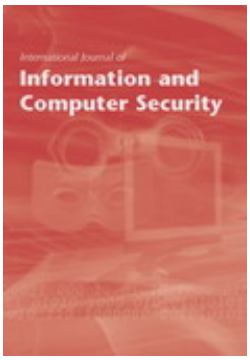


International Journal of Information and Computer Security

ISSN online: 1744-1773 - ISSN print: 1744-1765
https://www.inderscience.com/ijics

SDS-GS: a short group signature scheme enabling controlled traceability in secure medical data sharing

Xiaohui Yang, Xu Zhang

DOI: 10.1504/IJICS.2025.10071419

Article History:

Received:	17 March 2024
Last revised:	04 July 2024
Accepted:	05 July 2024
Published online:	05 September 2025

SDS-GS: a short group signature scheme enabling controlled traceability in secure medical data sharing

Xiaohui Yang and Xu Zhang*

School of Cyber Security and Computer,
Hebei University,
Baoding, China
Email: yxh@hbu.edu.cn
Email: zhangxuoo@stumail.hbu.edu.cn
*Corresponding author

Abstract: With the digitisation of information, the sharing of patient data in healthcare has increased dramatically, raising significant privacy concerns. We introduce a reliable dynamic group signature scheme for secure medical data sharing, ensuring secure and efficient data exchange and the identification of dishonest users. Our scheme also limits data managers' retrospective rights to prevent abuse. Security analysis demonstrates robust security and anonymity, while performance evaluations show high efficiency in healthcare data-sharing scenarios. Compared to other schemes, our scheme provides additional features such as privacy protection and secret sharing verification, and has higher scalability.

Keywords: group signatures; medical data sharing; controlled traceability; privacy protection.

Reference to this paper should be made as follows: Yang, X. and Zhang, X. (2025) 'SDS-GS: a short group signature scheme enabling controlled traceability in secure medical data sharing', *Int. J. Information and Computer Security*, Vol. 28, No. 1, pp.73–102.

Biographical notes: Xiaohui Yang is a Professor at the School of Cyber Security and Computer, Hebei University, China. He received his PhD from the University of Science and Technology of China in 2010. He has presided over or participated in more than 20 projects of National Key Research and Development Program, National Natural Science Foundation of China, National Science and Technology Support Program, Hebei Provincial Natural Science Foundation and horizontal research projects, published more than 40 academic papers, and won the third prise of Hebei Provincial Science and Technology Progress, and the second prise and third prise of Hebei Provincial Teaching Achievement. His primary research interests include distributed computing, privacy protection, information security and trusted computing.

Xu Zhang received his BS in Information and Computational Science from Huaqiao University, China, in 2020. He is currently working toward his Master's in Network and Information Security with the School of Cyber Security and Computer, Hebei University, China. His research interests include information security, digital signature and privacy protection.

1 Introduction

With the digitisation of information, the prevalence of digital information in healthcare has increased dramatically. Today, every hospital generates a large amount of intricate healthcare data, including electronic medical records and other medical information, daily (Kashani et al., 2021). However, as different hospitals adopt different medical data management systems, the lack of standardised data management methods has led to the prevalence of 'data silos'. The COVID-19 pandemic highlights the critical importance of data sharing for fostering swift learning and enabling impactful actions. Data sharing integrates data from all parties, enabling artificial intelligence technologies such as machine learning and deep learning (Liu et al., 2022; Bhattacharya et al., 2021) to analyse and extract critical information from large amounts of comprehensive data to improve overall healthcare (Apell and Eriksson, 2023).

In 2018, the National Health Commission of the People's Republic of China (NHC) issued the National Health Care Big Data Standards, Security and Service Management Measures (Trial) (NHC, 2018). The importance of establishing an open and sharing mechanism for health and medical big data has been clarified. Earlier laws, such as the Health Insurance Portability and Accountability Act (HIPAA) (HHS, 1996) in the USA and the Personal Information Protection and Electronic Documents Act (PIPEDA) (CANLII, 2000) in Canada, have focused heavily on protecting personal health information, yet privacy security remains a challenge in medical sharing applications (Zhang et al., 2022).

For example, Google Health builds its electronic health records on a cloud environment, providing users with self-managed services. The system allows users to share their own health information. However, this convenience can lead to loss of information or theft or even abuse of patient privacy. Besides, in 2018, the NHS, the UK's national healthcare agency and the world's largest single-payer healthcare system, suffered a massive data breach affecting 150,000 patients.

There are many initiatives around the world supporting the sharing of medical data, leading the way to open science while ensuring data security and respecting patients' privacy rights (Greene et al., 2022). Current research in healthcare data sharing focuses on the development of efficient and secure anonymisation systems, involving technologies such as blockchain (Azaria et al., 2016; Fan et al., 2018), attribute-based access control (Zhang et al., 2018), secure multiparty computation (Li et al., 2020), digital signatures (Carvalho et al., 2014; Aki, 1983), homomorphic encryption (Kocabaș and Soyata, 2016) and ciphertext policy attribute-based encryption (Ramesh et al., 2020). Among them, group signature technology (Chaum and van Heyst, 1991) is a highly scalable tool that is highly applicable to the field of medical data sharing, and has been widely used and proved its effectiveness (Chen et al., 2021).

Group signatures provide anonymity, authentication, and traceability. Group signatures allow a trusted entity or organisation to act as a group administrator and sign messages on behalf of users within the group. When verifying a group signature, the recipient of the message can only confirm that the signature was generated by a member of the group, without being able to determine the identity of the specific member. In addition, in the event of a dispute (Yu et al., 2012), the group administrator can open the group signature to reveal the true identity of the signer. In healthcare applications, a healthcare organisation is often faced with thousands of patients, and they are not only responsible for analysing and transmitting the patient's user data, but also responsible for storing and protecting the privacy of the data. Therefore, healthcare organisations are the natural group managers in the group signature scheme.

Healthcare organisations are burdened with heavy and urgent medical operations. One of the main challenges in this situation is to ensure that medical information is provided quickly and accurately to meet the needs of an emergency (Hulsen, 2020). On the other hand, institutions are just 'data custodians', in fact, data is the property of the patient (Hulsen et al., 2019). Although patients have a degree of trust in hospitals, it is limited to meeting basic medical needs. Patients do not want to grant hospitals absolute retroactive rights, allowing them to break anonymity at will.

The key to sharing medical data is mutual trust. However, most of the existing group signature schemes only focus on improving signature efficiency and ignore the trust crisis caused by the trace power of the group administrator. In this paper, we propose a short group signature scheme enabling controlled traceability for secure medical data sharing, named SDS-GS, to address the above challenges. The main contributions are highlighted as follows.

1 We design a dynamic group signature scheme enabling controlled traceability in large-scale healthcare data sharing. In our scheme, each healthcare organisation holds the right to turn on tracing, but needs to cooperate with other healthcare organisations for tracing. Multiple agencies working together on the trace process is safer and gives patients a stronger foundation of trust than a single agency dictatorial implementation.

2 We propose a short group signature technique for efficient and accurate transmission of medical information. Traditional threshold signature-based schemes for performing verification usually use complex zero-knowledge proofs, whereas our scheme uses public keys to perform verification directly, which has higher verification efficiency, shorter signature length, and more efficient use of resources in large-scale medical data sharing environment.

The remainder of this article is organised as follows. Section 2 reviews the related work, and Section 3 explains some preliminaries used in the proposed scheme. The system model and security model are defined in Section 4. Section 5 gives the detailed scheme. The security analysis is given in Section 6. Performance analysis is presented in Section 7, and Section 8 finally concludes this article.

2 Related work

Group signatures, initially proposed by Chaum and van Heyst (1991), are used to realise anonymous communication among users, but the initial group signature scheme does not support the joining and exiting of group members, which greatly limits its application scenarios. Thereafter, Bellare et al. (2005) proposed dynamic group signatures to overcome this limitation, but the efficiency of the scheme has been criticised. In view of the application prospect of group signature, the research on efficiency improvement of dynamic group signature has never stopped. Recently, Camenisch et al. (2020) proposed an efficient and secure dynamic group signature scheme based on Pointcheval-Sanders (PS) signatures (Pointcheval and Sanders, 2018). Based on this, Clarisse and Sanders (2020) proposed a new and shorter dynamic group signature scheme. This scheme combines PS signatures and Fuchsbauer-Hanser-Slamanig (FHS) signatures (Fuchsbauer et al., 2019), which greatly reduces the complexity of signatures. Inspired by these schemes, Wang (2022) developed a simplified variant of dynamic group signatures, which further reduces the computational overhead, but unfortunately, the scheme does not support tracing users. The work of these researchers is the foundation of our approach, and we build on their framework to further expand signature capabilities while simplifying signatures and optimising efficiency.

The continuous development of dynamic group signatures in recent years makes it unique in protecting privacy and efficient communication. In addition, because the structure of dynamic group signatures is well suited to the medical field, many scholars have used dynamic group signatures to develop anonymity systems in the medical field (Yu and Hou, 2014; Umamageswari and Suresh, 2014; Olakanmi and Odeyemi, 2023; Chen et al., 2021; Jiang et al., 2015; Su et al., 2022; Li et al., 2023); however, many of these schemes (Yu and Hou, 2014; Umamageswari and Suresh, 2014; Olakanmi and Odeyemi, 2023; Chen et al., 2021) do not support real-time tracking of users. To address this shortcoming, Su et al. (2022) attempt to perform user authentication by combining BU-VLR group signature and zero-knowledge proof techniques to achieve user anonymity and real-time traceability. And Li et al. (2023) design an anonymous system by combining group signatures and blind signatures, which not only achieves real-time traceability but also brings stronger anonymity. In the current medical environment, group signatures are often used as the underlying architecture of data sharing frameworks, which requires high signature efficiency. However, the schemes proposed by Su et al. (2022) and Li et al. (2023) have high computational overheads, which may not be acceptable to users. Our scheme not only supports real-time traceability and strong anonymity but also significantly reduces computational overhead by implementing an efficient public key verification algorithm. Moreover, these existing schemes do not consider the mutual trust between group members and group administrators, granting group administrators absolute control over group members, which poses a risk of power abuse.

To limit the privileges of group administrators, researchers have taken different measures. For example, Kiayias and Yung (2006) divide the authentication and tracing capabilities of group administrators into two different entities: issuers and arbiters. The former is responsible for authenticating new users and ensuring anonymity, while the latter is responsible for tracking malicious users and ensuring traceability. However, the scheme still suffers from the risk problem that the arbiter has too much trace power. To address this limitation, Li et al. (2009) combined group signatures with

threshold cryptography to achieve threshold traceability. They exploit the property of (n, t)-threshold signatures to assign the ability to trace the actual signer to n group members, such that any subset of no less than t members can collaboratively reconstruct the secret to reveal the identity of the signer, and the scheme can withstand a conspiracy attack by at most $t-1$ malicious signers. Blömer et al. (2016) similarly combine group signatures and threshold cryptography, and they constructed a simplified version of group signatures with distributed traceability using zero-knowledge proofs, achieving stronger and more desirable complete anonymity for CCA. Later, Lu et al. (2020) designed a group signature with distributed traceability based on the q-strong Diffie-Hellman assumption, which also ensures full CCA anonymity. However, all these schemes fail to fully consider integration with the medical domain. As a result, even though the trace rights of group administrators are successfully restricted, the efficiency of tracing users remains very low, making the system unable to meet the requirements of large-scale medical applications. Further research is needed to develop more efficient and practical dynamic group signature schemes for the privacy and security of medical data sharing. Our scheme limits the administrator's trace power through the secret sharing of multiple medical institutions. Unlike the above schemes, our approach fully considers the integration with the medical field, ensuring high efficiency of the trace algorithm while effectively safeguarding the interests of both patients and institutions. In practical medical applications, institutions are responsible for patient data. Therefore, in our scheme, the trace process for a patient can only be initiated by the institution that holds the patient's data, while other institutions have the right to restrict but not decide on the trace process.

In conclusion, considering the limitations of existing schemes, it is very meaningful to design a secure, efficient and practical dynamic group signature scheme for medical data sharing.

3 Preliminaries

3.1 PS signature

Pointcheval and Sanders (2018) introduced a randomisable signature scheme. It can derive a re-randomised version σ' of any valid signature σ. Without knowledge of the corresponding message, no one can connect σ' and σ. A PS signature should include a tuple of polynomial-time algorithms (Setup, KeyGen, Sign, Verify):

- $Setup\left(1^{\lambda}\right) \rightarrow (params)$: It takes the security parameter 1^{λ} as input and generates system parameters $params$, which includes public parameters $pp \leftarrow (G_1, G_2, G_T, e)$, generator elements $(g, \hat{g}) \in G_1 \times G_2$, and a pair of parameters computed from random scalar x: $\left(X, \hat{X}\right) \leftarrow (g^x, \hat{g}^x)$.
- $Keygen\,(pp) \rightarrow (y, (sk, pk))$: It takes a random scalar y and computes the key pair $(sk, pk) \leftarrow \left(g^y, \hat{Y} = \hat{g}^y\right)$.
- $Sign\,(sk, msg) \rightarrow (\sigma_1, \sigma_2)$: It generates a random scalar r and computes the signature pair $(\sigma_1, \sigma_2) \leftarrow (g^r, X^r \cdot g^{r \cdot y \cdot msg})$ on the message msg.

- $Verify\,(pk, msg, (\sigma_1, \sigma_2)) \rightarrow (0\ or\ 1)$: If it satisfies $e(\sigma_1, \hat{X}\hat{Y}^{msg}) = e(\sigma_2, \hat{g})$, then the verification passes.

The PS signature scheme has been proven to be EUF-CMA secure under the LRSW assumption (Lysyanskaya et al., 2000) customised for type-3 pairings.

3.2 FHS signature

Fuchsbauer et al. (2019) introduced an equivalence-based signature scheme. For a tuple $(M_1, ..., M_n) \in G_1^n$, if there exists a scalar r such that $N_i = M_i^r$, for $i \in [1, n]$, then $(N_1, ..., N_n)$ is considered equivalent to $(M_1, ..., M_n)$. In this work, we will only consider the case where $n = 2$. A FHS signature should include a tuple of polynomial-time algorithms (Setup, KeyGen, Sign, Verify):

- $Setup\left(1^\lambda\right) \rightarrow (params)$: It takes the security parameter 1^λ as input and generates system parameters $params$, which includes public parameters $pp \leftarrow (G_1, G_2, G_T, e)$, and generator elements $(g, \hat{g}) \in G_1 \times G_2$.
- $KeyGen\,(pp) \rightarrow (sk, pk)$: It generates random scalars α_1 and α_2, and outputs $sk \leftarrow (\alpha_1, \alpha_2)$, $pk \leftarrow (\hat{A}_1 = \hat{g}^{\alpha_1}, \hat{A}_2 = \hat{g}^{\alpha_2})$.
- $Sign\,(sk, (M_1, M_2)) \rightarrow (\tau_1, \tau_2, \hat{\tau})$: It generates a random scalar t and outputs the signature triplet $(\tau_1, \tau_2, \hat{\tau}) \leftarrow ((M_1^{\alpha_1} \cdot M_2^{\alpha_2})^t, g^{1/t}, \hat{g}^{1/t})$ on the message (M_1, M_2).
- $Verify\,(pk, (M_1, M_2), (\tau_1, \tau_2, \hat{\tau})) \rightarrow (0\ or\ 1)$: If it satisfies $e(\tau_1, \hat{\tau}) = e(M_1, \hat{A}_1) \cdot e(M_2, \hat{A}_2)$ and $e(\tau_2, \hat{g}) = e(g, \hat{\tau})$, then the verification passes.

The signature triplet $(\tau_1, \tau_2, \hat{\tau})$ is only valid for (M_1, M_2), but based on the equivalence relation, we can efficiently generate a valid signature triplet $(\tau_1^r, \tau_2, \hat{\tau})$ on its equivalent items (M_1^r, M_2^r). Furthermore, by applying the FHS signature approach described above, we can generate a random scalar t' and re-randomise the signature triplet on the equivalent class, resulting in $(\tau_1', \tau_2', \hat{\tau}') \leftarrow (\tau_1^{r \cdot t'}, \tau_2^{1/t'}, \hat{\tau}^{1/t'})$. The FHS signature scheme satisfies EUF-CMA security.

3.3 Computational assumptions

LRSW assumption (Lysyanskaya et al., 2000): Let G be a cyclic group of prime order p, with a generator g. For $(g,\ X = g^x,\ Y = g^y)$, with $x, y \xleftarrow{\$} Z_p^*$, we define the oracle $\mathcal{O}(m)$ on input $m \in Z_p$ that chooses a random $h \in G^*$ and outputs the triple $T = (h, h^y, h^{x+mxy})$. Given (g, X, Y) and unlimited access to this oracle $\mathcal{O}$, no adversary can efficiently generate such a triple for a new scalar m^*, not asked to $\mathcal{O}$.

(t, ε)-DLP assumption: Let G be a cyclic group of prime order p, with a generator g. For $g' = g^x$, with $x \xleftarrow{\$} Z_p^*$, there is no algorithm $\mathcal{A}$ with at least ε probability, find a scalar $a \in Z_p^*$ such that $g^a = g'$ within running at a maximum polynomial time t.

3.4 Security notion

EUF-CMA security: The standard security notion for a signature scheme is existential unforgeability under chosen message attacks (EUF-CMA, Goldwasser et al., 1988) which means that it is hard, even given access to a signing oracle, to output a valid pair (m, ε) for a message m never asked to the signing oracle. It is defined using the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

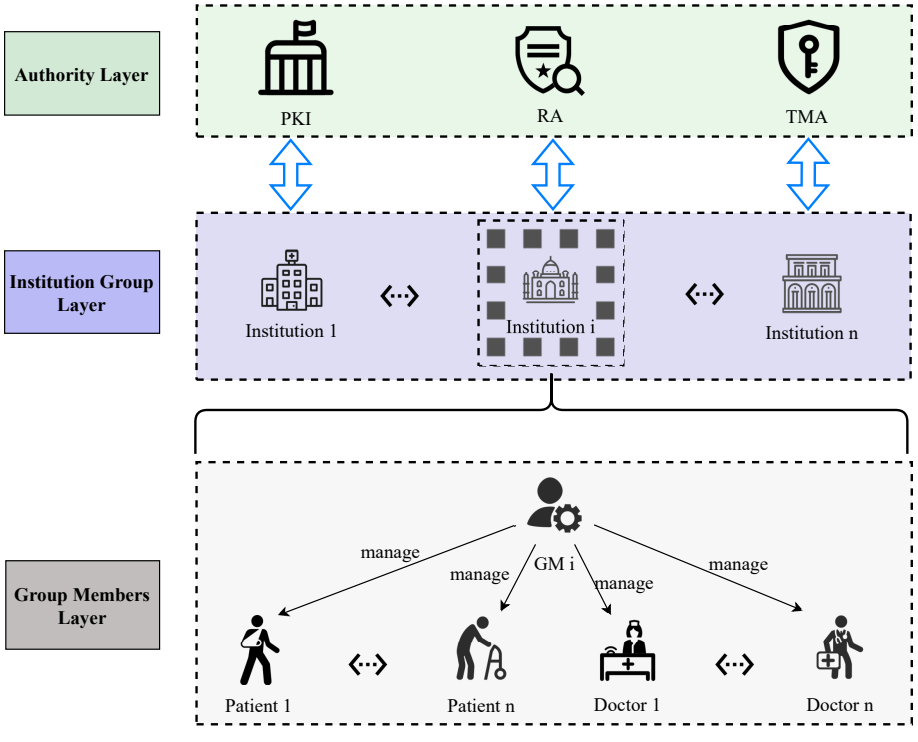
- *Setup:* $\mathcal{C}$ runs the $Setup$ and the $Keygen$ algorithms to obtain sk and pk. $\mathcal{A}$ is given the public key pk.
- *Query:* $\mathcal{A}$ adaptively requests signatures on at most q messages $m_1, ..., m_q$. $\mathcal{C}$ answers each query by returning $\sigma_i \leftarrow Sign(sk, m_i)$.
- *Forgery:* $\mathcal{A}$ outputs a message-signature pair $(\bar{m}, \bar{\sigma})$ and wins the game if $Verify(pk, \bar{m}, \bar{\sigma}) = 1$ and if $\bar{m} \neq m_i \forall i \in [1, q]$.

A signature scheme is EUF-CMA secure if no probabilistic polynomial-time adversary $\mathcal{A}$ can win this game with non-negligible probability.

4 System and security model

This section provides an initial introduction to the system model and security model, including an introduction to the entities and security requirements.

Figure 1 Entities and their layers in our model (see online version for colours)



4.1 System model

As shown in Figure 1, we design a distributed data sharing scheme for e-health services. There exist six entities in our model.

- *Public key infrastructure* (PKI) serves as an issuer to release public-private key pairs to the user. Its main purpose is to issue, certify, and manage digital certificates.
- *Registration authority* (RA) is responsible for reviewing user eligibility. Users are required to be confirmed by the RA when joining and exiting the system to ensure security.
- *Transmission management authority* (TMA) is a semi-honest entity that acts as the transfer in the group signature scheme. It works in the join phase and revoke phase, it is responsible for obfuscating, distributing, and validating communication messages between users and group managers.
- *Institutions* refer to a variety of healthcare facilities such as hospitals, clinics and nursing homes. Each institution forms its own group. Institutions have the capability to dynamically join the system and engage in cross-institution data sharing.
- *Group manager* (GM) acts as an agent of the institution, it is responsible for managing the entire group. GM holds the group public-private key pair(gpk, gsk) and the tracing key pair(osk, opk). When danger occurs, GM can perform *Open* algorithm to revoke the group members.
- *User*, includes patients and doctors, is the member of group. Group member is the basic unit within a group. All group members together form a group, which is used to ensure the anonymity and security of the group signature scheme. They are also the owners of the medical data.

4.2 Security model

Bellare et al. (2003) summarised two key attributes: full anonymity and full traceability. In addition, correctness and non-frameability are also essential attributes in the security requirements of signatures. The detailed definition and proof of security attributes can be found in Section 6. The oracles used to prove the security property is defined as follows:

- $\mathcal{O}Add(i)$ is an oracle used to add a new user U_i. It performs a random selection of $y_i \in Z_p$, generates a user key pair $(usk_i = y_i, upk_i = \hat{g}^{y_i})$, and returns the key pair.
- $\mathcal{O}Join(i)$ is an oracle that executes the *Join* algorithm. It assists the new user U_i in performing the *Join* algorithm to join the system.
- $\mathcal{O}Sign(gusk_i, msg)$ is an oracle that executes the *Sign* algorithm. It returns $Sign(gusk_i, msg)$.

- $\mathcal{O}\mathrm{Rec}(N_1, N_2)$ is an oracle used to recover credentials. It randomly selects $\alpha, \gamma \in Z_p$ and returns $upk = N_2 \cdot N_1^{-\alpha \cdot \gamma}$.
- $\mathcal{O}Ch(i_0, i_1, msg)$ is an oracle that requires input of the indices of two honest users. It randomly selects $i_b \in \{i_0, i_1\}$ and returns $Sign(params, gusk_{i_b}, msg)$.
- $\mathcal{O}Open(\sigma, osk, msg)$ is an oracle used for tracing signers. It run $Open$ algorithm, and if the $Open$ algorithm successfully opens the signature, it returns upk; otherwise, it returns $\perp$.

Table 1 System parameter notion

Notation	*Meaning*
Σ	The ElGamal algorithm
p	The large prime
e	The bilinear map
G_1, G_2	Two cyclic groups
Z_p^*	The group of integers with prime order p
H	The secure hash function
1^λ	Securty parameters
$params$	The system parameters
(tsk, tpk)	The public-private key pair of TMA
(rsk, rpk)	The public-private key pair of RA
(gsk, gpk)	The group public-private key pair of GM
(osk, opk)	The tracing public-private key pair of GM
(usk, upk)	The public-private key pair of user
$gusk$	The user's group member private key
σ	Group signature
msg	The message
$MapList$	Mapping table for recording secret storage relationships
$[REG]$	User registration list stored in the group
$[SNQ]$	A randomly ordered sequence of nodes used for secret sharing
C_1	The collection of encryption parameters for tracing
rc	Credentials used to encrypt and decrypt the identity of the signing user
H_1	The common hash identity of a set of secret sharing nodes.

5 The short group signature construction

This section details the composition of SDS-GS scheme, including the overall framework and specific algorithms.

5.1 Concrete construction

The system model design is shown in Figure 2. The overall process is divided into four phrases, which are shown as follows:

Phase 1: System initialisation

System runs Algorithm 1 (*Setup*) to initialise. All entities in the system generate corresponding public-private key pairs. TMA, RA and GM respectively perform Algorithms 2 (*TMAKeyGen*), 3 (*RAKeyGen*) and 4 (*GMKeyGen*).

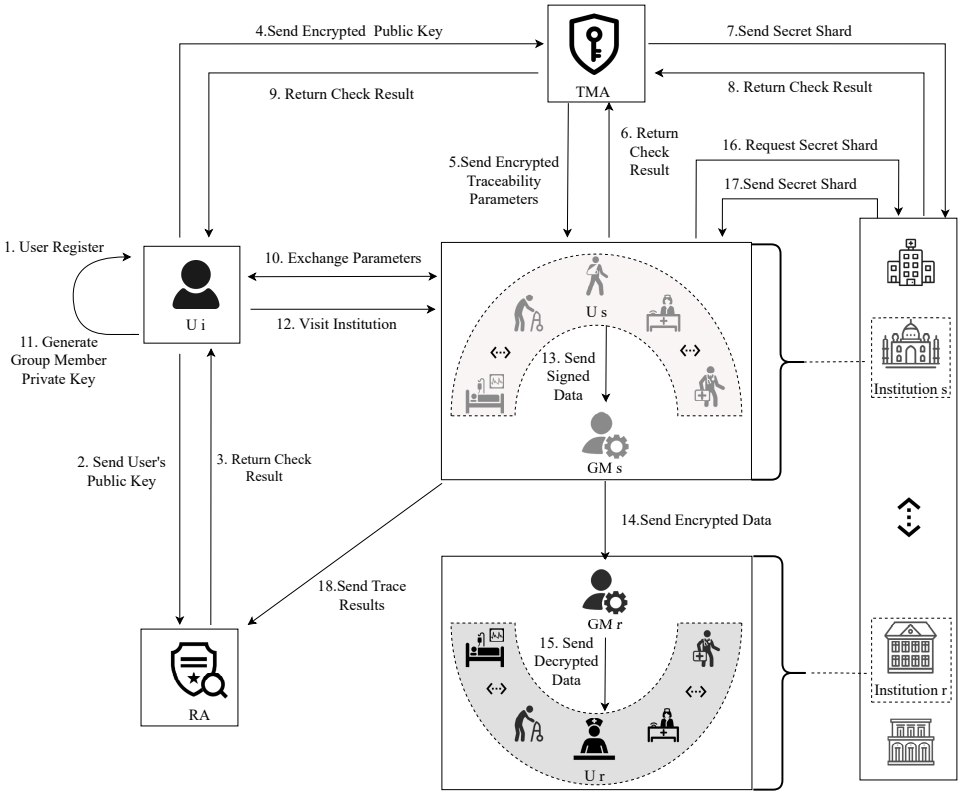
Phase 2: Data generation

User performs Algorithm 5 (*UserKeyGen*) to generate the public-private key pair (usk, upk) (1 – user register), with upk regarded as the user's real identity. Subsequently, the user executes Algorithm *Join* to join the system. The join process is divided into four sub-processes.

Firstly, the user executes Algorithm 6 (*Join.AuthenticateUser*), sends his/her public key and identity information to RA (2 – send user's public key). RA verifies the user's qualification and returns the result (3 – return check result).

Then user executes Algorithm 7 (*Join.PrepareDistribute*). First, the user sends the encrypted user information to TMA (4 – send encrypted public key). TMA uses the RA's public key to verify information, generates relevant parameters for tracing, and sends the encrypted parameters to GM_s (5 – send encrypted traceability parameters). The verification result is returned after GM_s authentication (6 – return check result).

Figure 2 System model



After the verification is passed, TMA executes Algorithm 8 (*Join.SecretDistribute*) to secretly distribute the traceable parameters. TMA sends the secret shards to GM nodes in [SNQ] (7 – send secret shard). Each GM node verifies secret shard and returns the result to TMA (8 – return check result). Once the TMA receives the verification results from all nodes, it returns the signal to the user (9 – return check result).

Finally, the user performs Algorithm 9 (*Join.GenerateGusk*) to generate the group member private key. After the user and GM calculate and exchange parameters with each other (10 – exchange parameters), the user computes and generates its own group member private key (11 – generate group member private key). At this time, the user officially joins the system, can access the medical institution, and generate the corresponding medical data (12 – visit institution).

Phrase 3: Data sharing

System allows data sharing across organisations. Sender U_s in institution s performs Algorithm 10 (*Sign*) to generate a group signature σ on medical data. Then U_s send the data and group signature to GM_s in institution s (13 – send signed data). GM_s encrypts data with the group public key of GM_r. Then, GM_s sends these encrypted data with group signature to GM_r (14 – send encrypted data). GM_r first performs Algorithm 11 (*Verify*) to verify group signature. Then, GM_r decrypts these data, and send these data to the receiver U_r in institution r (15 – send decrypted data). During the sharing process of the system, each GM can evaluate and interrupt the process.

Phase 4: Data tracking

When an emergency occurs, such as a medical emergency or a malicious attack on the system, the system needs to trace the real signer of some certain data. GM executes Algorithm 12 (*Open*) to open the group signature for tracing.

GM first opens the signature to find the corresponding entry in the registration list and sends the request to nodes in $[SNQ]$ (16 – request secret shard). The GM node in $[SNQ]$ evaluates the trace request and sends the secret shard (17 – send secret shard). After GM collects enough secret shards, it decrypts the user's identity information.

After tracing, GM_s send the traced user's identity information to RA (18 – send trace results), which runs Algorithm 13 (*Judge*) to confirm the validity of the tracing result. Then, RA processes the user's identity information accordingly, such as adding the malicious user's information to the $blacklist$.

5.2 Details of algorithms

For ease of exposition, see Table 1 for the meaning of the symbols. In some algorithms, U_i represents a new user, and GM_s represents the administrator of the group that U_i wants to join. The ElGamal algorithm is safe and efficient enough to meet the requirements of medical systems. In SDS-GS scheme, we use ElGamal algorithm to encrypt and decrypt data, and the ElGamal algorithm is notated as Σ. The details of algorithms are shown below.

Algorithm 1 Setup()

Input: 1^{λ}.
Output: $params$.
1: Select three cyclic groups G_1, G_2 and G_T, where p is a prime order, and g, $\hat{g}$ are generators of G_1, G_2 respectively;
2: Define a asymmetric bilinear map $e : G_1 \times G_2 \rightarrow G_T$,
3: $x \xleftarrow{\$} Z_p^*$, $X \leftarrow g^x$, $\hat{X} \leftarrow \hat{g}^x$;
4: Define a secure hash functions $H : \{0,1\}^* \rightarrow Z_p$;
5: $params \leftarrow \left(G_1, G_2, G_T, e, g, \hat{g}, X, \hat{X}, H\right)$;
6: **return** $params$.

Algorithm 2 TMAKeyGen()

Input: $params$.
Output: (tsk, tpk).
1: $\xi \xleftarrow{\$} Z_p^*$, $TK_1 \leftarrow g^{\xi}$, $TK_2 \leftarrow \hat{g}^{\xi}$;
2: $tsk \leftarrow \xi$, $tpk \leftarrow (TK_1, TK_2)$;
3: **return** (tsk, tpk).

Algorithm 3 RAKeyGen()

Input: $params$ and tpk.
Output: (rsk, rpk), $UserList$ and $BlackList$.
1: $o \xleftarrow{\$} Z_p^*$
2: $rsk \leftarrow o$, $rpk \leftarrow g^{1/o}$, $rupk_0 \leftarrow (TK_2)^{rsk}$
3: $UserList \leftarrow \{rupk_0\}$, $BlackList \leftarrow \varnothing$
4: **return** (rsk, rpk), $UserList$ and $BlackList$.

Algorithm 4 GMKeyGen()

Input: $params$.
Output: (gsk, gpk) and (osk, opk).
1: $\varsigma, \alpha \xleftarrow{\$} Z_p^*$
2: $osk \leftarrow \varsigma$, $opk \leftarrow \hat{g}^{\varsigma}$
3: $\hat{A} \leftarrow \hat{g}^{\alpha}$, $\hat{B} \leftarrow \hat{X}^{\alpha}$, $C \leftarrow g^{\alpha}$;
4: $gsk \leftarrow \alpha$, $gpk \leftarrow (\hat{A}, \hat{B}, C)$
5: **return** (gsk, gpk) and (osk, opk).

Algorithm 5 UserKeyGen()

Input: $params$.
Output: (usk, upk).
1: $\varphi \xleftarrow{\$} Z_p^*$
2: $usk \leftarrow \varphi$, $upk \leftarrow \hat{g}^{\varphi}$
3: **return** (usk, upk).

Algorithm 6 Join.AuthenticateUser()

Input: $params$, upk_i and rsk.
Output: Valid or invalid.
The algorithm involves two entities: U_i and RA.
1. U_i do:
$U_i.SEND(upk_i) \rightarrow RA$
2. RA do:
if upk_i not in $BlackList$ **then**
$rupk_i \leftarrow rupk_{i-1}\,(upk_i)^{rsk}$
$RA.ADD(rupk_i) \rightarrow UserList$
$RA.SEND(confirmation) \rightarrow U_i$
return valid
else
return invalid
end if

Algorithm 7 Join.PrepareDistribute()

Input: $params$, upk_i, (tsk, tpk), opk, rpk and $UserList$.
Output: $[SNQ]$ or $\perp$.
The algorithm involves three entities: U_i, GM_s and TMA.
1. U_i do:
$\beta, \gamma \xleftarrow{\$} Z_p^*$, $N_1 \leftarrow \hat{g}^{\beta/\gamma}$, $N_2 \leftarrow upk_i \cdot \hat{A}^{\beta}$, $Q_1 \leftarrow \hat{g}^{\beta}$, $rc \leftarrow \gamma$, $h_1 \leftarrow H\,(N_1||N_2||N_3)$
$O \leftarrow \Sigma.encrypt(TK_1, (N_1, N_2, Q_1, h_1, rc))$
$U_i.SEND(O) \rightarrow TMA$
2. TMA do:
$(N_1, N_2, Q_1, h_1, rc) \leftarrow \Sigma.decrypt\,(tsk, O)$
if $N_1^{rc} = Q_1$ and $h_1 = H\,(N_1||N_2||Q_1||rc)$ **then**
$Q_2 \leftarrow Q_1^{1/tsk}$, $n, t \xleftarrow{\$} Z_p^* \qquad t < n,\ n \leq TOTAL\ GM\ NODES$
$[SNQ] \xleftarrow{\$} \{ALL \quad GM \quad NODES\}, \quad [SNQ] = [GM_1, GM_2, GM_3, \cdots, GM_n]$
$ST \leftarrow TMA.TIMESTAMP()$
$H_1 \leftarrow H\,(SNQ \;||\; ST)$
$C_1 \leftarrow \Sigma.encrypt(opk, (t, H_1, [SNQ], N_1, N_2))$
$h_2 \leftarrow H\,(N_2||Q_2||C_1)$
$TMA.SEND(C_1, N_2, Q_2, h_2) \rightarrow GM_s$
else
return $\perp$
end if
3. GM_s do:
if $h_2 = H\,(N_2||Q_2||C_1)$ and $e\left(TK_1, Q_2^{gsk}\right) \cdot e\left(rpk, rupk_i \cdot rupk_{i-1}^{-1}\right) = e\,(g, N_2)$ **then**
$GM_s.SEND(confirmation) \rightarrow TMA$
return $[SNQ]$
else
return $\perp$
end if

Algorithm 8 Join.SecretDistribute()

Input: $params$, H_1, rc and $MapList$.
Output: Success or failure.
The algorithm involves two types of entities: TMA, GM_i in $[SNQ]$.
1. TMA do:
for GM_i *in* $[SNQ]$, $i = 1, 2, ..., t-1$ **do**
$A_i \xleftarrow{\$} GF(p)$
$f(x) \leftarrow rc + A_1 x + A_2 x^2 + ... + A_{t-1} x^{t-1}$
$sk_i \leftarrow (i, f(i))$, $H_{2_i} \leftarrow H(H_1 || i)$, $A_i \leftarrow g^{sk_i \xi}$
$TMA.SEND(H_{2_i}, A_i, sk_i) \rightarrow GM_i$
end for
2. Each GM_i do:
if $e(A_i, \hat{g}) = e\left(g^{sk_i}, TK_2\right)$ **then**
$GM_i.ADD(H_{2_i}, sk_i) \rightarrow MapList$
$GM_i.SEND(confirmation) \rightarrow TMA$
end if
3. TMA do:
if TMA receives all confirmation from GM_i **then**
$TMA.SEND(confirmation) \rightarrow U_i$
return Success
else
return Failure
end if

Algorithm 9 Join.GenerateGusk()

Input: $params$ and C_1.
Output: $gusk$.
The algorithm involves two entities: U_i and GM_s.
1. U_i do:
$u, y \xleftarrow{\$} Z_p^*$
$t_1 \leftarrow g^u$, $t_2 \leftarrow g^{uy}$, $t_3 \leftarrow \hat{g}^y$, $t_4 \leftarrow g^y$
$U_i.SEND(t_1, t_2, t_3) \rightarrow GM_s$
2. GM_s do:
$GM_s.ADD(C_1, t_1, t_2, t_3) \rightarrow REG$
$t \xleftarrow{\$} Z_p^*$
$T_1^{'} \leftarrow (t_1 t_2)^{\alpha t}$, $T_2 \leftarrow g^{1/t}$, $\hat{T} \leftarrow \hat{g}^{1/t}$
$GM_s.SEND(T_1^{'}, T_2, \hat{T}) \rightarrow U_i$
3. U_i do:
$T_1 \leftarrow \left(T_1^{'}\right)^{1/u}$
$gusk \leftarrow \left(T_1, T_2, \hat{T}, t_4\right)$.
return $gusk$.

Algorithm 10 Sign()

Input: $gusk_i$ and msg.
Output: σ on msg.

$s, r \xleftarrow{\$} Z_p^*$
$T_1^* \leftarrow T_1^{s \cdot r}$, $T_2^* \leftarrow T_2^{1/s}$, $\hat{T}^* \leftarrow \hat{T}^{1/s}$, $\sigma_1 \leftarrow g^r$, $\sigma_2 \leftarrow X^r / H(T_1^* || T_2^* || \hat{T}^* || \sigma_1 || msg) \cdot t_4^r$
$\sigma \leftarrow \left(T_1^*,\ T_2^*,\ \hat{T}^*,\ \sigma_1,\ \sigma_2\right)$
return σ;

Algorithm 11 Verify()

Input: msg, gpk and σ on msg.
Output: Valid or invalid.

$c' \leftarrow H(T_1^* || T_2^* || \hat{T}^* || \sigma_1 || msg)$
$R' \leftarrow e(\sigma_1 \sigma_2 T_2^*, \hat{A}) \cdot e(\sigma_1, \hat{B}^{-1/c'})$
if $R' == e(T_1^* C, \hat{T}^*)$ **then**
  return valid
else
  return invalid
end if

Algorithm 12 Open()

Input: $params$, msg, σ on msg, (osk, opk), (gsk, gpk), $[REG]$, $MapList$, gpk_i and tpk;
Output: upk.

The algorithm is executed by GM_s and involves two types of entities: GM_s and GM_i in $[SNQ]$.
1. GM_s do:
  $S \leftarrow e(\sigma_2, \hat{g}) \cdot e(\sigma_1, \hat{X}^{-1/H(T_1^* || T_2^* || \hat{T}^* || \sigma_1 || msg)})$
for $REG[i] = (C_{1_i}, t_{1_i}, t_{2_i}, t_{3_i})$ in REG **do**
  if $S = e(\sigma_1, t_{3_i})$ **then**
    $(H_1, [SNQ], N_1, N_2) \leftarrow \Sigma.decrypt(osk, C_1)$
  end if
end for
for each GM_i in $[SNQ]$ **do**
  $H_{2_i} \leftarrow H(H_1 || i)$
  $GM_s.SEND(H_{2_i}) \rightarrow GM_i$
end for
2. Each GM_i received H_{2_i} do:
if GM_i agree open requests **then**
  $sk_i \leftarrow GM_i.Search(MapList, H_{2_i})$
  $B \leftarrow g^{sk_i \alpha_i} A$
  $GM_i.SEND(B, sk_i) \rightarrow GM_s$
else
  return $\perp$
end if

3. GM_s do:
if $e(B, \hat{g}) = e(g^{sk_i}, \hat{A}_i \cdot TK_2)$ **then**
$GM_s.ACCEPT(sk_i)$
else
$GM_s.DISCARD(sk_i)$
end if
if $number\ of\ accepted\ sk_i \geq t-1$ **then**
$(x_i, f(x_i)) \leftarrow sk_i \quad i = 1, 2, ..., t-1$
$l_i(x) \leftarrow \prod_{j=0, i \neq j}^{n} \frac{x - x_j}{x_i - x_j} \quad i = 1, 2, ..., t-1$
$f(x) \leftarrow \sum_{i=1}^{t-1} f(x_i) l_i(x)$
$rc \leftarrow f(0)$
$upk \leftarrow N_2 \cdot N_1^{-gsk \cdot rc}$
end if
return upk.

Algorithm 13 Judge()

Input: msg, $params$, gpk, R_i and σ on msg.
Output: Valid or invalid.
1: $c' \leftarrow H(T_1^* || T_2^* || \hat{T}^* || \sigma_1 || msg)$;
2: $R_1' \leftarrow e(\sigma_1 \sigma_2 T_2^*, \hat{A}) \cdot e(\sigma_1, \hat{B}^{-1/c'})$
3: $R_2' \leftarrow e(\sigma_2, \hat{g}) \cdot e\left(\sigma_1, \hat{X}^{-1/c'}\right)$
4: **if** $R_1' == e(T_1^* C, \hat{T}^*)$ and $R_2' == e(\sigma_1, t_3)$ **then**
5: **return** valid
6: **else**
7: **return** invalid
8: **end if**

6 Security analysis

In this section, we first establish the correctness of our group signature scheme. Subsequently, we prove that the scheme satisfies full anonymity, full traceability, and non-frameability. Finally, we analysed how the SDS-GS scheme achieves information leakage prevention.

6.1 Correctness proof

Correctness: Signatures generated by honest group members must be verifiable correctly, and these group signatures must be traceable back to the original signers. This implies that the verification algorithm should confirm the authenticity of these signatures without error. Additionally, the group signatures should possess the property of traceability, allowing an authorised entity to reliably trace the signatures back to their original signers, ensuring accountability and non-repudiation within the group. The correctness of the SDS-GS scheme be evaluated from the following four aspects:

1 *Correctness proof for the verification about secret sharing is shown below:* In Algorithms *Join* and *Open*, SDS-GS scheme employs the Shamir secret sharing scheme. The correctness proof as below:

$$
\begin{aligned}
&e\left(A,\hat{g}\right)=e\left(g^{sk_i\xi},\hat{g}\right)=e\left(g^{sk_i},\hat{g}^{\xi}\right)=e\left(g^{sk_i},TK_2\right)\\
&e\left(B,\hat{g}\right)=e\left(g^{sk_i\alpha_i}A,\hat{g}\right)=e\left(g^{sk_i\alpha_i}g^{sk_i\xi},\hat{g}\right)=e\left(g^{sk_i},\hat{g}^{\alpha_i+\xi}\right)\\
&\qquad=e\left(g^{sk_i},\hat{A}_i\cdot TK_2\right)
\end{aligned}
$$

2 *Correctness proof for the algorithm join is shown below:*

$$
\begin{aligned}
&N_1^{rc}=\left(\hat{g}^{\beta/\gamma}\right)^{\gamma}=\hat{g}^{\beta}=Q_1\\
&e\left(TK_1,\ Q_2^{gsk}\right)e\left(rpk,\ rupk_i\cdot rupk_{i-1}^{-1}\right)\\
&\quad=e\left(g^{\xi},N_3^{\alpha/\xi}\right)e\left(g^{1/o},rupk_{i-1}\left(upk_i\right)^{o}rupk_{i-1}^{-1}\right)\\
&\quad=e\left(g^{\xi},\hat{g}^{(\alpha\beta)/\xi}\right)e\left(g^{1/o},\left(upk_i\right)^{o}\right)\\
&\quad=e\left(g,\hat{g}\right)^{\alpha\beta+\varphi_i}\\
&\quad=e\left(g,upk_i\cdot\hat{A}^{\beta}\right)\\
&\quad=e\left(g,N_2\right)
\end{aligned}
$$

3 *Correctness proof for the algorithm verify is shown below:*

$$
\begin{aligned}
&e\left(\sigma_1\sigma_2T_2^{*}\,,\,\hat{A}\right)e\left(\sigma_1\,,\,\hat{B}^{-1/H\left(T_1^{*}||T_2^{*}||\hat{T}^{*}||\sigma_1||msg\right)}\right)\\
&\quad=e\left(g^{r}\cdot g^{xr/H\left(T_1^{*}||T_2^{*}||\hat{T}^{*}||\sigma_1||msg\right)}\cdot g^{yr}\cdot g^{1/st},\hat{g}^{\alpha}\right)\\
&\quad\cdot\ e\left(g^{r},\hat{g}^{-x\alpha/H\left(T_1^{*}||T_2^{*}||\hat{T}^{*}||\sigma_1||msg\right)}\right)\\
&\quad=e\left(g,\hat{g}\right)^{\alpha\left(r+xr/H\left(T_1^{*}||T_2^{*}||\hat{T}^{*}||\sigma_1||msg\right)+yr+1/st\right)}\\
&\quad\cdot\ e\left(g,\hat{g}\right)^{(-x\alpha r)/H\left(T_1^{*}||T_2^{*}||\hat{T}^{*}||\sigma_1||msg\right)}\\
&\quad=e\left(g,\hat{g}\right)^{\alpha\left(r+yr+1/st\right)}\\
&\quad=e\left(g,\hat{g}\right)^{((1+y)\alpha tsr+\alpha)\cdot\left(1/st\right)}\\
&\quad=e\left(g^{(1+y)\alpha tsr+\alpha},\hat{g}^{1/st}\right)\\
&\quad=e\left(\left(g^{u}\cdot g^{uy}\right)^{(\alpha tsr)/u}\cdot g^{\alpha},\hat{g}^{1/st}\right)\\
&\quad=e\left(\left(t_1t_2\right)^{(\alpha tsr)/u}\cdot C,\hat{T}^{1/s}\right)\\
&\quad=e\left(T_1^{*}C\,,\,\hat{T}^{*}\right)
\end{aligned}
$$

4 *Correctness proof for the algorithm open is shown below:*

$$
\begin{aligned}
& e\left(\sigma_2\ ,\ \hat{g}\right) \cdot e\left(\sigma_1\ ,\ \hat{X}^{-1/H\left(T_1^*||T_2^*||\hat{T}^*||\sigma_1||msg\right)}\right) \\
&= e\left(g^{xr/H\left(T_1^*||T_2^*||\hat{T}^*||\sigma_1||msg\right)} \cdot g^{yr}, \hat{g}\right) \cdot e\left(g^r, \hat{g}^{-x/H\left(T_1^*||T_2^*||\hat{T}^*||\sigma_1||msg\right)}\right) \\
&= e\left(g^r, \hat{g}^y\right) \\
&= e\left(\sigma_1, t_3\right)
\end{aligned}
$$

6.2 Security proof

This section details three security definitions that capture the potential attacks in our threat model: full anonymity, full traceability, and non-frameability. In addition, a detailed analysis was conducted on information disclosure prevention of the system.

6.2.1 Full anonymity

Full anonymity: Attackers cannot compute the private key information of group managers or determine the certificate information of the signer. Given two group signatures and information about a signer, no attacker can with better than even chance determine which group signature was signed by the given signer. Full anonymity ensures that the signer is not recognised during the signing process, thereby preventing impersonation attack. The security definition of full-anonymity for the SDS-GS scheme is based on the game between challenger C and attacker A as follows.
Proof: We establish this through reduction. In this proof game, there are two roles: the attacker A and the challenger C. Suppose A can break the anonymity of our scheme; then, we can construct an algorithm A' that can break both the traceability and the (t, ε)-DLP assumptions.

Setup: At the start of the game, given a security parameter λ, challenger C runs the initialisation algorithm $Setup\left(1^\lambda\right)$ and sends the system public parameters $pp = (G_1, G_2, G_T, e, g, \hat{g}, X, \hat{X}, H)$ along with the relevant public keys (tpk, rpk, opk, gpk) to attacker A, where $gpk = (\hat{A}, \hat{B}, C)$. A engages in the game with the following queries:

- Join query: A issues a query $\mathcal{O}Add\left(i\right)$ to generate user key pairs $(usk_i = y_i, upk_i = \hat{g}^{y_i})$ and queries a random oracle $\mathcal{O}Join\left(i\right)$, while A' executes Algorithm *Join* and returns the user's group private key $gusk = \left(T_1, T_2, \hat{T}, t_4\right)$.
- Hash query: When A requests a hash value, C uniformly selects a random element from Z_p to respond. In the case of identical queries, C ensures the same response is provided.
- Open query: Challenger C maintains a list $L\left(upk_{i^*}, Sig_{i^*}\right)$, initially empty. C stores each signature generated by different users in the list L. When A queries the random oracle $\mathcal{O}Open\left(\sigma, osk, msg\right)$, it is processed as follows: if $\sigma \in L$, C

directly returns the corresponding upk_{i^*}; otherwise, A' executes $\mathcal{O}Open\,(\sigma, osk, msg)$. If $\mathcal{O}Open$ returns upk_{i^*}, the result is directly returned. If $\mathcal{O}Open$ returns $\perp$, a randomly generated upk_{i^*} is returned.

- User public key query: A queries the random oracle $\mathcal{O}\mathrm{Re}c\,(N_1, N_2)$, and $\mathcal{O}\mathrm{Re}c$ returns upk.

Challenge: A randomly selects two honest users i_0 and i_1, with public keys $upk^*_{i_0} = \hat{g}^{y_{i_0}}$ and $upk^*_{i_1} = \hat{g}^{y_{i_1}}$, and a message msg^*. A' provides A_{i_0} and A_{i_1} as challenge messages to challenger C. C executes $\mathcal{O}Ch(i_0, i_1, msg)$, returning the signature $\sigma = (T_1^*,\ T_2^*,\ \hat{T}^*,\ \sigma_1,\ \sigma_2)$, where σ is generated by attacker A_{i_b}, with $b \in \{0, 1\}$. A' exploits the above queries to attack anonymity: A' first queries the oracle $\mathcal{O}Join\,(i)$ to obtain the user's group private key, then proceeds to execute the open query to retrieve the secret-shared information stored in REG, i.e., N_1 and N_2, belonging to the actual signatory. Finally, it executes the user public key query to obtain the signatory's public key information. Throughout this process, A' needs to repeatedly execute the hash query to ensure the information passes verification.

Guess: A outputs a guess value b', and simultaneously, A' sends b' as the answer to the challenger. If $b = b'$, then A' wins. When A' wins in the game, that is, when A' successfully obtains the correct signatory's public key information, A also wins in the anonymity game. In the $Sign$ algorithm, signatures are generated by randomly choosing $s, r \in Z_p^*$ to ensure that even for the same signatory, group signatures generated each time are different. Therefore, in the open query, in most cases, $\sigma \in L$, and A' needs to query $\mathcal{O}Open$ to obtain upk_{i^*}. We define Adv^{tra} as the advantage of attacker A in traceability, and $Adv^{an} = \left| Pr\,(A\ win)\ -\ \frac{1}{2} \right|$ as the advantage of attacker A in full anonymity. Additionally, we define Adv_{DLP} as the advantage of attacker A in the DLP assumption (Advantage). If $Adv^{an} = \varepsilon$, then $\varepsilon \leq Adv_{DLP} + Adv^{tra} + \frac{1}{2}$. □

Definition 1: The SDS-GS scheme achieves full anonymity if SDS-GS scheme satisfies traceability and the (t, ε)-DLP assumption holds.

6.2.2 Full traceability

Full traceability: Full traceability is stronger than traceability and can be considered a powerful form of collusion resistance. Specifically, even if a signer creates a signature through collusion or other means, group managers can track the actual signer in a highly reliable manner. Full traceability can resist key exposure attacks and group member tracking attacks. The security definition of full traceability for the SDS-GS scheme based on the game between challenger C and attacker A as follows.

Proof: In this proof game, there are two roles: attacker A and challenger C. Assuming A is an attacker on the traceability of our group signatures with a success probability of ε, then A can attack the EUF-CMA security of FHS signatures with the same success probability. C is the challenger for the EUF-CMA security of FHS signatures. Technically, A, by returning a valid signature σ on the message msg, either disrupts the opening process or provides a signature that can be opened but cannot be reconstructed

into the correct public key information. The scheme ensures the correct transmission of user public key information in Algorithm *Join* through trusted TMA and RA, thus ruling out the latter case.

Setup: At the start of the game, given a security parameter λ, challenger C runs the initialisation algorithm $Setup(1^{\lambda})$ and sends the system public parameters $pp = (G_1, G_2, G_T, e, g, \hat{g}, X, \hat{X}, h)$ along with the relevant public keys tpk, rpk, and gpk to attacker A, where $gpk = (\hat{A}, \hat{B}, C)$. A engages in the game with the following queries:

- Join query: A randomly selects $y_i \in Z_p$, generates user key pairs $(usk_i = y_i, upk_i = \hat{g}^{y_i})$, and queries the oracle $\mathcal{O}Join(i)$ to return the user's group private key $gusk = (T_1, T_2, \hat{T}, t_4)$.
- Signature query: A queries the random oracle $\mathcal{O}Sign(gusk_i, msg)$ to obtain the signature σ.
- Hash query: When A requests a hash value, C uniformly selects a random element from Z_p to respond. In the case of identical queries, C ensures the same response is provided.

Forgery: If the signature σ is valid, $Verify(gpk, msg, \sigma) \rightarrow valid$, where $gpk = (\hat{A}, \hat{B}, C)$. Then A outputs the signature σ obtained through the queries described above.

Output: For an untraceable valid signature $\sigma = (T_1^*,\ T_2^*,\ \hat{T}^*,\ \sigma_1,\ \sigma_2)$ satisfying the following conditions:

$$e\left(\sigma_1\sigma_2 T_2^*\ ,\ \hat{A}\right) \cdot e\left(\sigma_1\ ,\ \hat{B}^{-1/H\left(T_1^*||T_2^*||\hat{T}^*||\sigma_1||msg\right)}\right) = e\left(T_1^* C\ ,\ \hat{T}^*\right) \tag{1}$$

$$e\left(\sigma_2\ ,\ \hat{g}\right) \quad \cdot e\left(\sigma_1\ ,\ \hat{X}^{-1/H\left(T_1^*||T_2^*||\hat{T}^*||\sigma_1||msg\right)}\right) \neq e\left(\sigma_1, t_3\right) \tag{2}$$

The equation (1) imply:

$$\begin{aligned} & e\left(\sigma_1\sigma_2\ ,\ \hat{A}\right) \cdot e\left(\sigma_1\ ,\ \hat{B}^{-1/H\left(T_1^*||T_2^*||\hat{T}^*||\sigma_1||msg\right)}\right) \\ & = e\left(\sigma_1\sigma_2\ ,\ \hat{A}\right) \cdot e\left(\sigma_1^{-x/H\left(T_1^*||T_2^*||\hat{T}^*||\sigma_1||msg\right)}\ ,\ \hat{A}\right) \\ & = e\left(\sigma_1\ ,\ \hat{A}\right) \cdot e\left(\sigma_2\sigma_1^{-x/H\left(T_1^*||T_2^*||\hat{T}^*||\sigma_1||msg\right)}\ ,\ \hat{A}\right) \\ & = e\left(T_1^*\ ,\ \hat{T}^*\right) \end{aligned} \tag{3}$$

$$e\left(T_2^*, \hat{g}\right) = e\left(g, \hat{T}^*\right) \tag{4}$$

Equations (3) and (4) demonstrate that $(T_1^*,\ T_2^*,\ \hat{T}^*)$ is a valid FHS signature on the message $(\sigma_1,\ \sigma_2\sigma_1^{-x/H(T_1^*||T_2^*||\hat{T}^*||\sigma_1||msg)})$. In algorithm *Join*, users U_i construct their

$gusk_i$ for generating messages, while the group manager receives and stores $t_{3_i} = \hat{g}^{y_i}$. Clearly, for valid FHS signatures:

$$\begin{aligned} e\left(\sigma_1, \hat{g}^{y_i}\right) &= e\left(\sigma_2 \sigma_1^{-x / H\left(T_1^* || T_2^* || \hat{T}^* || \sigma_1 || msg\right)}, \hat{g}\right) \\ &= e\left(\sigma_2\ ,\ \hat{g}\right) e\left(\sigma_1^{-x / H\left(T_1^* || T_2^* || \hat{T}^* || \sigma_1 || msg\right)}\ ,\ \hat{g}\right) \\ &= e\left(\sigma_2\ ,\ \hat{g}\right) \cdot e\left(\sigma_1\ ,\ \hat{X}^{-1 / H\left(T_1^* || T_2^* || \hat{T}^* || \sigma_1 || msg\right)}\right) \end{aligned}$$

This contradicts equation (2). Therefore, if the scheme satisfies the EUF-CMA security of FHS signatures, it also achieves full traceability. □

Definition 2: If the SDS-GS scheme satisfies the EUF-CMA security notion of FHS signatures, then the scheme achieves full traceability.

6.2.3 Non-frameability

Non-frameability: Only legitimate group members, who possess valid membership certificates and signing keys, can generate valid group signatures. Non-frameability can resist key complex impersonation attack. It prevents any malicious entity or other group members from creating a valid signature that could be falsely attributed to an innocent member, thereby protecting the integrity and trust within the group. The non-frameability security definition of the SDS-GS scheme is based on the following reduction.

Proof: We prove this through reduction. Suppose an attacker A successfully attacks the non-frameability of our scheme. Then, A can forge a valid group signature $\sigma' = (T_1^{*\prime},\ T_2^{*\prime},\ \hat{T}^{*\prime},\ \sigma_1',\ \sigma_2')$ on a message msg' that can be traced back to an honest user U_i. Let us assume that the public key of U_i is upk_i, and REG contains $t_{3_i} = \hat{g}^{y_i}$ for user U_i. U_i can generate a valid group signature $\sigma = (T_1^*,\ T_2^*,\ \hat{T}^*,\ \sigma_1,\ \sigma_2)$ on the message msg'. We distinguish between the following three cases:

- Case 1: A forged group signature satisfies $\sigma' = \sigma$.
- Case 2: A forged group signature satisfies $\sigma' \neq \sigma$. However, A queries $\mathcal{O}Open$ and can map to U_i, which is stored in REG as $t_{3_i} = \hat{g}^{y_i}$.
- Case 3: A forged group signature satisfies $\sigma' \neq \sigma$, and A queries $\mathcal{O}Open$ but cannot map to U_i in REG as $t_{3_i} = \hat{g}^{y_i}$. Nevertheless, the user's public key is reconstructed after secret sharing, denoted as $upk_i' = upk_i$.

Case 1: If A queries $\mathcal{O}Join$ and obtains $gusk' \neq gusk$, then A queries $\mathcal{O}Sign$, computes: $T_1^{*\prime} = (T_1')^{s' \cdot r'}$, $T_2^{*\prime} = (T_2')^{1/s'}$, $\hat{T}^{*\prime} = (\hat{T}')^{1/s'}$, $\sigma_1' = g^{r'}$, $\sigma_2' = X^{r' / H\left(T_1^{*\prime} || T_2^{*\prime} || \hat{T}^{*\prime} || \sigma_1' || msg'\right)} \cdot t_4^{r'}$.

Assuming $\sigma' = \sigma$, we have

$$\left(T_1^{*\prime},\ T_2^{*\prime},\ \hat{T}^{*\prime},\ \sigma_1',\ \sigma_2'\right) = \left(T_1^*,\ T_2^*,\ \hat{T}^*,\ \sigma_1,\ \sigma_2\right)$$

This directly implies that if A's signature satisfies $\sigma' = \sigma$, then A must achieve a collision in the hash function H and break the DLP problem.

Case 2: Following the proof of Lemma 2, valid group signatures are also valid FHS signatures. If the group signature σ' satisfies:

$$e\left(\sigma_1'\sigma_2'T_2^*\ ,\ \hat{A}\right)\cdot e\left(\sigma_1'\ ,\ \hat{B}^{-1/H\left(T_1^{*\prime}||T_2^{*\prime}||\hat{T}^{*\prime}||\sigma_1'||msg'\right)}\right)=e\left(T_1^{*\prime}C\ ,\ \hat{T}^{*\prime}\right)$$

then it satisfies:

$$e\left(\sigma_2'\ ,\ \hat{g}\right)\cdot e\left(\sigma_1'\ ,\ \hat{X}^{-1/H\left(T_1^{*\prime}||T_2^{*\prime}||\hat{T}^{*\prime}||\sigma_1'||msg'\right)}\right)=e\left(\sigma_1',t_3'\right)$$

where $t_3' = \hat{g}^{y'}$ and y' is obtained by A querying $\mathcal{O}Add$. If the group signature σ' can be mapped to U_i in REG as $t_{3_i} = \hat{g}^{y_i}$, then it satisfies:

$$e\left(\sigma_2'\ ,\ \hat{g}\right)\cdot e\left(\sigma_1'\ ,\ \hat{X}^{-1/H\left(T_1^{*\prime}||T_2^{*\prime}||\hat{T}^{*\prime}||\sigma_1'||msg'\right)}\right)=e\left(\sigma_1',t_{3_i}\right)$$

and it is evident that $e(\sigma_1', t_3') = e(\sigma_1', t_{3_i})$. This demonstrates that Case 2 holds, with the prerequisite that A can solve the DLP problem.

Case 3: In Algorithm *Join*, the scheme is subjected to secure verification by the TMA and RA entities to ensure that user public key information remains anonymous and unmodifiable during the transmission process. The protocol also incorporates a secret sharing scheme that eliminates the possibility of collusion attacks by fewer than t users. Therefore, if an attacker A were to achieve the condition where $gusk' = gusk$ after querying $\mathcal{O}Join$, then A must also be able to satisfy their public key information $upk' = upk_i$ through the $\mathcal{O}Add$ query before joining the protocol. However, it is evident that A would need to break the DLP problem to achieve $upk' = upk_i$.

In summary, if the (t, ε)-DLP assumption holds, then the scheme satisfies non-frameability. □

Definition 3: If the (t, ε)-DLP assumption holds, then SDS-GS scheme satisfies non-frameability.

6.2.4 *Information disclosure prevention*

The SDS-GS scheme effectively implements information disclosure prevention mechanisms. The prevention mechanism is based on information isolation between various entities, as shown below:

1 *The analysis of GM:* GM has two responsibilities in the system: as the administrator of an organisation's group, he is responsible for managing the members of the group; on the other hand, as an entity storing secret shares, he is responsible for storing secret shares from other groups that are about user identity information.

In Algorithm *Join*, the new user U_i firstly sends the public key upk_i to RA, then RA computes and publish $rupk_i = rupk_{i-1}\,(upk_i)^{rsk}$. Secondly, U_i negotiates with the TMA to obfuscate the upk_i, obtaining a series of obfuscated data. TMA sends these data to GM. GM verifies them with $rupk_i$, which ensures that the upk_i passed from TMA to GM is the actual identity information. In this process, GM cannot know the actual public key information of the user directly. GM can only get upk by executing the open algorithm

When GM as an entity storing secret shares, it will store the secret share by a key-value table $MapList$, and the key is a value of a conflict-resistant hash function, which computed by TMA. In this process, it is impossible for GM to know which new user's identity information corresponds to the secret sharing in their table, that eliminates the possibility of GM to carry out a conspiracy attack to leak the user's privacy.

2 *The analysis of RA:* RA is responsible for vetting new users, so it is necessary for it to know the user's public key upk. However, when a user joins the system, he will get an group private key $gusk$. Then user runs Algorithm *Sign* to generate a group signature σ on a message. The correspondence between σ and upk can only be known by the GM performing Algorithm *Open*. Therefore RA cannot leak the user's privacy.

3 *The analysis of TMA:* TMA works in Algorithm *Join.ValidataUserAnd GenerateSNQ* and Algorithm *Join.SecretDistribute*. It is responsible for transferring the parameters. Its main work is as follows:

One is to obfuscate the user's public key upk. During the obfuscation process, the TMA does not know the value of upk. TMA cannot disclose the user's privacy; □

The other is to distribute the secret shares to different GM randomly. Although TMA holds secret collection and restoration credentials, the user's activation ID must be obtained by the corresponding group administrator executing Algorithm *Open* to open the group signature. TMA is unable to determine which user the credential parameters they hold correspond to. TMA is unable to disclose information during this process.

7 Performance evaluation

This section presents simulation experiments on proposed scheme SDS-GS and compares its performance and functionality with other schemes.

Parameter setting: The experiments were conducted on a desktop computer equipped with an i7-12700CPU@2.10GHz processor, 16GB of memory, running the Ubuntu 22.04 operating system. The implementation was done using Charm-Crypto 0.50, which relies on GMP 5.x, PBC and OPENSSL in Python 3.7.9. The bilinear groups are generated by using BN-254 curves, which belong to Barreto-Naehrig curves (Barreto and Naehrig, 2006), and that the representation length of the elements in Z_p and G_1 is 254 bits, while the representation length of the elements in G_2 is 508 bits.

Table 2 provides an overview of the symbols used in the scheme along with their respective execution times. Table 3 summarises the time consumption and signature lengths for three different group signature schemes, SGSDT (Blömer et al., 2016);

GSDT (Lu et al., 2020); STDGS (Camenisch et al., 2020), in three crucial steps: signature generation, signature verification, and signature tracing. Considering the worst-case scenario, set the threshold for secret sharing members to be equal to the number of members in the entire group.

Table 2 Operation time consumption

Symbol	*Description*	*Time (ms)*
T_E	Exponentiation	1.18716
T_M	Multiplication	0.00435
T_I	Inversion operation	0.01044
T_H	Hash function operation	0.00202
T_P	Bilinear pairing operation	31.6292

Table 3 Comparison of computational cost among different algorithms

Scheme	*Signature verification*	*Signature generation*
GSDT	$17T_E + 9T_M + T_I + T_H$	$13T_E + 9T_M + 2T_P + T_H$
SGSDT	$13T_E + 6T_M + 4T_P + T_H$	$13T_E + 9T_M + T_H$
STDGS	$4T_E + 3T_M + 4T_P + T_H$	$4T_E + 3T_M + 4T_P + T_H$
SDS-GS	$6T_E + T_M + T_H$	$T_E + 4T_M + 3T_P + T_I + T_H$
Signature length/bits	*Tracing (traversal maximum number of times)*	
$9G_1^* + 8Z_p^* = 4{,}319$	$(13 + 5t)T_E + (9 + 4t)T_M + 2T_P + (1 + t)T_H$	
$3G_1^* + 7Z_p^* = 2{,}540$	$(13 + t)T_E + (2t + 9)T_M + 4tT_P + T_H$	
$2G_1^* + 3Z_p^* = 1{,}270$	$(mt + m)T_E + (mt + m)T_M + (m + 1)T_P$	
$4G_1^* + 1G_2^* = 1{,}524$	$(2t+2)T_E + (2t+2)T_M + (2t+m+2)T_P + (t+1)T_H + 2T_I$	

Note: G_1^*, G_2^*, Z_p^*: The length of non-zero points in G_1,G_2 and Z_p.
m, t: The number of members in each group and the threshold value for participants in secret sharing.

It is evident that the signature length of STDGS and SDS-GS scheme is shorter than GSDT and SGSDT scheme. This is attributed to the GSDT and SGSDT schemes, which employ complex zero-knowledge proof techniques and incorporate them into the signature. Although the signature of STDGS scheme is the shortest, the signature structure of STDGS scheme is very primitive, and it does not fully consider the efficiency of the user's traceability process. The signature of STDGS scheme does not set the initial screening mark. In the open algorithm, it requires collecting almost all the secret shares in the system, which is the vast majority of the meaningless work, and this efficiency can not satisfy the requirements of large systems. On the other hand, the signature of SDS-GS scheme has a preliminary screening mark, and only certain secret shares need to be collected.

In the signature generation phase, the SDS-GS scheme is notably faster than the other schemes because it combines PS signatures and FHS signatures to achieve security without the use of zero-knowledge proofs. In the signature verification phase, the SDS-GS scheme is almost on par with the GSDT scheme and outperforms the others. However, the GSDT scheme's signatures with larger signature lengths. Overall, the SDS-GS scheme holds an advantage.

Table 4 Functional comparison of different schemes

Scheme	*APP*	*MUT*	*UDJ*	*EVL*	*ICI*	*TUC*	*GSL*
GSDT	Yes	Yes	Yes	Yes	No	No	Long
SGSDT	Yes	Yes	Yes	No	No	No	Long
STDGS	Yes	Yes	Yes	No	No	No	Short
SDS-GS	Yes	Yes	Yes	Yes	Yes	Yes	Short

Note: APP – the abilities of privacy preservation; MUT – malicious user tracing; UDJ – user dynamic join; EVL – efficient verify in large-scale communication scenarios; ICI – information communication isolation between entities; TUC – traceability under control; GSL – group signature length.

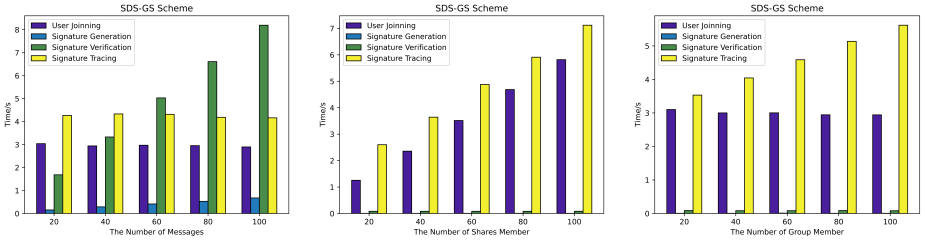
Table 5 Experimental condition

Condition	s	t	m
A	10–100	50	50
B	1	10–100	50
C	1	50	10–100

Note: m – the number of group members; s – message quantity; t – the number of participants in secret sharing.

Additionally, SDS-GS scheme has certain advantages over other existing group signature schemes in terms of security, traceability, and other aspects. Comparisons were made, as shown in Table 4. The efficiency of the SGSDT and STDGS schemes is severely limited by the number of group members, and cannot meet the requirements of large-scale communication scenarios. The GSDT scheme adjusts the signature structure to speed up the efficiency of signature verification, but this leads to excessively long signature lengths. It is also vulnerable to security threats in the constructed signature phase. In addition, none of these schemes considered the need for cross-agency communication. They did not design information communication isolation between entities, nor did they consider restricting the trace rights of group administrators.

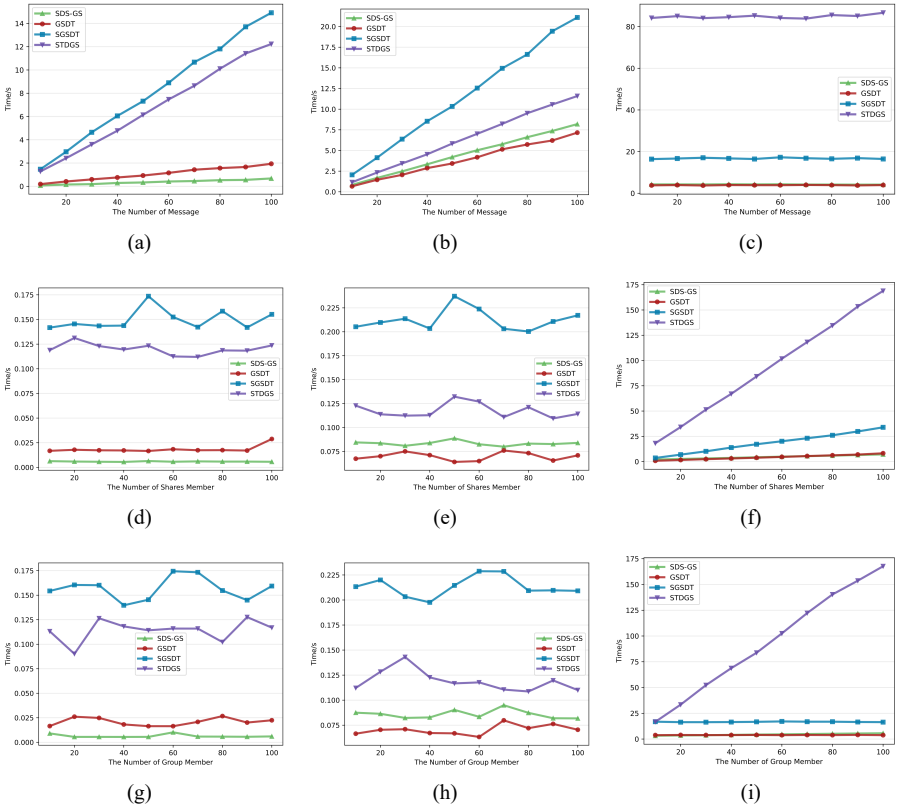
Figure 3 SDS-GS scheme's performance under three conditions, (a) SDS-GS scheme under condition A (b) SDS-GS scheme under condition B (c) SDS-GS scheme under condition C (see online version for colours)



(a) (b) (c)

This research conducted simulation experiments to analyse the time consumption of the SDS-GS scheme under three different conditions, shown in Table 5. Figure 3 illustrates the results of these experiments.

Figure 4 Comparison of SDS-GS scheme with other schemes under three conditions, (a) sign algorithm under condition A (b) verify algorithm under condition A (c) open algorithm under condition A (d) sign algorithm under condition B (e) verify algorithm under condition B (f) open algorithm under condition B (g) sign algorithm under condition C (h) verify algorithm under condition C (i) open algorithm under condition C (see online version for colours)



This research also compared the SDS-GS scheme with three other schemes in the same simulation environment, as shown in Figure 4. Clearly, under all three conditions, the SDS-GS scheme outperforms in both signature generation and tracing algorithms. In signature verification, it performs closely to the GSDT scheme and is among the top performers. The SDS-GS scheme achieves security in signature generation using a combination of FHS and PS signature approaches instead of relying heavily on complex zero-knowledge proofs.

In condition A, as the number of signature messages increases, the efficiency advantage of the SDS-GS scheme in signature generation and verification becomes increasingly apparent. This is crucial in systems involving a large volume of message transmission. On the other hand, the signature algorithm of the SDS-GS scheme avoids frequent secret collection and only initiates secret collection once after locating the malicious signers' entries. Therefore, the SDS-GS scheme has a significant advantage in the open algorithm.

Evidently, in conditions B and C, as the number of group members or participants in the secret sharing scheme increases, the increase in time consumption for the SDS-GS scheme's tracing algorithm is relatively small. This is advantageous for building large-scale data sharing systems. Therefore, our healthcare data sharing system's performance in tracing malicious users is not significantly affected by an increase in the number of hospitals, patients, or participants in the system.

Table 6 Comprehensive comparison of different schemes

Scheme	*CAC (s)*	*CTV (KB)*	*HIR*	*EIE*	*SC*
GSDT	96.298	527.22	High	No	Low
SGSDT	411.09	310.05	High	No	Low
STDGS	236.26	155.02	Medium	No	Low
SDS-GS	86.761	186.03	Low	Yes	High

Note: CAC – the computational cost of a thousand messages under condition A, including the complete signature generation process and signature verification process; CTV – communication transmission volume of 1,000 signatures under condition A; HIR – hardware and infrastructure requirements; EIE – ease of integration into existing systems; SC – scalability.

Overall, the SDS-GS scheme has significant advantages in practice. The comprehensive comparison between SDS-GS scheme and other schemes is shown in Table 6. The SDS-GS scheme exhibits lower communication costs and is suitable for use in bandwidth limited or delay sensitive environments. The lower computational cost also allows the solution to be used on devices with low computing power, such as smart wearable devices. By comparison, the GSDT scheme and SGSDT scheme require higher requirements for device network environment and computing power, while the trace process efficiency of the STDGS scheme cannot meet the needs of the medical environment. Although all four schemes use secret sharing to improve security, the remaining three schemes only consider the needs of new users, while the SDS-GS scheme fully considers the needs of new institutions. New institutions can join the system at any time to meet the needs of business growth. Overall, SDS-GS has higher scalability and is easy to integrate into existing systems.

8 Conclusions

In this research, we have designed a short group signature algorithm based on FHS and PS signatures, which enables controlled traceability. The short group signature, with its short length and low computational overhead, helps to alleviate the computational, transmission, and storage burdens associated with healthcare data. Security analysis has demonstrated that the SDS-GS scheme offers robust security and anonymity. Compared to other schemes, SDS-GS boasts additional functionalities. Performance evaluations have shown that SDS-GS is highly efficient for large healthcare data-sharing scenarios. In addition, the enhanced security features of SDS-GS can be applied to various domains where data privacy and security are paramount, such as financial transactions, government communications, and smart city applications. By reducing computational and transmission costs, SDS-GS can facilitate the deployment of secure data-sharing solutions in environments with limited resources.

Future research will focus on the implementation and evaluation of the SDS-GS scheme within healthcare consortium blockchain environments. This includes developing practical deployment strategies, addressing scalability issues, and ensuring seamless integration with existing healthcare systems. Additionally, we will explore the scheme's applicability to other sectors, investigating its performance and security in diverse real-world scenarios. SDS-GS scheme may not meet the needs of other environments in terms of user engagement efficiency. Further studies will aim to optimise the scheme for even greater efficiency and to enhance its usability for non-technical stakeholders.

In conclusion, the SDS-GS scheme presents a significant advancement in secure group signatures, offering a practical solution to the challenges of healthcare data sharing. Its robustness, efficiency, and additional functionalities position it as a promising candidate for widespread adoption.

References

Aki, S.G. (1983) 'Digital signatures: a tutorial survey', *Computer*, Vol. 16, No. 2, pp.15–24.

Apell, P. and Eriksson, H. (2023) 'Artificial intelligence (AI) healthcare technology innovations: the current state and challenges from a life science industry perspective', *Technology Analysis & Strategic Management*, Vol. 35, No. 2, pp.179–193.

Azaria, A., Ekblaw, A., Vieira, T. and Lippman, A. (2016) 'MedRec: using blockchain for medical data access and permission management', *2016 2nd International Conference on Open and Big Data (OBD)*, IEEE, Vienna, Austria, pp.25–30.

Barreto, P.S.L.M. and Naehrig, M. (2006) 'Pairing-friendly elliptic curves of prime order', in Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Rangan, C.P., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M.Y., Weikum, G., Preneel, B. and Tavares, S. (Eds.): *Selected Areas in Cryptography*, Vol. 3897, pp.319–331, Springer Berlin Heidelberg, Berlin, Heidelberg.

Bellare, M., Micciancio, D. and Warinschi, B. (2003) 'Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions', in Biham, E. (Ed.): *Advances in Cryptology – EUROCRYPT 2003, Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp.614–629.

Bellare, M., Shi, H. and Zhang, C. (2005) 'Foundations of group signatures: the case of dynamic groups', *Topics in Cryptology – CT-RSA 2005*, Springer, Berlin, Heidelberg, pp.136–153.

Bhattacharya, S., Maddikunta, P.K.R., Pham, Q-V., Gadekallu, T.R., Chowdhary, C.L., Alazab, M. and Piran, M.J. (2021) 'Deep learning and medical image processing for coronavirus (COVID-19) pandemic: a survey', *Sustainable Cities and Society*, Vol. 65, p.102589.

Blömer, J., Juhnke, J. and Löken, N. (2016) 'Short group signatures with distributed traceability', in Kotsireas, I.S., Rump, S.M. and Yap, C.K. (Eds.): *Mathematical Aspects of Computer and Information Sciences, Lecture Notes in Computer Science*, pp.166–180, Springer International Publishing, Cham.

Camenisch, J., Drijvers, M., Lehmann, A., Neven, G. and Towa, P. (2020) 'Short threshold dynamic group signatures', in Galdi, C. and Kolesnikov, V. (Eds.): *Security and Cryptography for Networks – 12th International Conference, SCN 2020, Proceedings*, 14–16 September, Springer, Amalfi, Italy, Vol. 12238 of *Lecture Notes in Computer Science*, pp.401–423.

CANLII (2000) *Parliament of Canada Personal Information Protection and Electronic Documents Act*, Canadian Legal Information Institute [online] https://www.canlii.org/en/ca/laws/stat/sc-2000-c-5/latest/sc-2000-c-5.html (accessed 2 July 2024).

Carvalho, L.F., Fernandes, G., de Assis, M.V.O., Rodrigues, J.J.P.C. and Proenca, M.K, (2014) 'Digital signature of network segment for healthcare environments support', *IRBM*, Vol. 35, No. 6, pp.299–309.

Chaum, D. and van Heyst, E. (1991) 'Group signatures', in Davies, D.W. (Ed.): *Advances in Cryptology — EUROCRYPT '91, Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp.257–265.

Chen, H-Y., Wu, Z-Y., Chen, T-L., Huang, Y-M. and Liu, C-H. (2021a) 'Security privacy and policy for cryptographic based electronic medical information system', *Sensors (Basel, Switzerland)*, Vol. 21, No. 3, p.713.

Clarisse, R. and Sanders, O. (2020) 'Group signature without random oracles from randomizable signatures', in Nguyen, K., Wu, W., Lam, K.Y. and Wang, H. (Eds.): *Provable and Practical Security*, Vol. 12505, pp.3–23, Springer International Publishing, Cham.

Fan, K., Wang, S., Ren, Y., Li, H. and Yang, Y. (2018) 'MedBlock: efficient and secure medical data sharing via blockchain', *Journal of Medical Systems*, Vol. 42, No. 8, p.136.

Fuchsbauer, G., Hanser, C. and Slamanig, D. (2019) 'Structure-preserving signatures on equivalence classes and constant-size anonymous credentials', *Journal of Cryptology*, Vol. 32, No. 2, pp.498–546.

Goldwasser, S., Micali, S. and Rivest, R.L. (1988) 'A digital signature scheme secure against adaptive chosen-message attacks', *SIAM Journal on Computing*, Vol. 17, No. 2, pp.281–308.

Greene, S.M., Ahmed, M., Chua, P.S. and Grossmann, C. (2022) *Sharing Health Data: The Why, the Will, and the Way Forward*, National Academies Press, Washington, D.C.

HHS (1996) *U.S. Congress Health Insurance Portability and Accountability Act-Privacy Rule*, US Department of Health and Human Services [online] https://www.hhs.gov/hipaa/for-professionals/index.html (accessed 2 July 2024).

Hulsen, T. (2020) 'Sharing is caring – data sharing initiatives in healthcare', *International Journal of Environmental Research and Public Health*, Vol. 17, No. 9, p.3046.

Hulsen, T., Jamuar, S.S., Moody, A.R., Karnes, J.H., Varga, O., Hedensted, S., Spreafico, R., Hafler, D.A. and McKinney, E.F. (2019) 'From big data to precision medicine', *Frontiers in Medicine*, Vol. 6, p.34.

Jiang, T., Chen, X. and Ma, J. (2015) 'Public integrity auditing for shared dynamic cloud data with group user revocation', *IEEE Transactions on Computers*, Vol. 65, No. 8, pp.2363–2373.

Kashani, M.H., Madanipour, M., Nikravan, M., Asghari, P. and Mahdipour, E. (2021) 'A systematic review of IoT in healthcare: applications, techniques, and trends', *Journal of Network and Computer Applications*, Vol. 192, p.103164.

Kiayias, A. and Yung, M. (2006) 'Secure scalable group signature with dynamic joins and separable authorities', *International Journal of Security and Networks*, Vol. 1, Nos. 1–2, pp.24–45.

Kocabaș, O. and Soyata, T. (2016) 'Medical data analytics in the cloud using homomorphic encryption', *Handbook of Research on Cloud Infrastructure for Big Data Analytics*, pp.751–768, IGI Global.

Li, C., Jiang, B., Guo, Y. and Xin, X. (2023) 'Efficient group blind signature for medical data anonymous authentication in blockchain-enabled IoMT', *CMC – Computers Materials & Continua*, Vol. 76, No. 1, pp.591–606.

Li, D., Liao, X., Xiang, T., Wu, J. and Le, J. (2020) 'Privacy-preserving self-serviced medical diagnosis scheme based on secure multi-party computation', *Computers & Security*, Vol. 90, p.101701.

Li, X-x., Qian, H-f. and Li, J-h. (2009) 'Democratic group signatures with threshold traceability', *Journal of Shanghai Jiaotong University (Science)*, Vol. 14, No. 1, pp.98–101.

Liu, T., Siegel, E. and Shen, D. (2022) 'Deep learning and medical image analysis for COVID-19 diagnosis and prediction', *Annual Review of Biomedical Engineering*, Vol. 24, No. 1, pp.179–201.

Lu, T., Li, J., Zhang, L. and Lam, K-Y. (2020) 'Group signatures with decentralized tracing', in Liu, Z. and Yung, M. (Eds.): *Information Security and Cryptology, Lecture Notes in Computer Science*, pp.435–442, Springer International Publishing, Cham.

Lysyanskaya, A., Rivest, R.L., Sahai, A. and Wolf, S. (2000) 'Pseudonym systems', in Goos, G., Hartmanis, J., van Leeuwen, J., Heys, H. and Adams, C. (Eds.): *Selected Areas in Cryptography*, Vol. 1758, pp.184–199, Springer Berlin Heidelberg, Berlin, Heidelberg.

NHC (2018) *National Health and Medical Big Data Standards, Security and Service Management Measures*, National Health Commission of the People's Republic of China [online] https://www.cac.gov.cn/2018-09/15/c_1123432498.htm (accessed 2 July 2024).

Olakanmi, O.O. and Odeyemi, K.O. (2023) 'Expressible access control scheme for data sharing and collaboration in cloud-centric internet of medical things system', *Journal of Ambient Intelligence and Humanized Computing*, Vol. 14, No. 6, pp.7189–7205.

Pointcheval, D. and Sanders, O. (2018) 'Reassessing security of randomizable signatures', in Smart, N.P. (Ed.): *Topics in Cryptology – CT-RSA 2018*, Springer International Publishing, Cham, Vol. 10808, pp.319–338.

Ramesh, D., Sharma, R.P. and Edla, D.R. (2020) 'HHDSSC: harnessing healthcare data security in cloud using ciphertext policy attribute-based encryption', *International Journal of Information and Computer Security*, Vol. 13, Nos. 3–4, pp.322–336.

Su, Y., Sun, J., Qin, J. and Hu, J. (2022) 'Publicly verifiable shared dynamic electronic health record databases with functional commitment supporting privacy-preserving integrity auditing', *IEEE Transactions on Cloud Computing*, Vol. 10, No. 3, pp.2050–2065.

Umamageswari, A. and Suresh, G.R. (2014) 'Secure medical image communication using ROI based lossless watermarking and novel digital signature', *Journal of Engineering Research*, Vol. 2, No. 3, pp.87–108.

Wang, Z. (2022) 'Blockchain-based edge computing data storage protocol under simplified group signature', *IEEE Transactions on Emerging Topics in Computing*, Vol. 10, No. 2, pp.1009–1019.

Yu, Y-C. and Hou, T-W. (2014) 'An efficient forward-secure group certificate digital signature scheme to enhance EMR authentication process', *Medical & Biological Engineering & Computing*, Vol. 52, No. 5, pp.449–457.

Yu, Y-C., Huang, T-Y. and Hou, T-W. (2012) 'Forward secure digital signature for electronic medical records', *Journal of Medical Systems*, Vol. 36, No. 2, pp.399–406.

Zhang, X., Yadollahi, M.M., Dadkhah, S., Isah, H., Le, D-P. and Ghorbani, A.A. (2022) 'Data breach: analysis, countermeasures and challenges', *International Journal of Information and Computer Security*, Vol. 19, Nos. 3–4, pp.402–442.

Zhang, Y., Zheng, D. and Deng, R.H. (2018) 'Security and privacy in smart health: efficient policy-hiding attribute-based access control', *IEEE Internet of Things Journal*, Vol. 5, No. 3, pp.2130–2145.