A Certificateless Designated Verifier Sanitizable Signature

Yonghua Zhan[1], Bixia Yi[1], Yang Yang[1,2](✉), Renjie He[1], and Rui Shi[3]

[1] School of Computer and Big Data, Fuzhou University, Fuzhou, Fujian, China
[2] School of Computing and Information Systems, Singapore Management University, Singapore, Singapore
yang.yang.research@gmail.com
[3] Beijing Electronic Science and Technology Institute, Beijing, China

Abstract. Sanitizable Signature is a digital signature variant that enables modification operations, allowing sanitizers to alter the signed data in a regulated manner without requiring any interaction with the original signer. It is widely used in scenarios such as healthcare data privacy protection, social networks, secure routing, etc. In existing sanitizable signature schemes, anyone can verify the validity and authenticity of the sanitized message, which results in costly certificate management overhead or complicated key escrow problems. To address these challenges, a designated verifier certificateless sanitizable signature scheme is proposed. This scheme introduces the concept of a designated verifier into sanitizable signatures, allowing sanitizers to specify verifier for the sanitized message. Only the specified verifiers can verify the validity of the message/signature pair, ensuring that the message information remains confidential to parties other than the designated verifiers. Security analysis demonstrates that the proposed scheme satisfies the properties of unforgeability, immutability, privacy and non-transferability. Comparative analysis demonstrates that the proposed scheme provides additional support for designated verifiers compared to traditional sanitizable signature schemes, while eliminating certificate management and key escrow issues. We have conducted an experiment, and the results demonstrate that the proposed scheme displays excellent efficiency.

Keywords: Data Sharing · Sanitizable Signature · Certificateless · Designated Verifier

1 Introduction

Given the swift advancement and integration of technologies such as cloud computing, big data, and the Internet of Things (IoT), data is being shared among various users and organizations, and its value is becoming increasingly significant. However, as data empowers and enables intelligence, it also faces severe privacy protection challenges. To strengthen the management and protection of data, many countries have enacted relevant laws and regulations. Examples include China's "Data Security Law of the People's Republic of China" and "Personal Information Protection Law of the People's Republic of China," the European Union's General Data Protection Regulation (GDPR), and the United States' Health Insurance Portability and Accountability Act (HIPAA) for medical data security [3]. These regulations aim to establish sound data security governance systems and enhance data security capabilities.

Digital signatures play a crucial role in cryptographic techniques, guaranteeing the authenticity and integrity of data. Any slight modification to signed data will result in the failure of signature verification. While digital signatures provide a solid security foundation for identity authentication and data integrity, they also limit the ability to make reasonable modifications to signed data in certain application scenarios, hindering users from flexibly and efficiently utilizing documents. For example, when the government discloses documents, sensitive data involving personal information and national secrets often need to be redacted. If traditional digital signature schemes are used to sign the documents, citizens are unable to verify the authenticity of the disclosed information due to the partial modification of the document's data. To ensure the validity of the modified document, it would require the signer to re-sign it. If there are numerous versions of modified documents, this method would incur significant computational overhead. Similarly, in scenarios such as electronic health records [8,14], multimedia forensics, e-commerce, and smart grids, using only traditional digital signature schemes cannot efficiently verify the validity of shared sub-documents. To address this issue, sanitizable signatures provide an effective solution [1].

Sanitizable Signature permits a semi-trusted intermediary, referred to as the sanitizer, to meticulously and non-interactively modify the signed data, all the while retaining the capability to generate a new signature that can be validated effectively. It ensures the integrity and authenticity of shared data and enables the flexible hiding of specific sensitive information based on different data sharing scenarios and recipients, thus ensuring data security while harnessing the value of the data and promoting the development of data-driven applications.

Sanitizable signatures enable the hiding of sensitive data by the sanitizer for privacy protection. However, in practice, the unmodified parts of the sanitized document may still contain private information. Unfortunately, most sanitizable signature schemes do not consider the risk of information leakage from these unmodified parts during the data sharing process. Taking medical data as an example, after a doctor signs an electronic medical record, the patient can act as the sanitizer to hide sensitive information before submitting the sanitized record to an insurance company for claims processing. At this point, the insurance company may sell the patient's medical information to interested parties, and any other party that obtains the sanitized document can verify its validity, posing a severe privacy threat to the patient. Therefore, it is necessary to adopt a Designated Verifier mechanism to limit the scope of data verification and ensure that only specific recipients can authenticate and validate the integrity of the shared data [2,11].

Sanitizable signatures are typically based on traditional public key cryptography, requiring a trusted certificate authority to issue certificates for authenticating user public keys. As the number of users increases, this leads to expensive certificate management overhead. To tackle this issue, several identity-based sanitizable signature schemes [7,10,12,13] and attribute-based sanitizable signature schemes [4,6] have been proposed. The former uses unique public identity information of users, such as ID numbers, phone numbers, and email addresses, as public keys, while the latter associates ciphertext and keys with attribute sets and access structures. However, these schemes face the issue of key escrow. The Private Key Generation Center (PKG) holds the master key for generating all user private keys, making the security of user keys entirely dependent on the PKG. This poses significant challenges to data privacy protection. Certificateless cryptography, on the other hand, offers a solution to the key escrow problem. In certificateless cryptography, the Key Generation Center (KGC) collaborates with users to generate complete private keys, eliminating the need for a central authority like the PKG [5,9,15].

Contribution. The primary contributions of this paper can be succinctly summarized as follow: addressing the issues of data leakage by data sharing parties in sanitizable signature schemes, as well as the challenges of certificate management and key escrow, we propose a designated verifier certificateless sanitizable signature scheme based on the idea of Universal Designated Verifier Signatures (UDVS).

- Firstly, the signer generates a message and signs it, and then sends it to the sanitizer via a secure channel. The sanitizer acts as both the sanitizer of the message and the holder of the UDVS signature. They specify the designated verifier for the sanitized message and its signature according to their specific requirements. Only the designated verifier can validate and have confidence in the validity of the message and its signature, ensuring that the message information remains confidential to parties other than the designated verifier.
- Secondly, the proposed scheme builds upon certificateless signatures, eliminating the necessity for complex certificate management and resolving the key escrow issue.
- Thirdly, by conducting formal security analysis, we establish that the scheme effectively safeguards against malicious users and a malicious Key Generation Center (KGC), ensuring unforgeability, immutability, privacy, and non-transferability.
- Fourthly, both experimental results and theoretical analysis substantiate that the performance of the proposed scheme surpasses that of comparable schemes in terms of efficiency and effectiveness.

2 System Model and Security Goals

2.1 System Model

Figure 1 illustrates the system model of the proposed scheme in this chapter. The system consists of four entities: Key Generation Center (KGC), Signer, Sanitizer, and Verifier. Each entity is defined as follows:

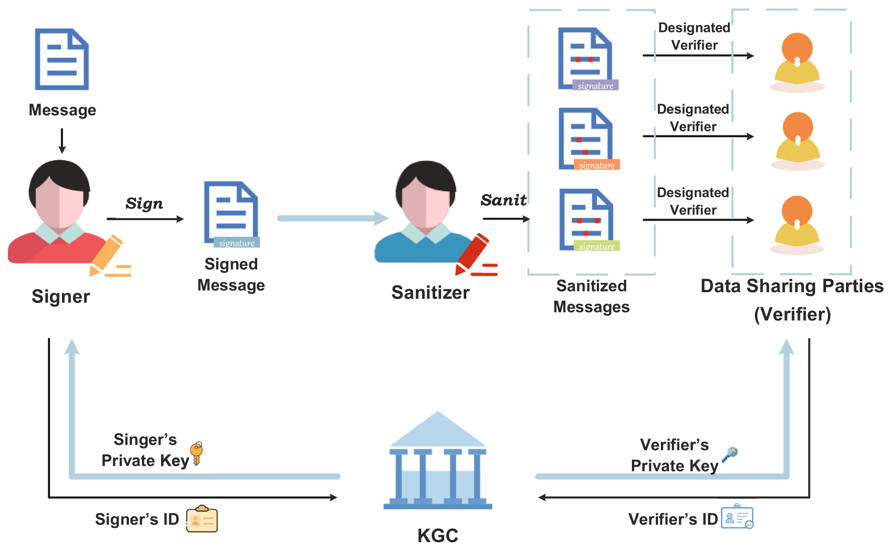


Fig. 1. System Model

- **Key Generation Center (KGC):** The Key Generation Center (KGC) assumes the responsibility of system initialization and generates the corresponding key pairs based on user identities. A segment of the user's private key is securely transmitted through a protected channel.
- **Signer:** The Signer assumes the responsibility for generating and signing message. They can define the content that can be sanitized and send the message and its signature to the Sanitizer through a secure channel.
- **Sanitizer:** The Sanitizer can sanitize sensitive data in the shared message, generate a new signature, and specify the Verifier to validate the sanitized message and its signature.
- **Verifier:** The Verifier is assigned by the data sharing party. Only the designated Verifier can validate the authenticity of the corresponding message.

2.2 Security Goals

In order to facilitate secure data sharing effectively, this scheme aims to accomplish the following objectives:

- **Unforgeability:** Only the signer, sanitizer, and designated verifier can generate valid signatures. No one else can forge a valid signature.
- **Immutability:** The sanitizer cannot modify the data blocks in the message that are not allowed to be sanitized.
- **Privacy:** Privacy guarantees that only the signer and sanitizer possess information regarding the sanitized portions of the data, preventing anyone else from obtaining such details.
- **Non-Transferability:** The designated verifier lacks the ability to prove the validity of the signature to third parties.

These objectives play a critical role in ensuring the security and integrity of the shared data while preserving the privacy of sensitive information.

3 The Proposed Scheme

3.1 Overview

This scheme implements a certificateless sanitizable signature scheme with designated verifiers, using the concepts of certificateless signatures and universal designated verifier signatures. In this scheme, the signer and verifiers obtain the certificateless signature key pair from the Key Generation Center (KGC). The signer first computes the certificateless signature on the message m and generates auxiliary information aux required for sanitization. The message/signature pair (m, s) and aux are then transmitted to the sanitizer through a secure channel. In the universal designated verifier signature scheme, the sanitizer assumes the role of the "signature holder". Upon receiving (m, s), the sanitizer first verifies its validity. If valid, the sanitizer utilizes the auxiliary information to sanitize the document, resulting in a sanitized document m' and a certificateless signature σ'. The sanitizer then designates a verifier for the message/signature pair (m', σ') and generates a designated verifier signature σ'' for it. This yields a message/signature pair (m', σ'') that can only be verified by the designated verifier.

3.2 Detail

The specific algorithm for the designated verifier sanitizable signature without certificates scheme is as follow:

$\boldsymbol{(pp, msk) \leftarrow Setup(\lambda)}$. Input the security parameter λ, the KGC executes the *Setup* algorithm to generate the system's public parameters PP and the system master key msk.

– Let the bilinear pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are respectively cyclic groups of prime order q for addition and multiplication. Let P be the generator of $\mathbb{G}$.
– Select the hash function $h_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_1, H_2 : \{0,1\}^* \rightarrow \mathbb{G}^*$.
– Randomly choose $s \in \mathbb{Z}_p^*$, then let $msk = s$ and compute the main public key $P_{pub} = sP$

KGC keeps the msk in secret and output the system public parameter $pp = \{q, e, \mathbb{G}, \mathbb{G}_T, P, P_{pub}, h_1, h_2, H_1, H_2, H_3\}$.

$\boldsymbol{(pk_{Sig/Vrf}, sk_{Sig/Vrf}) \leftarrow KGen(pp, msk, ID_{Sig/Vrf})}$**.** Input pp, msk and the identity of the signer ID_{Sig}, KGC interacts with the signer to execute the $KGen$ algorithm to generate the secret key pair (pk_{Sig}, sk_{Sig}) of the signer.

– Generating the secret value: the signer randomly chooses the secret value $s_{Sig} \in \mathbb{Z}_q^*$ and compute $P_{Sig} = s_{Sig}P$, $Q_{Sig} = H_1(ID_{Sig}||P_{Sig})$, and then sends Q_{Sig} to KGC.
– Generating public key: the signer generates the public key $pk_{Sig} = (P_{Sig}, Q_{Sig})$.
– Generating partial secret key: KGC computes the partial secret key of the signer $d_{Sig} = sQ_{Sig}$, and sends it to the signer through a secure channel.
– Generating private key: the signer generates the private key $sk_{Sig} = (d_{Sig}, s_{Sig})$.

And finally we get the secret key pair (pk_{Sig}, sk_{Sig}) of the signer.

Similarly, KGC interacts with the verifier and executes $(pk_{Vrf}, sk_{Vrf}) \leftarrow KGen(pp, msk, ID_{Vrf})$ to obtain the verifier's key pair (pk_{Vrf}, sk_{Vrf}).

$\boldsymbol{(m, \sigma, aux) \leftarrow Sign(pp, m, pk_{Sig}, sk_{Sig}, ID_{Sig}, ADM)}$**.** Input the system public parameter pp, message m, the secret key pair (pk_{Sig}, sk_{Sig}), the identity of the signer ID_{Sig} and ADM, the signer executes $Sign$ algorithm to generate the signature σ for m, and generates the required aux for the sanitizable data blocks.

– Divide m to n data blocks $m = m_1||m_2||...||m_n$.
– For each data block $m_i (i \in [1, n])$, randomly chooses $t_i \in \mathbb{Z}_q^*$ and computes the signature for m_i as follow: Compute $T_i = t_iP, \alpha_i = h_1(ID_{Sig}||P_{Sig}||m_i||T_i)$ and $W = H_2(P_{pub})$. Then compute $\sigma_i = d_{Sig} + \alpha_i \cdot t_i \cdot W + s_{Sig} \cdot W = d_{Sig} + (\alpha_i \cdot t_i \cdot s_{Sig}) \cdot W$. And for each $i \in ADM$, computes the transformation value $\chi_i = t_i \cdot W$.
– Finally, the signer obtains he message/signature pair (m, σ) and aux, where $\sigma = (\{\sigma_i\}_{i\in[1,n]}, \{T_i\}_{i\in[1,n]})$, $aux = (\{\chi_i\}_{i\in ADM}, ADM)$. The signer sends them to the sanitizer through a secure channel.

$\boldsymbol{0/1 \leftarrow Verify(pp, m, \sigma, pk_{Sig}, ID_{Sig})}$**.** Input the system public parameter pp, original message/signature pair (m, σ), the public key $pk_{Sig} = (P_{Sig}, Q_{Sig})$

of the signer and the identity of the signer ID_{Sig}, the sanitizer executes $Verify$ algorithm to authenticate the validity of the message/signature pair (m, σ).

The sanitizer computes $W = H_2(P_{pub})$. For each $i \in [1, n]$, computes $\alpha_i = h_1(ID_{Sig}||P_{Sig}||m_i||T_i)$. And then computes $\sigma = \Sigma_{i=1}^{n}\sigma_i, T = \Sigma_{i=1}^{n}\alpha_i \cdot T_i$ and verifies:

$$e(\sigma, P) = e(P_{pub}, Q_{Sig})^n \cdot e(T, W) \cdot e(P_{Sig}, W)^n \tag{1}$$

is satisfied or not. If the condition is true, it indicates that (m, σ) is a valid message/signature pair and the output is 1. Otherwise, the output is 0.

$\boldsymbol{(m', \sigma') \leftarrow Sanit(pp, m, \sigma, pk_{Sig}, ID_{Sig}, MOD, aux)}$. Input the system public parameter pp, original message/signature pair (m, σ), the public key $pk_{Sig} = (P_{Sig}, Q_{Sig})$ of the original signer, the identity of the signer ID_{Sig}, the description MOD of the data block to be sanitized and aux, the sanitizer executes $Sanit$ algorithm to sanitize the message m and generates a valid signature σ' for the sanitized message m'.

- If $ADM(MOD) = 0$, output $\perp$.
- Sanitize m to obtain $m' = MOD(m) = \{m'\}_{i\in[1,n]}$, where for $i \in MOD, m'_i \neq m_i$. Otherwise $m'_i = m_i$.
- For each data block $m'_i (i \in [1, n])$, compute sanitized signature σ': If $i \notin MOD$, then $\sigma'_i = \sigma_i$; if $i \in MOD$, then computes $\alpha'_i = h_1(ID_{Sig}||P_{Sig}||m_i||T_i)$ and $\alpha'_i = h_1(ID_{Sig}||P_{Sig}||m'_i||T_i)$, then $\sigma'_i = \sigma_i - \alpha_i \cdot \chi_i + \alpha'_i \cdot \chi_i = \sigma_i + (\alpha'_i - \alpha_i) \cdot \chi_i$.

After the sanitization, fo each $i \in [1, n], \sigma'_i = d_{Sig} + \alpha'_i \cdot t_i \cdot W + s_{Sig} \cdot W$. And finally, the sanitizer obtains the sanitized message/signature pair (m', σ'), where $\sigma' = (\{\sigma'_i\}_{i\in[1,n]}, \{T_i\}_{i\in[1,n]})$.

$\boldsymbol{(m', \sigma'') \leftarrow DSign(pp, m', \sigma', pk_{Vrf})}$. Input the system public parameter pp, sanitized message/signature pair (m', σ') and the public key $pk_{Vrf} = (P_{Vrf}, Q_{Vrf})$ of the designated verifier, the sanitizer execute $DSign$ to assign verifier to (m', σ').

For each signature $\sigma'_i (i \in [1, n])$, the sanitizer compute $\bar{\sigma}_i = e(\sigma', P_{Vrf})$. Finally, it outputs the message/signature pair (m', σ'') of the designated verifier, where $\sigma'' = (\{\bar{\sigma}_i\}_{i\in[1,n]}, \{T_i\}_{i\in[1,n]})$.

$\boldsymbol{0/1 \leftarrow DVerify(pp, m', \sigma'', pk_{Sig}, ID_{Sig}, sk_{Vrf})}$. Input the system public parameter pp, message/signature pair (m', σ'') of the designated verifier, the public key of the signer $pk_{Sig} = (P_{Sig}, Q_{Sig})$, the identity of the signer ID_{Sig} and the private key of the designate verifier $sk_{Vrf} = (d_{Vrf}, s_{Vrf})$, the designated verifier execute $DVerify$ algorithm to authenticate the validity of the message/signature pair (m', σ'').

The designated verifier computes $W = H_2(P_{pub})$. For each $i \in [1, n]$, computes $\alpha'_i = h_1(ID_{Sig}||P_{Sig}||m'_i||T_i)$. And then computes $\bar{\sigma} = \Pi_{i=1}^{n}\bar{\sigma}_i, T = \Sigma_{i=1}^{n}\alpha_i \cdot T_i$, and verifies:

$$\bar{\sigma} = [e(P_{pub}, Q_{Sig})^n \cdot e(T', W) \cdot e(P_{Sig}, W)^n]^{s_{Vrf}} \tag{2}$$

is satisfied or not. If the condition is true, it indicates that (m', σ'') is a valid message/signature pair and the output is 1. Otherwise, the output is 0.

3.3 Security Analysis

Unforgeability. This scheme exhibits unforgeability against adaptive chosen message attacks and selective identity attacks. Below, we provide security analysis for forgery attacks from two types of adversaries: $\mathcal{A}_I$ and $\mathcal{A}_{II}$. Assuming a Computational Diffie-Hellman (CDH) problem instance is given $(P, Q_1 = aP, Q_2 = bP, Q_3 = abP)$.

For $\mathcal{A}_I$ type forgeries, the attacker does not have access to the master key s of the system but can replace a legitimate public key of the user. Let's assume an $\mathcal{A}_I$ attacker replaces the public key $pk_s = (P_s, Q_s)$ of a legitimate user ID_s with $pk_s^* = (P_s^*, Q_s^*)$. Using the replaced public key, the attacker successfully forges a signature (m^*, σ^*) for a message m. Let $P_{pub} = Q_1 = aP, Q_s^* = \vartheta_s Q_2 = \vartheta_s bP, W = \omega P$. Computes $\sigma = \Sigma_{i=1}^n \sigma_i, T = \Sigma_{i=1}^n \alpha_i \cdot T_i$, and then sign and verify the equation $e(\sigma, P) = e(P_{pub}, Q_{Sig})^n \cdot e(T, W) \cdot e(P_{Sig}, W)^n$ is satisfied. We have:

$$\begin{aligned}
&e(\sigma_i^*, P) = e(P_{pub}, Q_S) \cdot e(\alpha_i^* \cdot T_i, W) \cdot e(P_S, W) \\
&\Rightarrow e(P_{pub}, Q_S) = e(\sigma_i^*, P) \cdot [e(\alpha_i^* \cdot T_i, W) \cdot e(P_S, W)]^{-1} \\
&\Rightarrow e(aP, \vartheta_S bP) = e(\sigma_i^*, P) \cdot [e(\alpha_i^* \cdot T_i, \omega P) \cdot e(P_S, \omega P)]^{-1} \\
&\Rightarrow e(\vartheta_S \cdot abP, P) = e(\sigma_i^*, P) \cdot [e(\omega \cdot \alpha_i^* \cdot T_i, P) \cdot e(\omega \cdot P_S, P)]^{-1} \\
&\Rightarrow abP = \vartheta_S^{-1} (\sigma_i^* - \omega \cdot \alpha_i^* \cdot T_i - \omega \cdot P_S)
\end{aligned}$$

By successfully solving the CDH problem using adversary $\mathcal{A}_I$, the CDH problem itself is resolved. However, the CDH problem is a difficult problem that cannot be currently solved in the real world. Therefore, the forgery attack by adversary $\mathcal{A}_I$ cannot succeed.

For adversary $\mathcal{A}_{II}$, this particular adversary possesses access to the system master key s, but lacks the capability to substitute legitimate the public keys of the user. Suppose an $\mathcal{A}_{II}$ adversary successfully performs a forgery attack using the public key $pk_s = (P_s, Q_s)$ of user ID_s on message m, resulting in a forged signature (m^*, σ^*). Let $P_s = Q_1 = aP, W = \omega bP$. Computes $\sigma = \Sigma_{i=1}^n \sigma_i, T = \Sigma_{i=1}^n \alpha_i \cdot T_i$, and then sign and verify the equation $e(\sigma, P) = e(P_{Pub}, Q_{Sig})^n \cdot e(T, W) \cdot e(P_{Sig}, W)^n$ is satisfied. We have:

$$\begin{aligned}
&e(\sigma_i^*, P) = e(P_{pub}, Q_S) \cdot e(\alpha_i^* \cdot T_i, W) \cdot e(P_S, W) \\
&\Rightarrow e(P_S, W) = e(\sigma_i^*, P) \cdot [e(P_{pub}, Q_S) \cdot e(\alpha_i^* \cdot T_i, W)]^{-1} \\
&\Rightarrow e(aP, \omega bP) = e(\sigma_i^*, P) \cdot [e(sP, Q_S) \cdot e(\alpha_i^* \cdot T_i, W)]^{-1} \\
&\Rightarrow e(\omega \cdot abP, P) = e(\sigma_i^*, P) \cdot [e(sQ_S, P) \cdot e(\alpha_i^* \cdot t_i \cdot W, P)]^{-1} \\
&\Rightarrow abP = (\omega)^{-1} (\sigma_i^* - sQ_S - \alpha_i^* \cdot t_i \cdot W)
\end{aligned}$$

The adversary $\mathcal{A}_{II}$, as a subroutine, successfully solves the CDH problem. However, it is based on the security assumption that the CDH problem is difficult and currently unsolved in the real world. Therefore, the forgery attack by the adversary $\mathcal{A}_{II}$ cannot be successful.

Immutability. In this scheme, only the data blocks that are allowed to be sanitized that $i \in ADM$, have corresponding computation transformation values χ_i. Data blocks that are not allowed to be sanitized do not have corresponding χ_i values in the scheme. If the sanitizer wants to sanitize the data block, it needs to compute $\sigma_i' = \sigma_i - \alpha_i \cdot \chi_i + \alpha_i' \cdot \chi_i = \sigma_i + (\alpha_i' - \alpha_i) \cdot \chi_i$, which requires the transformation value χ_i, while the data blocks that are not allowed to be sanitized $\sigma_i' = d_{Sig} + \alpha_i' \cdot t_i \cdot W + s_{Sig} \cdot W = d_{Sig} + \alpha_i' \cdot \chi_i + s_{Sig} \cdot W$, cannot obtain the value of t_i through computation, and $\chi_i = t_i \cdot W$ can be regard as a Discrete Logarithm (DL) problem. According to the DL problem, the Sanitizer cannot obtain t_i. Therefore, the Sanitizer is unable to perform sanitization on these data blocks. As a result, the proposed scheme satisfies the immutability property.

Privacy. Verification parties must not have the ability to deduce sensitive information from the sanitized message/signature pairs. Because this scheme is designated for specific verification parties, the confirmation of the validity of the message/signature pairs is restricted solely to the designated parties, and cannot be accomplished by any other entity, and therefore, the information that has been sanitized cannot be recovered. However, for the designated verification parties, the message/signature pairs (m', σ'') do not involve information about the sanitized part of the message $m_i (m_i \neq m_i')$. As a result, this scheme meets the privacy requirement.

Non-transferability. Non-Transferability refers to the property that given a message/signature pair (m', σ'') for a designated verifier, no one other than the designated verifier can determine whether (m', σ'') was generated by the sanitizer or the designated verifier, even if they have knowledge of all users' private keys. This is because the designated verifier can simulate the generation of a signature counterpart σ^*, and this counterpart cannot be distinguished from the signature σ'' output by the sanitizer. The process of the designated verifier simulating the signature is as follows:

The designated verifier computes $W = H_2(P_{pub})$. For each $i \in [1, n]$, randomly chooses $T_i^* \in \mathbb{G}$ and computes $\alpha_i^* = h_1(ID_{Sig} || P_{Sig} || m_i' || T_i^*)$. Then computes $T^* = \Sigma_{i=1}^{n} \alpha_i^* \cdot T_i^*$, and outputs a signature counterpart $\sigma^* = \bar{\sigma}^*, \{T_i^*\}_{i \in [1,n]}$ for the message m', where $\bar{\sigma}^* = e(s_{Vrf} \cdot P_{pub}, Q_{Sig})^n \cdot e(s_{Vrf} T^*, W) \cdot e(s_{Vrf} \cdot P_{Sig}, W)^n$. This counterpart satisfied the correctness:

$$\begin{aligned}
\bar{\sigma}^* &= e(s_{Vrf} \cdot P_{pub}, Q_{Sig})^n \cdot e(s_{Vrf} \cdot T^*, W) \cdot e(s_{Vrf} \cdot P_{Sig}, W)^n \\
&= e(P_{pub, Q_{Sig}})^{s_{Vrf} \cdot n} \cdot e(T^*, W)^{s_{Vrf}} \cdot e(P_{Sig}, W)^{s_{Vrf} \cdot n} \\
&= [e(Q_{Sig}, P_{pub})^n \cdot e(T^*, W) \cdot e(P_{Sig}, W)^n]^{s_{Vrf}}
\end{aligned}$$

Therefore, for message m', the designated verifier generates a signature counterpart σ^* that is indistinguishable from the signature σ'' produced by the sanitizer.

4 Performance Analysis

In this section, we conduct a comparison between our scheme and the identity-based revocable proxy signature scheme [13] and the certificateless revocable proxy signature scheme [16] in terms of theoretical and experimental evaluations.

4.1 Theory Comparison

Table 1. Function Comparison

Scheme	Unforgeability	Immutability	Certificate Management Problems	Key Escrow Problems	Designated verifier
[13]	✓	×	✓	×	×
[16]	–	✓	✓	✓	×
Ours	✓	✓	✓	✓	✓

Table 1 shows the difference of the features between our proposed scheme and the related purifiable signature schemes [13] and [16]. The following analysis is conducted on these features:

– Unforgeability is a fundamental and important property that ensures that individuals without the signing key cannot generate valid signatures. Scheme [13] and our proposed scheme both satisfy this property, while scheme [16] does not define or discuss this property.
– Immutability ensures that the sanitizer can only sanitize data within the range specified by the signer. In scheme [13], the signer computes transformation values for sanitization, and the sanitizer with access to these values can modify any data block, violating the property of immutability. Scheme [16] and our proposed scheme both satisfy this property.
– Traditional public key encryption involves users generating public/private key pairs and having their public keys certified by a Certification Authority (CA), resulting in expensive certificate management overhead. Scheme [13] is based on identity-based cryptography, while schemes [16] and our proposed scheme are based on certificateless cryptography, eliminating this problem.
– The identity-based scheme [13] relies on a private key generator (PKG) to generate the private keys for the user, resulting in a key escrow problem. The certificateless cryptography schemes [16] and our proposed scheme do not have this problem.
– Only designated verifiers are allowed to verify the validity of message/signature pairs. Except for our proposed scheme, the other two sanitizable signature schemes do not support designated verifiers.

Based on the aforementioned information, we can conclude that our proposed scheme surpasses existing schemes in several aspects. It eliminates the need for costly certificate management, resolves the challenges associated with key escrow, guarantees non-forgeability and immutability, and provides support for designated verifiers.

Table 2. Single Execution Time of Each Operation

Notations	Description	Single execution time (ms)
$pair$	The operation of bilinear pairing	4.50315
$Exp_{\mathbb{G}_T}$	The exponentiation operation on $\mathbb{G}_T$	0.55745
$Mul_{\mathbb{G}_T}$	The multiplication operation on $\mathbb{G}_T$	0.00515
$Add_{\mathbb{G}}$	The addition operation on $\mathbb{G}$	0.04015
$Mul_{\mathbb{G}}$	The multiplication operation on $\mathbb{G}$	0.04015
$Hash_{\mathbb{G}}$	The hash operation on $\mathbb{G}$	18.6542

Due to the computational overhead mainly focused on operations in groups $\mathbb{G}$ and $\mathbb{G}_T$, operations on $\mathbb{Z}_q$ are ignored in this context. To directly compare the computational costs of different schemes, we tested the execution time of various operations based on the JPBC library. The experimental environment consists of an Intel(R) Core(TM) i7-10875H CPU @ 2.30 GHz, 16 GB RAM, and Windows 10(x64) OS. Table 2 provides the symbol descriptions and the time overhead of single executions for various operations. Additionally, the number of data blocks is represented as n, the size of the set of data blocks that can be sanitized (ADM) is denoted as l_{adm}, and the size of the set of data blocks to be sanitized (MOD) is denoted as l_{mod}.

Table 2 presents a comparison of the computational costs between the sanitizable signature scheme [13], the scheme [16], and the proposed scheme. Since the scheme [13] and the scheme [16] do not support specified verifier, only the proposed scheme provides the costs for the specified verifier signature algorithm and the specified verifier verification algorithm. From the table, it can be observed that the costs of signature and verification are linearly related to the number of data blocks, denoted as n. The cost of sanitization exhibits a linear relationship with either the number of blocks that can be sanitized (l_{adm}) or the number of blocks to be sanitized (l_{mod}). In the proposed scheme, the costs associated with designated verifier signature and designated verifier verification also demonstrate a linear increase as the number of data blocks, n, grows.

In the signature generation phase, compared to scheme [13], the proposed scheme reduces n $Add_{\mathbb{G}}$ operations, $(n - l_{adm})$ $Mul_{\mathbb{G}}$ operations, and $(2n - 1)$ $Hash_{\mathbb{G}}$ operations. Compared to scheme [16], the proposed scheme reduces n $Mul_{\mathbb{G}}$ operations and $(n - 1)$ $Hash_{\mathbb{G}}$ operations. Since both $Hash_{\mathbb{G}}$ and $Mul_{\mathbb{G}}$ have significant costs, overall, the proposed scheme exhibits significantly lower signature generation costs compared to scheme [13] and scheme [16].

In the sanitization phase, since $l_{mod} \leq l_{adm} \leq n$, the computing consumption of our scheme and scheme [13] are the same and generally lower than scheme [16]. In the verification phase, the proposed scheme reduces computational costs compared to scheme [13] by $2Add_{\mathbb{G}} + Mul_{\mathbb{G}} + (n-1)Hash_{\mathbb{G}} - Mul_{\mathbb{G}_T} - Exp_{\mathbb{G}_T} - Pair$, and this difference increases linearly with n. The proposed scheme also reduces computing consumption compared to scheme [16] by $nAdd_{\mathbb{G}} + 2Mul_{\mathbb{G}} + (n-1)Hash_{\mathbb{G}} - Mul_{\mathbb{G}_T} - 2Exp_{\mathbb{G}_T} - Pair$, and this difference also increases linearly with n.

Overall, Our proposed scheme exhibits lower overhead compared to scheme [13] and scheme [16], and the difference between them increases linearly with the increase in n. Furthermore, our scheme surpasses both scheme [13] and scheme [16] in terms of security and functionality.

Table 3. Computation Cost Comparison Between Sanitizable Signature Scheme

Algorithm	[13]	[16]	Ours
Signature	$2nAdd_{\mathbb{G}} + 3nMul_{\mathbb{G}} + 2nHash_{\mathbb{G}}$	$nAdd_{\mathbb{G}} + (3n + l_{adm})Mul_{\mathbb{G}} + nHash_{\mathbb{G}}$	$nAdd_{\mathbb{G}} + (2n + l_{adm})Mul_{\mathbb{G}} + Hash_{\mathbb{G}}$
Verification	$2nAdd_{\mathbb{G}} + (n + 1)Mul_{\mathbb{G}} + nHash_{\mathbb{G}} + Mul_{\mathbb{G}_T} + Exp_{\mathbb{G}_T} + 3Pair$	$(3n-2)Add_{\mathbb{G}} + (n + 2)Mul_{\mathbb{G}} + nHash_{\mathbb{G}} + Mul_{\mathbb{G}_T} + 3Pair$	$(2n-1)Add_{\mathbb{G}} + nMul_{\mathbb{G}} + Hash_{\mathbb{G}} + 2Mul_{\mathbb{G}_T} + 2Exp_{\mathbb{G}_T} + 4Pair$
Sanitization	$l_{mod}Add_{\mathbb{G}} + l_{mod}Mul_{\mathbb{G}}$	$l_{adm}Add_{\mathbb{G}} + l_{adm}Mul_{\mathbb{G}}$	$l_{mod}Add_{\mathbb{G}} + l_{mod}Mul_{\mathbb{G}}$
Designated verifier signature	–	–	$nPair$
Designated verifier verification	–	–	$(n-1)Add_{\mathbb{G}} + nMul_{\mathbb{G}} + Hash_{\mathbb{G}} + (n+1)Mul_{\mathbb{G}_T} + 3Exp_{\mathbb{G}_T} + 3Pair$

4.2 Experiment Comparison

In this section, we conducted performance testing of our proposed scheme using the JPBC library and compared it with scheme [13] and scheme [16]. The experiments were conducted on an Intel(R) Core(TM) i7-10875H CPU @ 2.30 GHz, 16 GB RAM, and Windows 10 (x64) OS. All three schemes were implemented based on the supersingular curve SS512, which achieves an 80-bit security level.

Figure 2 shows the computational overhead of each algorithm in our scheme. For this test, we set the number of data block n, to 50, and the total number of blocks that can be sanitized, l_{adm}, and the total number of blocks to be sanitized, l_{mod}, both to 20. As shown in Fig. 2, the required time for signature, verification, sanitization, designated verifier, and designated verifier verification algorithms were 0.92 s, 0.41 s, 0.15 s, 0.21 s, and 0.41 s, respectively. Among them, the signature algorithm has the highest overhead, while the verification algorithm and the designated verifier verification algorithm have similar overheads.

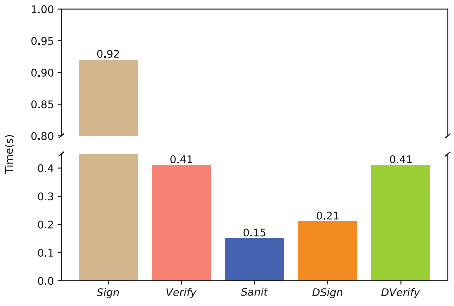
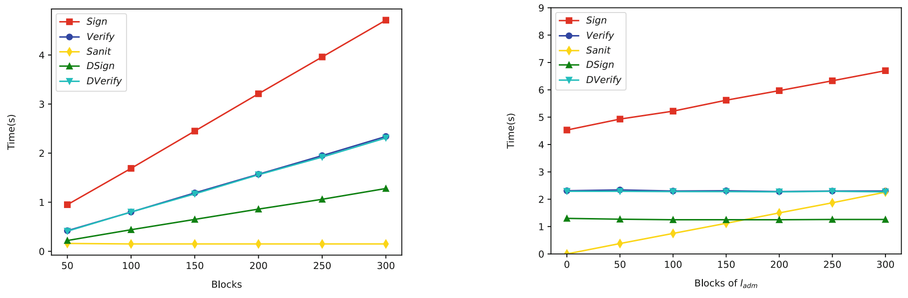


Fig. 2. Algorithm Computation Cost of This Scheme

Figure 3a demonstrates the impact of the number of data blocks, n, on the computational overhead of our proposed scheme. With $l_{adm} = 20$, we gradually increase n from 50 to 300 and test the computational overhead of each algorithm. It can be observed that the sanitization overhead remains constant and unaffected by n. On the contrary, the computational overhead of the remaining algorithms experiences a linear increase as n increases. According to the analysis in Table 3, the signature, verification, and designated verifier verification algorithms involve time-consuming $Mul_{\mathbb{G}}$ and $Hash_{\mathbb{G}}$, resulting in higher computational overhead compared to the designated verifier signature algorithm, which only requires $Pair$ operations. Additionally, among the signature overhead, verification overhead, and designated verifier verification overhead, the coefficient of $Mul_{\mathbb{G}}$ is larger, indicating a more pronounced growth trend in the signature overhead.



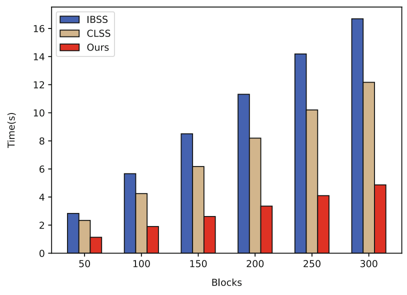
(a) Impact of n

(b) Impact of l_{adm}

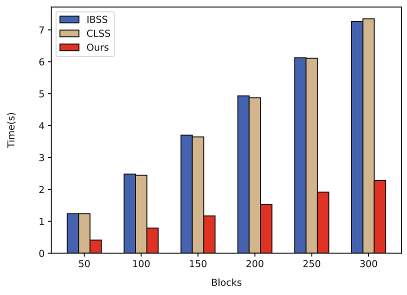
Fig. 3. Impact of n and l_{adm}

Figure 3b demonstrates the impact of the total number of sanitizable data blocks, l_{adm}, on the computational cost of our scheme. We set n = 300 and gradually increase l_{adm} from 0 to 300, while keeping $l_{mod} = l_{adm}$. We test the computational cost of various algorithms. It shows that only the signature cost and sanitization cost increase linearly with l_{adm}, while the costs of other algorithms remain relatively constant. According to the analysis in Table 3, the signature algorithm requires l_{adm} multiplications on $\mathbb{G}$ to generate the auxiliary information required for sanitization, and the sanitization algorithm requires l_{adm} additions and multiplications on $\mathbb{G}$ to generate the sanitizable signature. Since the cost of $Add_{\mathbb{G}}$ is negligible and the cost of $Mul_{\mathbb{G}}$ is approximately 8ms, the signature cost and sanitization cost increase by approximately 2.4 s when $l_{adm} = 300$ compared to $l_{adm} = 0$, under the current settings.

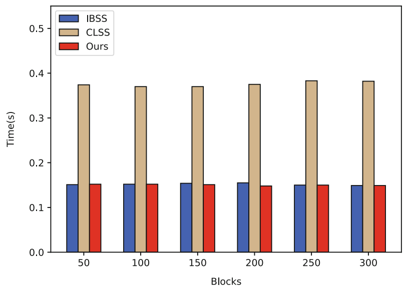
Figure 4 illustrates the computational costs of the signature, verification, and sanitization algorithms for [13], scheme [16], and our proposed scheme. Here, we set $l_{adm} = 50$, $l_{mod} = 20$, and gradually increase n from 50 to 300.



(a) Signature



(b) Verification



(c) Sanitization

Fig. 4. Computation Cost Comparison of Algorithm Between IBBS, CLSS and Ours

As shown in Fig. 4a, the signature cost of the proposed scheme is significantly lower than that in the IBSS scheme and the CLSS scheme, and the gap between them increases with the increase of n. When $n = 300$, the signature cost of our proposed scheme is approximately 11.8 s less than the IBSS scheme and approximately 7.3 s less than the CLSS scheme.

As shown in Fig. 4b, the cost in verification of the IBSS scheme and the CLSS scheme remains similar, while the verification cost of the proposed scheme is significantly lower than these two schemes. Moreover, this difference increases noticeably with the increase of n, which is consistent with the analysis in Table 3. When $n = 300$, the difference between our proposed scheme and the IBSS scheme is approximately 4.98 s, and the difference between our proposed scheme and the CLSS scheme is approximately 5.07 s.

As shown in Fig. 4c, the sanitization cost of the proposed scheme is the same as that in the IBSS scheme and both are lower than the CLSS scheme. According to the analysis in Table 3, the sanitization cost of our proposed scheme and the IBSS scheme is related to l_{mod}, while the CLSS scheme depends on l_{adm}. Therefore, when $l_{adm} \geq l_{mod}$, the sanitization cost of the CLSS scheme is the highest among the three schemes. When $l_{adm} = l_{mod}$, the sanitization cost is the same for all three schemes.

Through experimental comparisons, it can be observed that the performance of our proposed solution is not inferior and even superior to existing sanitizable signature schemes. Our approach is based on certificateless signatures, which avoids the key escrow issue present in the IBSS scheme. In comparison to the CLSS scheme, our solution incorporates an additional mechanism for designated verifiers, addressing the concern of information leakage on the verifier's side and further ensuring the privacy and security of shared data.

5 Conclusion

In order to ensure the privacy of shared data, our proposed sanitizable signature scheme addresses the issues of information leakage by data sharers, the costly certificate management overhead, and the potential security risks associated with key escrow in existing schemes.

Firstly, our solution introduces a certificateless approach for sanitizable signatures. The public key is derived from the public information of the user, and the private key is generated jointly by the user and the KGC, eliminating the need for a dedicated organization to manage user certificates and preventing the compromise of user private keys by the KGC, thus safeguarding data security.

Secondly, our scheme supports designated verifiers. After generating multiple sanitized versions of a message based on different usage scenarios, the sanitizer can designate a verifier for each version. The designated verifier represents the current data sharer, ensuring that only the sharer can verify the validity of the corresponding sanitized message, thereby preventing the leakage of message information to others.

Finally, through theoretical and experimental comparisons with similar sanitizable signature schemes, we have demonstrated that our proposed scheme offers improved security and functionality, while also being more efficient than existing solutions.

References

1. Bilzhause, A., Pöhls, H.C., Samelin, K.: Position paper: the past, present, and future of sanitizable and redactable signatures. In: Proceedings of the 12th International Conference on Availability, Reliability and Security, pp. 1–9 (2017)
2. Deng, L., Yang, Y., Gao, R.: Certificateless designated verifier anonymous aggregate signature scheme for healthcare wireless sensor networks. IEEE Internet Things J. **8**(11), 8897–8909 (2021)
3. Centers for Disease Control and Prevention: HIPAA privacy rule and public health. Guidance from CDC and the US department of health and human services. MMWR Morb. Mortal. Weekly Rep. **52**(Suppl. 1), 1–17 (2003)
4. Hou, H., Ning, J., Zhao, Y., Deng, R.H.: Fine-grained and controllably editable data sharing with accountability in cloud storage. IEEE Trans. Dependable Secure Comput. **19**(5), 3448–3463 (2021)
5. Hussain, S., Ullah, S.S., Ali, I., Xie, J., Inukollu, V.N.: Certificateless signature schemes in industrial internet of things: a comparative survey. Comput. Commun. **181**, 116–131 (2022)
6. Liu, X., Ma, J., Xiong, J., He, T., Li, Q.: Attribute-based sanitizable signature scheme in cloud computing environment. J. Electron. Inf. Technol. **36**(7), 1749–1754 (2014)
7. Liu, Z., Ren, L., Li, R., Liu, Q., Zhao, Y.: Id-based sanitizable signature data integrity auditing scheme with privacy-preserving. Comput. Secur. **121**, 102858 (2022)
8. Ma, J., Liu, J., Huang, X., Xiang, Y., Wu, W.: Authenticated data redaction with fine-grained control. IEEE Trans. Emerg. Top. Comput. **8**(2), 291–302 (2017)
9. Ma, K., et al.: An efficient certificateless signature scheme with provably security and its applications. IEEE Syst. J. (2023)
10. Ming, Y., Shen, X., Peng, Y.: Identity-based sanitizable signature scheme in the standard model. In: Zhu, R., Zhang, Y., Liu, B., Liu, C. (eds.) ICICA 2010. CCIS, vol. 105, pp. 9–16. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16336-4_2
11. Ng, S., Tauber, T., Cheung, L.: ECDSA-compatible privacy preserving signature with designated verifier. In: 2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP), pp. 84–89. IEEE (2021)
12. Shen, W., Qin, J., Yu, J., Hao, R., Hu, J.: Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. IEEE Trans. Inf. Forensics Secur. **14**(2), 331–346 (2018)
13. Xu, Y., Ding, L., Cui, J., Zhong, H., Yu, J.: PP-CSA: a privacy-preserving cloud storage auditing scheme for data sharing. IEEE Syst. J. **15**(3), 3730–3739 (2020)
14. Xu, Z., Luo, M., Kumar, N., Vijayakumar, P., Li, L.: Privacy-protection scheme based on sanitizable signature for smart mobile medical scenarios. Wirel. Commun. Mob. Comput. **2020** (2020)
15. Xu, Z., Wu, L., Li, L., He, D.: A new certificateless generalized designated verifier aggregate signature scheme. J. Commun. (2017)
16. Zhang, W., Yan, Y., Wu, Y., Hu, J.: Certificateless revocable proxy signature scheme in cloud storage. Comput. Syst. Appl. **1**, 281–287 (2022)