2021: 5566430:1–5566430:9
30. China o S C A. Public key crypto graphical gorithm sm2 based on elliptic curves - part 2: digital signature algorithm. Accessed 4 Mar. 2023. http://www.sca.gov.cn/sca/xwdt/2010-12/17/1002386/files/b791a9f908bb4803875ab6aeeb7b4e03.pdf, 2010
31. Bonneau J, Narayanan A, Miller A, Clark J, Kroll J A, Felten E W. Mixcoin: Anonymity for Bitcoin with accountable mixes. In: Christin N, Safavi-Naini R, eds, Proceeding of the 2014 International Conference Financial Cryptography and Data Security, (FC). 2014, 486–504
32. Valenta L, Rowan B. Blindcoin: Blinded, accountable mixes for Bitcoin. In: Brenner M, Christin N, Johnson B, Rohloff K, eds, Proceeding of the 2015 International Conference Financial Cryptography and Data Security, (FC). 2015, 112–126
33. Tran M, Luu L, Kang M S, Bentov I, Saxena P. Obscuro: A Bitcoin mixer using trusted execution environments. In: Proceedings of the 34th Annual Computer Security Applications Conference, (ACSAC). 2018, 692–701
34. Ruffing T, Moreno-Sanchez P, Kate A. Coinshuffle: Practical decentralized coin mixing for Bitcoin. In: Kutylowski M, Vaidya J, eds, Proceeding of 2014 the European Symposium on Research in Computer Security, (ESORICS). 2014, 345–364
35. Kappos G, Yousaf H, Maller M, Meiklejohn S. An empirical analysis of anonymity in Zcash. In: Enck W, Felt A P, eds, Proceeding of the 27th USENIX Security Symposium. 2018, 463–477
36. Rivest R L, Shamir A, Tauman Y. How to leak a secret. In: Proceeding of the 7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT). 2001, 552–565
37. Fujisaki E. Sub-linear size traceable ring signatures without random oracles. In: Proceeding of the 2011 Topics in Cryptology, (CT-RSA). 2011, 393–415
38. Au M H, Liu J K, Susilo W, Yuen T H. Secure id-based linkable and revocable-iff-linked ring signature with constant-size construction. Theoretical Computer Science, 2013, 469: 1–14
39. Teng D, Yao Y, Wang Y, Zhou L, Huang C. An SM2-based traceable ring signature scheme for smart grid privacy protection. In: Wireless Algorithms, Systems, and Applications, (WASA). 2022, 296–313
40. Scafuro A, Zhang B. One-time traceable ring signatures. In: Proceeding of the 26th European Symposium on Research in Computer Security (ESORICS). 2021, 481–500
41. Fan Q, He D, Luo M, Li X H D. Ring signature schemes based on sm2 digital signature algorithm. Journal of Cryptologic Research, 2021, 8: 710–723
42. Peng C, He D, Luo M, Li X H D. An identity-based ring signature scheme for sm9 algorithm. Journal of Cryptologic Research, 2021, 8: 724–734

43. Chen C, Wang L, Long Y, Luo Y, Chen K. A blockchain-based dynamic and traceable data integrity verification scheme for smart homes. Journal of Systems Architecture, 2022, 130: 102677
44. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D. Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS). 2007, 598–609
45. Wang Q, Wang C, Li J, Ren K, Lou W. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Proceeding of the 14th European Symposium on Research in Computer Security (ESORICS). 2009, 355–370
46. Dingledine R, Mathewson N, Syverson P F. Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium. 2004, 303–320
47. Zhang Y, Wang Q, Lu N, Shi W, Lei H. Traceable ring signature schemes based on SM2 digital signature algorithm and its applications in the evidence-storage system. In: Proceeding of the 4th International Conference on Blockchain and Trustworthy Systems. 2022, 122–133