Sanitizable and Accountable Endorsement for Dynamic Transactions in Fabric

Zhaoman Liu, Jianting Ning, Huiying Hou, and Yunlei Zhao

***Abstract*—Hyperledger Fabric, an open-source, enterprise-grade consortium platform, employs an endorsement policy wherein a set of endorsers signs transaction proposals from clients to confirm their authenticity. The signatures from endorsers constitute the core component of endorsement. However, when dealing with dynamic transactions with high timeliness and frequent updates (e.g., stock trading, real-time ad delivery, news reporting, etc.), the current endorsement process somewhat slows down the transaction execution. Meanwhile, handling these continuously updated transactions consumes significant resources from endorsers, thereby constraining overall application efficiency.**

To address these issues, this paper devises a novel sanitizable and accountable endorsement scheme by proposing a sanitizable multi-signature (SMS) as the theoretical tool. Specifically, we introduce the novel concept of sanitizable multi-signature and detail its instantiation. SMS combines the advantages of multi-signature and sanitizable signature, maintaining the compactness of the signature while allowing the sanitizer to adjust the initial endorsement result to fit the updated transaction content without interacting with the endorsers, so that both the authenticity and timeliness of transactions can be ensured. Additionally, SMS incorporates an innovative accountability mechanism to trace instances of improper data updates, thereby enhancing the security and reliability of the endorsement process.

We demonstrate the security of the proposed scheme through rigorous security analysis. Performance evaluations show that SMS can significantly reduce verification overhead and transaction size compared to the default ECDSA scheme in Fabric. Specifically, when verifying multiple endorsers' endorsements, our scheme exhibits a storage space reduction by approximately 30%-40% and a verification time reduction ranging from 9.2% to nearly 26.3%.

***Index Terms*—Fabric, endorsement, dynamic transaction, sanitizable multi-signature, accountability.**

I. Introduction

HYPERLEDGER Fabric is an open-source consortium blockchain platform specifically designed for enterprise applications, which distinguishes itself through its innovative system architecture that decouples transaction execution from ledger updates. By employing an innovative endorsement and validation mechanism, Fabric ensures that only transactions certified by a specific group of endorsers are committed to the blockchain [1]. This mechanism not only enhances transaction security but also provides the flexibility and scalability necessary for diverse application scenarios.

In a myriad of economic activities, dynamic transactions like stock trading, real-time ad delivery, timely news reporting and government document revisions are widespread, which require high timeliness for data updates. When Fabric is employed to manage these transactions, it is objectively required to ensure both transaction security and rapid data processing for timely updates [2]. However, Fabric's endorsement mechanism exhibits significant inefficiencies in handling such transactions. Specifically, continuously auditing these updates is not only cumbersome but also requires endorsers to remain perpetually online. If any endorser experiences a failure or goes offline, it directly results in increased processing delays. Furthermore, the constant handling of dynamic transaction updates consumes a significant amount of resources from the endorsers, further degrading the overall system performance. Consequently, optimizing the endorsement process for dynamic transactions has become an imperative challenge that must be addressed.

The concept of sanitizable signature was initially introduced by Ateniese et al. [3], which aims to permit an authorized third party, also known as a sanitizer, to modify signatures in a controlled and non-interactive manner without affecting their verifiability. This capability enables a variety of applications [3–5] and presents potential benefits when employed as an endorsement tool for dynamic transactions. By mitigating the need for repetitive endorsements during transaction updates, it can enhance the overall efficiency of transaction processing. Meanwhile, multi-signature, which compresses multiple signatures into a singular one, plays a crucial role in blockchain applications due to its ability to effectively reduce transaction size and enhance block utilization [6–8]. Integrating this feature with sanitizable signature to create an efficient endorsement mechanism for dynamic transactions is a promising area that has not yet been fully explored. Such an integration could further enhance transaction efficiency. However, this mechanism also introduces the risk of malicious modifications by either signers or sanitizers, with the potential for mutual blame. For instance, in stock trading, a portfolio manager might manipulate stock price fluctuations for personal gain and shift the blame to their investment bank. In news reporting, a news editor might intentionally exaggerate or distort facts about an event to attract traffic and then accuse the news platform. Therefore, designing a sanitizable multi-signature that supports accountability for malicious users is also a crucial consideration for endorsing dynamic transactions.

In this study, we firstly introduce the concept of sanitizable Multi-Signatures (SMS) and design its instantiation in detail. Based on this scheme, we further devise an efficient endorsement mechanism tailored for dynamic transaction, which not only represents a significant theoretical innovation but also further broadens Fabric's application scenarios. Our contributions can be summarized as follows.

A. Contribution

- we introduce the novel concept of SMS, which merges the advantages of sanitizable signature and multi-signature. This innovation preserves the compactness of signatures while incorporating the role of sanitizers, enabling efficient management of dynamic updates. Specifically, sanitizers can adjust the initial endorsement to reflect updated transaction content without requiring interaction with the original endorsers, thereby ensuring the authenticity, timeliness, and usability of transactions.
- Based on the SMS scheme, we devise a sanitizable and accountable endorsement mechanism that offers a more efficient solution for auditing dynamic transactions with high temporal requirements. This mechanism utilizes a lightweight endorsement scale, significantly streamlining the audit process for dynamically adjusted transactions. Additionally, the proposed framework includes an accountability feature that can trace malicious modification actions, effectively preventing fraudulent transactions and mutual scapegoating.
- We conducted a comprehensive performance analysis of the proposed SMS scheme and evaluated its practical application in the Fabric platform. Compared to the existing ECDSA [9] scheme, our scheme significantly reduces time and storage overhead when verifying endorsements from multiple endorsers. Specifically, with 10 endorsers, our scheme can reduce verification time by 9.2%; with 1000 endorsers, it reduces by 26.28%. Furthermore, we demonstrated that SMS can substantially decrease storage space usage both at the individual transaction and block levels, achieving reduction rates of approximately 30% and 40%, respectively. While we acknowledge the potential value of comparing our scheme to other sanitizable or multi-signature schemes, focusing on ECDSA, which is the default and standard algorithm in Fabric, underscores the practical and immediate enhancements our SMS scheme brings to Fabric. This choice emphasizes the tangible benefits of adopting SMS in existing frameworks.

B. Organization

The rest of this paper is organized as follows. The related work is summarized in Section II, followed by the preliminaries in Section III. In Section IV, we outline the construction idea of SMS, and gives its formal definition, security model, general construction and instantiation in Section V. After that, We describe how to apply the proposed scheme in Fabric in Section VI, and give a performance evaluation and implementation on it in Section VII. Finally, we conclude our work in Section VIII.

II. RELATED WORK

A. Sanitizable Signature

Since the pioneering work of Ateniese et al.[3] on sanitizable signature, this field has attracted extensive academic attention. Bilzhause et al.[10] have categorized sanitizable signature schemes into four main categories: 1) schemes offering additional security attributes, such as non-interactive public accountability [11] and invisibility [12, 13]; 2) schemes supporting more fine-grained sanitization [14–16]; 3) schemes restricting sanitizers to use only values selected by the signer [17, 18]; 4) schemes allowing sanitization of encrypted data [19, 20]. Considering that trapdoor-based sanitizable signatures often necessitate interaction between the sanitizer and the signer to obtain trapdoor information after signature generation [14, 15], there was a pressing need for new solutions. In response, Samelin and Slamanig [16] proposed the first policy-based sanitizable signature scheme (P3S), which assigns sanitizable rights to any sanitizer that satisfies a predefined access policy. After that, Afia and AlTawy [21] present an Unlinkable Policy-based Signature Scheme (UP3S), which ensures that the generated sanitized versions of original document are unlinkable where it is infeasible to associate them with the same original one. Aside from the above categories, some studies have explored scenarios with strong unforgeability of signature [22] and conducted some generalization efforts, such as integrating the functionalities of sanitizeable signature and redactable signature [23, 24].

B. Multi-Signature

Multi-Signature (MS) [25] allows a group of signers (each possessing their own key pair) to run an interaction protocol and produce a single signature on the same message. Due to the rise of distributed applications such as Bitcoin and other blockchains, MS has seen a resurgence of interest. It's worth noting that the Bitcoin community has adopted the Schnorr signature [26] proposed in BIP 340 [27], and is seeking a practical MS scheme that is fully compatible with Schnorr signature.

One of the most well-known Schnorr-based MS schemes is proposed by Bellare and Neven [28] (abbreviated as BN scheme), which has been proved to be secure under the plain public key model. In this model, each signer independently generates its own key pair, without any interactive key generation requirements [29] or any knowledge of secret key assumptions [30, 31] employed in previous works.

Following the work of Bellare and Neven [28], the subsequent researches have improved on two aspects, which are summarized in Table I: (1) Key Aggregation. It means to aggregate a set of verification keys into a single, short aggregated key and keep the verification time constant using the fixed-size aggregated public key. By making adjustments to [28], schemes Musig [25], BDN [32], Musig-DN [33] and DWMS [34] add the feature of key aggregation. (2) Two rounds of interaction. Based on [28] that requires three rounds of interactions, several schemes [25, 28, 35, 36] have attempted to reduce the number of interactions to two rounds, however, all of these schemes have been frustrated by Drijvers et al. [37], who pointed that when the adversary is allowed to conduct an arbitrary number of concurrent sessions, none of these above two-round schemes can be proven secure in a pure DL setting (without pairing), and all of them are vulnerable to attacks of sub-exponential complexity. Then, they proposed an improved scheme called mBCJ [37] based on [35]. Although

TABLE I
COMPARISONS OF MULTI-SIGNATURE SCHEMES

Scheme	Type	Key Aggregation	Rounds	concurrent security
BN [28]	Schnorr-based	×	3	✓
BCJ [38]	Schnorr-based	×	2	×
BDN [32]	BLS-based	✓	3	✓
Musig [25]	Schnorr-based	✓	3	✓
mBCJ [37]	Schnorr-based	×	2	✓
Musig-DN [33]	Schnorr-based	✓	2	✓
DWMS [34]	Schnorr-based	✓	2	✓
Musig2 [39]	Schnorr-based	✓	2	✓

mBCJ requires only two rounds, its output is not Schnorr-like and makes it unsuitable as a replacement for Schnorr signature. MuSig-DN also requires only two rounds of interactions, but it relies on heavy zero-knowledge proofs, which greatly increases the complexity of the implementation and makes MuSig-DN actually less efficient than the three-round MuSig. MuSig2 is the first scheme that makes improvements on the above two aspects, with a signer complexity similar to that of ordinary Schnorr signature, and is secure under concurrent signing sessions. It is worth noting that DWMS, a two-round MS scheme derived using linear combinations of multiple nonces, is very similar to MuSig2, but it lacks some of the optimizations present in MuSig2 such as: aggregating the first-round messages from all signers, which saves bandwidth and ensures that each signer performs only a constant number of exponential operations; setting the coefficient of one nonce to constant 1, which saves an exponential operation for each signer when aggregating the nonces; and setting the coefficient of one public key to constant 1, which achieves one exponential optimization when aggregating the public keys.

III. PRELIMINARIES

A. Multi-Signature

Consider a group of signers labeled $1, \cdots, n$ that collaborate to generate a signature for message m, where each of them has his own key pairs as well as the public keys of the others. A Multi-Signature scheme MS consists of four algorithms, namely MS = (*Setup*, *KGen*, *Sign*, *Verify*). The *Setup* algorithm takes 1^λ as input, and outputs the public parameters pp. The *KGen* algorithm is run by each signer independently, which takes pp as input, and outputs their respective key pair (sk, pk). The *Sign* algorithm is an interactive protocol that is run by multiple signers simultanesouly. Suppose L is a multiset of public keys $\{pk_1, \cdots, pk_n\}$. After several rounds of interactions, each signer $i \in L$ can output a compact signature $\hat{\sigma}$. The *Verify* algorithm takes L, a message m and a signature $\hat{\sigma}$ as inputs, and outputs 1 or 0 representing $\hat{\sigma}$ is valid or not. The completeness requires that if $\hat{\sigma}$ is a multi-signature generated by $Sign$ algorithm for message m, then the *Verify* algorithm must be verified as valid under the corresponding public keys and message m.

Definition 1 (MS-EUF-CMA Security). Let MS = (*Setup*, *KGen*, *Sign*, *Verify*) be a multi-signature scheme. Assuming the target honest signer is identified by 1, and $(sk_1, pk_1) \leftarrow KGen(1^\lambda)$ is the secret key pair of target honest signer. Consider the following MS-EUF-CMA game between forger $\mathcal{F}$ and challenger $\mathcal{C}$.

Setup. The forger $\mathcal{F}$ is given public key pk_1 generated by *KGen* and can play the role of signers $(2, \cdots, n)$, in particular it can choose public keys $pk_2, \cdots, pk_n$ arbitrarily.

Queries. The forger $\mathcal{F}$ has access to the signing oracle. It can adaptively request signatures on any messages under any multiset L of $\{pk_1, \cdots, pk_n\}$ including at least one pk_1.

Response. The forger $\mathcal{F}$ outputs a message m^*, a multiset L^* of $\{pk_1, \cdots, pk_n\}$ and a multi-signature $\hat{\sigma}^*$.

The advantage of $\mathcal{F}$ in above game is defined as $Adv_{\mathcal{A}}^{\text{MS-EUF-CMA}}(1^\lambda) = \Pr[Verify(m^*, L^*, \hat{\sigma}^*) = 1]$, where n is polynomial in λ. To make the security definition meaningful, we require that $pk_1 \in L^*$ and $\mathcal{F}$ has never queried (m^*, L^*) to signing oracle. The probability is taken over the random coins used by *KGen* and the above game. A multi-signature scheme is MS-EUF-CMA secure if for any PPT forger $\mathcal{F}$, its advantage is negligible in λ.

B. Chameleon Hash Function

Setup(1^λ): The *Setup* algorithm takes a security parameter 1^λ as input, and outputs public parameter PP_{CH} that determines the hash key space $\mathcal{HK}$, the trapdoor space $\mathcal{TD}$, the input domain $\mathcal{M}$ and the randomness domain $\mathcal{HR}$.

KGen(PP_{CH}): The *KGen* algorithm takes PP_{CH} as input, and outputs a trapdoor key $td \in \mathcal{TD}$ and a hash key $hk \in \mathcal{HK}$.

Hash(hk, m): The *Hash* algorithm takes the hash key hk, a message $m \in \mathcal{M}$ as inputs, and outputs the hash value h and a randomness hr.

Adapt(td, hk, m, hr, m'): The *Adapt* algorithm takes the trapdoor key td, the hash key hk, a message $m \in \mathcal{M}$, a randomness $hr \in \mathcal{HR}$ and a new message $m' \in \mathcal{M}$ as inputs, and outputs a new randomness hr'.

A correctness definition and a formal security definition of chameleon hash are given in [40]. Due to space constraints here we will not repeat them.

IV. SYSTEM MODEL AND SCHEME OVERVIEW

In this section we briefly describe the system model and design ideas of sanitizable and accountable endorsement scheme.

A. System Model

Our proposed sanitizable and accountable endorsement scheme is shown in Fig. 1. This model mainly involves two types of participants: the client and the endorsers. Their roles are defined as follows.

Client: The client is responsible for generating a transaction proposal, which is then sent to specified endorsers according to a predetermined endorsement policy. Upon receiving the endorsement result, the client verifies its compliance with the endorsement policy.

Endorser: The endorsers are a group of privileged nodes that simulate the execution of a transaction and collaboratively generate a single endorsement result. A key distinction of our endorsement scheme is that it produces a single, consolidated

endorsement rather than multiple individual endorsements, thereby maintaining a constant endorsement size regardless of the number of endorsers.

When the transaction proposal requires dynamic updates, the client can individually sanitize the transaction content and adjust the endorsement to ensure its continued validity for the updated transaction. Then, it forward the updated proposal along with the endorsement result to the orderer. To ensure accountability and prevent arbitrary modifications by the client, our endorsement scheme includes mechanisms to track the origin of the updated transaction.

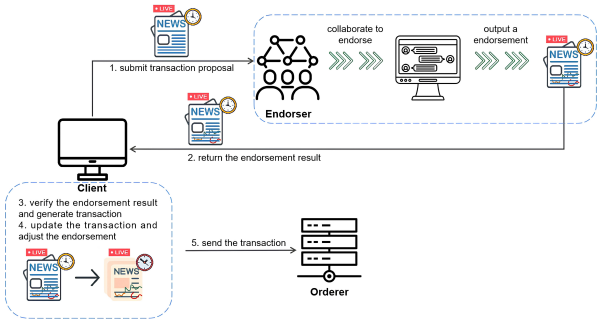


Fig. 1. The Framework of Sanitizable and Accountable Endorsement Scheme

B. Scheme Overview

The proposed scheme is designed to effectively facilitate the endorsement of transactions that require dynamic updates, without compromising the authenticity and usability of on-chain transactions. To achieve this, we introduce a sanitizable multi-signature (SMS) as the technical foundation, which incorporates an accountability mechanism to prevent users—whether signers or sanitizers—from posting false transactions and falsely accusing one another.

The SMS scheme involves three key roles: signer, sanitizer, and verifier. To illustrate the execution process of SMS, we draw an analogy with real-time news reporting, as depicted in Fig. 2.

- **Signer:** The role of the signer is performed by multiple participants who interactively sign the same message and output a compact multi-signature. This multi-signature acts as a collective endorsement from all signers, ensuring that the transaction is authenticated by multiple parties.
- **Sanitizer:** The sanitizer is responsible for updating the message content when changes are necessary, such as in the case of breaking news or evolving events. Upon modifying the message, the sanitizer also updates the multi-signature to maintain its validity for the revised message. This step is crucial in ensuring that the transaction remains current and accurate without losing the initial endorsements.
- **Verifier:** The verifier checks the validity of the multi-signature. In the event of a dispute, where a signer or another party questions the legitimacy of the sanitization, the verifier examines the changes made by the sanitizer.

Constructing a General SMS: We present a general method for constructing Sanitizable Multi-Signatures (SMS)

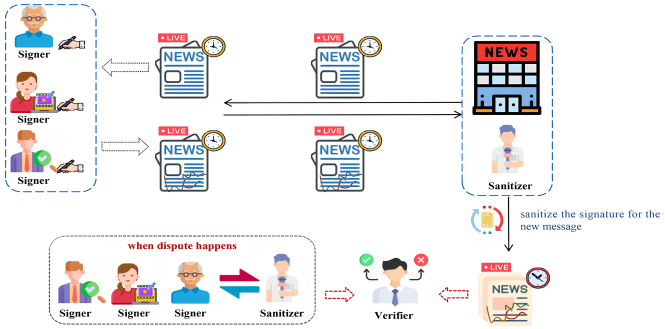


Fig. 2. Sanitizable Multi-Signature (SMS)

based on the combination of a Multi-Signature (MS) and a Chameleon Hash (CH). Typically, the MS algorithm takes a message m as input. In our approach, we compute the CH value h of a message m using a random number hr and adjust the MS algorithm's input to h. Consequently, the MS algorithm outputs a signature σ on h rather than m. This transformation allows the sanitizer, holding the trapdoor key, to compute a collision hr' given the message-randomness pair (m, hr) and a new message m', ensuring that the signature σ remains valid for (m', hr'). This method forms the foundational construction of SMS.

Regulating the Sanitizer: As highlighted, the ability of the sanitizer to derive a valid message-signature pair by finding a hash collision introduces potential risks of privilege abuse. To mitigate this risk, we introduce the principle of *accountability*. Specifically, in addition to signing the CH value, both signers and sanitizers sign the randomness used to compute h, thereby ensuring accountability.

To maintain a constant signature size, we employ an aggregation method where an aggregator among the signers selects the randomness used in computing h and broadcasts it to the co-signers. This allows multiple signatures on the same randomness to be aggregated into a single signature, maintaining efficiency while ensuring accountability.

V. Sanitizable Multi-Signature

We give a formal definition and outline the security properties of Sanitizable Multi-Signature (SMS) primitive. Specifically, a SMS scheme is a secure signature scheme designed to enable a semi-honest sanitizer to modify a signed message without needing interaction with co-signers, while ensuring that the original signature remains equally valid for the revised message. In addition, SMS incorporates accountability mechanisms to deter malicious behavior, such as posting false transactions and making false accusations among users (co-signers or sanitizer).

A. Definition

A Sanitizable Multi-Signature (SMS) scheme with public accountability for a message space $\mathcal{M}$ consists of the following algorithms.

$pp \leftarrow Setup(1^\lambda)$: It takes a security parameter λ as input, and outputs the public parameter pp. Assume that the public parameter pp is an implicit input to all other algorithms.

$(sk, pk) \leftarrow KGen(pp)$: It takes the public parameter pp as input, and generates a key pair (sk, pk), where the private key sk is kept in secret and the public key pk is available to all users in the system.

$(td, hk) \leftarrow HKGen(pp)$: It takes the public parameter pp as input, and generates a trapdoor key pair (td, hk), where the trapdoor td is kept in secret and the hash key hk is public to all users.

$(hr, \hat{\sigma}) \leftarrow Sign(sk_i, hk, m, L)$: The $Sign$ algorithm is a protocol that is executed collaboratively by all signers. Assuming that there are n signers in the system, in terms of signer i ($i \in \{1, \cdots, n\}$), it takes as inputs the private key sk_i, the hash key hk, a message m to be signed and a multiset L of public keys $\{pk_1, \cdots, pk_n\}$. After a few rounds of interactions, each signer i corresponding to L will generate a signature σ_i and obtain the signatures of the remaining co-signers $j (j \in L \backslash \{i\})$. The algorithm finally outputs a randomness hr and a multi-signature $\hat{\sigma}$ with the same size as a normal signature σ_i.

$(hr', \sigma') \leftarrow Sanitize(sk_s, td, hk, m, hr, \hat{\sigma}, m')$. It takes the sanitizer's signing key sk_s, the trapdoor key pair (td, hk), a message-signature pair $(m, hr, \hat{\sigma})$ and a new message m' as inputs, and outputs a new randomness hr' and a valid signature σ',which satisfies σ' is a valid signature on (m', hr').

$\{0, 1\} \leftarrow VerifyMS(L, hk, m, hr, \hat{\sigma})$. Given a multiset L of public keys $\{pk_1, \cdots, pk_n\}$, the hash key hk, a message-randomness pair (m, hr) and a candidate signature $\hat{\sigma}$ as inputs, it outputs 1 if $\hat{\sigma}$ is a valid signature on (m, hr) under L and 0 otherwise.

$\{0, 1\} \leftarrow VerifySS(L, pk_s, m, hr, \hat{\sigma})$. Given a multiset L of public keys $\{pk_1, \cdots, pk_n\}$, the sanitizer's signing key pk_s, a message-randomness pair (m, hr) and a candidate signature $\hat{\sigma}$ as inputs, it outputs 1 if $\hat{\sigma}$ is a valid signature on (m, hr) under L and pk_s, and 0 otherwise.

Correctness. Correctness requires that for all security parameter λ, for all $n \in \mathbb{Z}$, for all $m \in \mathcal{M}$, for all $pp \leftarrow Setup(1^\lambda)$, for all $(sk_i, pk_i) \leftarrow KGen(pp)$ and for all $(td, hk) \leftarrow HKGen(pp)$, it holds that $VerifyMS(L, hk, m, hr, \hat{\sigma}) = 1$ with probability 1, where $(hr, \hat{\sigma}) \leftarrow Sign(sk_i, hk, m, L)$ for $i \in \{1, \cdots, n\}$. We also require that for all $m' \in \mathcal{M}$, for all $(sk_s, pk_s) \leftarrow KGen(pp)$ and for all $(hr', \sigma') \leftarrow Sanitize(sk_s, td, hk, m, hr, \hat{\sigma}, m')$, we have that $VerifySS(L, pk_s, hk, m', hr', \sigma') = 1$.

B. Security Model

By constructing security experiments $\text{Exp}^{\text{EUF-CMA}}_{\mathcal{A},\text{SMS}}(\lambda)$ and $\text{Exp}^{\text{ACT}}_{\mathcal{A},\text{SMS}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, we give the required security definitions for SMS, including unforgeability and accountability, respectively. The oracles used in the security experiments are defined in Table II.

- Unforgeability. Unforgeability requires that an adversary cannot forge a valid multi-signature involving at least one honest signer or act as a sanitizer to forge a valid signature for a sanitized message. Without loss of generality, we assume there exists an honest signer identified by 1 and the adversary can corrupt all other co-signers (namely, choosing corrupted public keys arbitrarily or even as a function of the honest signer's public key). In this way, the adversary can compute multi-signatures indirectly by accessing the honest signer's signing oracle. The unforgeability experiment $\text{Exp}^{\text{EUF-CMA}}_{\mathcal{A},\text{SMS}}(\lambda)$ is detailed in Table III. The challenger $\mathcal{C}$ generates key pairs (sk_1, pk_1) for the honest signer and (sk_s, pk_s) for the sanitizer, and provides $\mathcal{A}$ with pk_1 and pk_s. The security experiment allows $\mathcal{A}$ to access oracles $\mathcal{O}_{Sign}$ and $\mathcal{O}_{Sanitize}$, simulating the signing process of the honest signer and the sanitizing process of the sanitizer, respectively.
To win the game, the adversary must output a message-randomness pair (m^*, hr^*), a multiset L^* of public keys $\{pk_1, \cdots, pk_n\}$ and a signature σ^*. The conditions for the adversary's success are:
1) Multi-Signature Fogery: If $\mathcal{A}$ returns a multi-signature, it wins if $pk_1 \in L^*$, $VerifyMS(L^*, hk, m^*, hr^*, \sigma^*) = 1$ and it has never accessed $\mathcal{O}_{Sign}$ with (m^*, hr^*, L^*).
2) Sanitized Signature Fogery: If $\mathcal{A}$ returns a sanitized signature, it wins if $VerifySS(L^*, pk_s, hk, m^*, hr^*, \sigma^*) = 1$ and it has never accessed $\mathcal{O}_{Sanitize}$ with (m^*, hr^*) as the sanitized message-randomness pair.
Definition 2. A Sanitizable Multi-Signature (SMS) scheme is EUF-CMA secure if for any PPT adversary $\mathcal{A}$,

$$Adv^{\text{EUF-CMA}}_{\mathcal{A},\text{SMS}}(\lambda) = \Pr[\text{Exp}^{\text{EUF-CMA}}_{\mathcal{A},\text{SMS}}(\lambda) = 1] = negl(\lambda).$$

- Accountability. The accountability experiment $\text{Exp}^{\text{ACT}}_{\mathcal{A},\text{SMS}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ is defined in Table III, where the adversary owns private key sk_s (or sk_1) and can access to $\mathcal{O}_{Sign}$ (or $\mathcal{O}_{Sanitize}$). We can see accountability grants the adversary greater capabilities, as it ensures that even with access to one party's private key, the adversary cannot falsely accuse another party. Consequently, our definition of the accountability security experiment encompasses the requirements of the unforgeability experiment, making the latter a special case of the former. By defining the accountability security experiment, we implicitly address the unforgeability requirements.
Definition 3. A Sanitizable Multi-Signature (SMS) scheme is accountable if for any PPT adversary $\mathcal{A}$,

$$Adv^{\text{ACT}}_{\mathcal{A},\text{SMS}}(\lambda) = \Pr[\text{Exp}^{\text{ACT}}_{\mathcal{A},\text{SMS}}(\lambda) = 1] = negl(\lambda).$$

TABLE II
SMS SECURITY ORACLES

$Q_1, Q_2 := \perp$
Oracle $Sign(sk_1, \cdot, \cdot, \cdot)$ on input hk, m and L:
  if $pk_1 \notin L$, **return** $\perp$
  $(hr, \sigma) \leftarrow Sign(sk_1, hk, m, L)$
  $Q_1 := Q_1 \cup \{(m, hr, L)\}$
  return σ
Oracle $Sanitize(sk_s, td, hk, \cdot, \cdot)$ on input (m, hr, L, σ), m':
  if $VerifyMS(L, hk, m, hr, \sigma) = 0$, **return** $\perp$
  $(hr', \sigma') \leftarrow Sanitize(sk_s, td, hk, (m, hr, \sigma), m')$
  $Q_2 := Q_2 \cup \{(m', hr')\}$
  return (hr', σ')

TABLE III
SMS UNFORGEABILITY AND ACCOUNTABILITY EXPERIMENTS

Experiment $\text{Exp}^{\text{EUF,ACT}}_{\mathcal{A},\text{SMS}}(\lambda)$
$pp \leftarrow Setup(\lambda)$
$\{(sk_1, pk_1), (sk_s, pk_s)\} \leftarrow KGen(pp)$
$(td, hk) \leftarrow Gen(pp)$
$Q_1, Q_2 := \emptyset$
$\mathcal{A}$ chooses a random bit $b \in \{0, 1\}$
if $b == 0$,
$(m^*, hr^*, L^*, \sigma^*) \leftarrow \mathcal{A}^{Sign}(pk_1, hk, pk_s, sk_s, td)$
return $pk_1 \in L^* \wedge (m^*, hr^*, L^*) \notin Q_1 \wedge VerifyMS(L^*, hk, m^*, hr^*, \sigma^*)$
elseif $b == 1$,
$(m^*, hr^*, L^*, \sigma^*) \leftarrow \mathcal{A}^{Sanitize}(pk_1, hk, pk_s, sk_1)$
return $(m^*, hr^*) \notin Q_2 \wedge VerifySS(L^*, pk_s, hk, m^*, hr^*, \sigma^*)$

C. General Construction

The proposed general construction consists of two building blocks, an EUF-CMA secure multi-signature scheme $\Sigma = (Setup, KGen, Sign, Verify)$ and a collision-resistant chameleon hash function $CH = (Setup, KGen, Hash, Adapt)$. Define the sanitizable multi-signature scheme SMS as follows.

1) $pp \leftarrow \text{SMS}.Setup(1^\lambda)$. The *Setup* algorithm takes 1^λ as input, and outputs $pp := \{pp_\Sigma, pp_{\text{CH}}\}$, where $pp_\Sigma \leftarrow \Sigma.Setup(1^\lambda)$ and $pp_{\text{CH}} \leftarrow \text{CH}.Setup(1^\lambda)$.
2) $(sk, pk) \leftarrow \text{SMS}.KGen(pp)$. Parsing pp as $(pp_\Sigma, pp_{\text{CH}})$, each signer $i \in \{1, \cdots, n\}$ and the sanitizer s use pp_Σ to generate their respective signing key pairs (sk_i, pk_i) and (sk_s, pk_s). Specifically, $(sk_i, pk_i) \leftarrow \Sigma.KGen(pp_\Sigma)$ for each signer and $(sk_s, pk_s) \leftarrow \Sigma.KGen(pp_\Sigma)$ for the sanitizer. All participants in the system expose their public key and keep their private key secret.
3) $(td, hk) \leftarrow \text{SMS}.HKGen(pp)$. Parsing pp as $(pp_\Sigma, pp_{\text{CH}})$, the sanitizer takes pp_{CH} as inputs and outputs $(td, hk) \leftarrow \text{CH}.KGen(pp_{\text{CH}})$, where td is a trapdoor key and hk is a hash key.
4) $(hr, \hat{\sigma}) \leftarrow \text{SMS}.Sign(sk_i, hk, m, L)$. The *Sign* algorithm is an interactive protocol, which is described here in terms of signer i.
   a) Given a message m to be signed and a randomness hr, compute $h \leftarrow \text{CH}.hash(hk, m, hr)$.
   b) Let L be a multiset of public keys $\{pk_1, \cdots, pk_n\}$, compute $\hat{\sigma}_h \leftarrow \Sigma.Sign(sk_i, h, L)$, $\hat{\sigma}_{hr} \leftarrow \Sigma.Sign(sk_i, hr||m, L)$, and output the randomness hr and multi-signature $\hat{\sigma} := (\hat{\sigma}_h, \hat{\sigma}_{hr})$.
5) $(0, 1) \leftarrow \text{SMS}.VerifyMS(L, hk, m, hr, \hat{\sigma})$. Given a multiset L of public keys $\{pk_1, \cdots, pk_n\}$, a hash key hk and a message-signature pair $(m, hr, \hat{\sigma})$, the verifier parses $\hat{\sigma}$ as $(\hat{\sigma}_h, \hat{\sigma}_{hr})$, computes $h \leftarrow \text{CH}.hash(hk, m, hr)$ and returns $\Sigma.Verify(L, h, \hat{\sigma}_h) \wedge \Sigma.Verify(L, hr||m, \hat{\sigma}_{hr})$.
6) $(hr', \hat{\sigma}') \leftarrow \text{SMS}.Sanitize(sk_s, td, hk, m, hr, \hat{\sigma}, m')$. Given the signing key sk_s of sanitizer s, the hash key pair (td, hk), a message-signature pair $(m, hr, \hat{\sigma})$ and a new message m', the sanitizer parses $\hat{\sigma}$ as $(\hat{\sigma}_h, \hat{\sigma}_{hr})$, computes $hr' \leftarrow \text{CH}.Adapt(td, hk, m, hr, m')$ and $\sigma'_{hr} \leftarrow \Sigma.Sign(sk_s, hr'||m', pk_s)$, and returns $\hat{\sigma}' := (\hat{\sigma}_h, \sigma'_{hr})$.
7) $(0, 1) \leftarrow \text{SMS}.VerifySS(L, pk_s, hk, m, hr, \hat{\sigma})$. Given a multiset L of public keys $\{pk_1, \cdots, pk_n\}$, the sanitizer's public key pk_s, the hash key hk and a message-signature pair $(m, hr, \hat{\sigma})$, the verifier parses $\hat{\sigma}$ as $(\hat{\sigma}_h, \hat{\sigma}_{hr})$, computes $h \leftarrow \text{CH}.hash(hk, m, hr)$ and returns $\Sigma.Verify(L, h, \hat{\sigma}_h) \wedge \Sigma.Verify(pk_s, hr||m, \hat{\sigma}_{hr})$.

Note: In designing the accountability mechanism, we opted to sign $hr||m$ rather than only hr to enhance system security against impersonation attacks. Signing hr solely would allow a malicious adversary to use the output hr from the *Sanitize* algorithm as input to the *Sign* algorithm and generate valid signatures σ_h and σ_{hr} under L and pk_s. This could mislead the verification process and compromise data integrity and accountability.

By signing $hr||m$, the signature is tethered to both the randomness hr and the specific data m. Even if an adversary obtains hr output from the *Sanitize* algorithm, it cannot forge a valid signature without m.

D. Security Proof

Here we will prove the unforgeability and accountability of the proposed SMS scheme. Due to space constraints, the detailed security proofs are submitted as supplementary material.

Theorem 1. *If MS is EUF-CMA secure and CH is CR secure, then the SMS scheme is EUF-CMA secure.*

Specifically, for any PPT adversary $\mathcal{A}$ with advantage $Adv^{\text{EUF-CMA}}_{\mathcal{A},\text{SMS}}$, there exists PPT adversaries $\mathcal{B}_{\text{MS}}$ and $\mathcal{B}_{\text{CH}}$, such that $Adv^{\text{EUF-CMA}}_{\mathcal{A},\text{SMS}}(1^\lambda) \leq Adv^{\text{EUF-CMA}}_{\mathcal{B}_{\text{MS}}}(1^\lambda) + Adv^{\text{CR}}_{\mathcal{B}_{\text{CH}}}(1^\lambda)$, where $Adv^{\text{EUF-CMA}}_{\mathcal{B}_{\text{MS}}}(1^\lambda)$ represents the advantage of adversary $\mathcal{B}_{\text{MS}}$ against the underlying MS scheme and $Adv^{\text{CR}}_{\mathcal{B}_{\text{CH}}}(1^\lambda)$ represents the advantage of adversary $\mathcal{B}_{\text{CH}}$ against the underlying CH scheme.

Theorem 2. *The SMS scheme achieves accountability if the underlying signature scheme is EUF-CMA secure and the underlying chameleon hash scheme is CR secure.*

Specifically, assuming that $Adv_{\mathcal{B}_s}(1^\lambda)$ denotes the advantage of $\mathcal{B}_s$ in finding a CH collision together with forging a valid signature, and $Adv_{\mathcal{B}_1}(1^\lambda)$ denotes the advantage of $\mathcal{B}_1$ in finding a CH collision together with forging a valid multi-signature, the advantage $Adv^{\text{EUF-CMA}}_{\mathcal{A}}(1^\lambda)$ of $\mathcal{A}$ in breaking our instantiated SMS scheme satisfied $Adv^{\text{EUF-CMA}}_{\mathcal{A}}(1^\lambda) \leq Adv_{\mathcal{B}_s}(1^\lambda) + Adv_{\mathcal{B}_1}(1^\lambda)$.

E. Instantiation

In this section, we present the proposed instantiation, along with implementation and evaluation analysis. For simplicity, we assume an aggregator among the signers is responsible for receiving external messages to be authenticated, verifying and aggregating outputs from other signers, and sending the multi-signature to outsides. The instantiation of SMS based on Musig2[39] and CH [40] is as follows.

- $Setup(1^\lambda) \rightarrow pp$. On input of a security parameter λ, choose a group $\mathbb{G}$ of order q with generator g, where q is a κ-bit prime. Choose hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}^*_q$ and $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}$, and output the public parameter $pp := \{\mathbb{G}, q, g, H_1, H_2\}$.

- $KGen(pp) \rightarrow (sk, pk)$. On input of the public parameter pp, each signer $i \in \{1, \cdots, n\}$ sets the signing key $sk_i := x_i$ with $x_i \xleftarrow{\$} \mathbb{Z}_q^*$, computes $X_i := g^{x_i}$, and publishes the public key $PK_i = X_i$. Accordingly, the sanitizer s computes its signing key pair as (x_s, X_s).
- $HKGen(pp) \rightarrow (td, hk)$. On input of the public parameter pp, the sanitizer sets the trapdoor key $td := x_t$ with $x_t \xleftarrow{\$} \mathbb{Z}_q^*$, computes the hash key $hk := g^{x_t}$ and publishes pk_s and hk.
  In Musig2 [39], there exists a *KeyAgg* algorithm, which takes a multiset L of public keys $\{pk_1, \cdots, pk_n\} := \{X_1, \cdots, X_n\}$ as inputs, computes and outputs $X := \prod_{i=1}^{n} X_i^{a_i}$ where $a_i = H_1(L, X_i)$ for $i \in \{1, \cdots, n\}$.
- $Sign(sk_i, hk, m, L) \rightarrow (hr, \hat{\sigma})$. Each signer $i \in \{1, \cdots, n\}$ chooses $r_{i,1}, r_{i,2}, d_{i,1}, d_{i,2} \xleftarrow{\$} \mathbb{Z}_q^*$, computes nounces $R_{i,1} = g^{r_{i,1}}$, $R_{i,2} = g^{r_{i,2}}$, $D_{i,1} = g^{d_{i,1}}$, $D_{i,2} = g^{d_{i,2}}$ and sends these nonces to the aggregator. The aggregator then computes $R_1 := \prod_{i=1}^{n} R_{i,1}$, $R_2 := \prod_{i=1}^{n} R_{i,2}$, $D_1 := \prod_{i=1}^{n} D_{i,1}$, $D_2 := \prod_{i=1}^{n} D_{i,2}$ and outputs (R_1, R_2, D_1, D_2).
  When receiving a message m to be signed, the aggregator computes $hr := (g^{\alpha}, hk^{\alpha})$ with $\alpha \xleftarrow{\$} \mathbb{Z}_q^*$, and broadcasts (m, hr) to all signers.
  Once receiving (R_1, R_2, D_1, D_2), each signer *i* computes $b_1 := H_1(X, R_1, R_2, m)$ and $b_2 := H_1(X, D_1, D_2, hr)$ using the aggregated public key X from the *KeyAgg* algorithm, then computes $h := g^{\alpha} \cdot H_2(hk)^{H_1(m)}$, $R := \prod_{j=1}^{2} R_j^{b_1^{j-1}} = R_1 \cdot R_2^{b_1}$, $D := \prod_{j=1}^{2} D_j^{b_2^{j-1}} = D_1 \cdot D_2^{b_2}$, $c_1 := H_1(X, R, h)$, $c_2 := H_1(X, D, hr||m)$, $s_i := (r_{i,1} + r_{i,2} b_1 + c_1 a_i x_i) \mod q$, $p_i := (d_{i,1} + d_{i,2} b_2 + c_2 a_i x_i) \mod q$ where $a_i := H_1(L, X_i)$, and sends $\sigma_i := (s_i, p_i)$ to the aggregator.
  The aggregator parses σ_i as (s_i, p_i) for $i \in \{1, \cdots, n\}$, computes $\hat{s} := \sum_{i=1}^{n} s_i$ and $\hat{p} := \sum_{i=1}^{n} p_i$, and then outputs the randomness hr and the multi-signature $\hat{\sigma} := (\hat{\sigma_1}, \hat{\sigma_2}) := ((R, \hat{s}), (D, \hat{p}))$.
- $VerifyMS(X, hk, m, hr, \hat{\sigma}) \rightarrow (0, 1)$. On input of the aggregated public key X, hash key hk, message tuple (m, hr) where $hr = (g^{\alpha}, hk^{\alpha})$ and aggregated signature $\hat{\sigma} = (\hat{\sigma_1}, \hat{\sigma_2})$, the verifier parses $(\hat{\sigma_1}, \hat{\sigma_2})$ as $((R, \hat{s}), (D, \hat{p}))$, computes $h := g^{\alpha} \cdot H_2(hk)^{H_1(m)}$, $c_1 := H_1(X, R, h)$, $c_2 := H_1(X, D, hr||m)$, and accepts the aggregated signature if $g^{\hat{s}} = R \cdot X^{c_1}$ and $g^{\hat{p}} = D \cdot X^{c_2}$.
- $Sanitize(sk_s, td, hk, m, hr, \hat{\sigma}, m') \rightarrow (hr', \hat{\sigma}')$. On input of the sanitizer's signing key sk_s, trapdoor key pair (td, hk), a message-signature pair $(m, hr, \hat{\sigma})$ and a new message m', the sanitizer computes a new randomness hr'. Here, $hr = (g^{\alpha}, hk^{\alpha})$, $hr' = (g^{\alpha'}, hk^{\alpha'})$ where $g^{\alpha'} := g^{\alpha} \cdot H_3(hk)^{H_1(m) - H_1(m')}$ and $hk^{\alpha'} := hk^{\alpha} \cdot H_2(hk)^{x(H_1(m) - H_1(m'))}$. Then, it chooses $d_s \xleftarrow{\$} \mathbb{Z}_q^*$, computes a new signature $\hat{p} := d_s + c_2 x_s$ for hr' with $D_s := g^{d_s}$ and $c_2 := H_1(X_s, D_s, hr'||m')$. Finally, it adapts the signature for m' to $\hat{\sigma}' = (\sigma_1, \sigma_2) = ((R, \hat{s}), (D_s, \hat{p}))$, where σ_1 is a valid aggregated signature for m, which is also valid for m' due to the chameleon hash.
- $VerifySS(X, pk_s, hk, m, hr, \hat{\sigma}) \rightarrow (0,1)$. Given the aggregated public key X, the sanitizer's public key $pk_s := X_s$, the hash key hk and a message-signature pair $(m, hr, \hat{\sigma})$ with $hr := (g^{\alpha}, hk^{\alpha})$ and $\hat{\sigma} = (\hat{\sigma_1}, \hat{\sigma_2}) = ((R, \hat{s}), (D, \hat{p}))$, the verifier computes $h := g^{\alpha} \cdot H_2(hk)^{H_1(m)}$, if $g^{\hat{s}} = R \cdot X^{H_1(X,R,h)}$ and $g^{\hat{p}} = D X_s^{H_1(X_s, D, hr||m)}$, the verifier outputs 1 indicating that $\hat{\sigma}$ is the signature of the new message m' sanitized by sanitizer s. Otherwise, it outputs 0.

VI. Applying Sanitizable and Accountable Endorsement to Fabric

A. The Current Transaction Flow in Fabric

The transaction execution process in Fabric generally follows six steps, as shown in Fig. 3: (1) the client builds a transaction proposal and signs it, then sends the transaction request to the endorsers specified in the endorsement policy. (2) The endorsers validate the signature, simulate the proposed transaction without committing it to the ledger, and return the proposal response to the client. (3) The client checks whether the responses are consistent and satisfy the endorsement policy. If the required number of matching endorsements is obtained, the transaction, along with its endorsements, are submitted to the orderer. (4) The orderer sorts the transactions, encapsulates them into a block and broadcasts the block to all the committers in the channel. (5) The committers verify the transactions within the block. Upon validation, the block is appended to the blockchain. (6) The peer nodes write each valid transaction to their respective ledger and update the world state.

The endorsement process primarily refers to steps 2, 3 and 4 of the transaction execution flow. In handling dynamic transactions, the existing endorsement process often encounters performance bottlenecks due to multi-node signature generation. Furthermore, the failure of critical endorsers can degrade system robustness and availability. We apply our proposed SMS to construct a sanitizable and accountable endorsement scheme, aiming to optimize the existing endorsement process and enhance the efficiency of dynamic transaction processing.

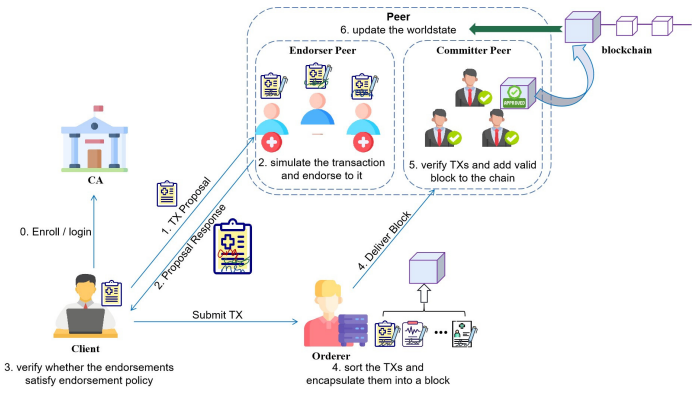


Fig. 3. Hyperledger Fabric Workflow with Our Endorsement Scheme

B. Sanitizable and Accountable Endorsement for Fabric

Here, we describe how to apply SMS to construct a sanitizable and accountable endorsement scheme for dynamic transaction updates. By integrating the concepts shown in Fig.1 and Fig.2, the client first sends a real-time transaction proposal to the endorsers for authenticity verification. A committee of endorsers assesses the proposal based on the given endorsement policy. If the proposal is supported, the endorsers collaboratively execute the *SMS.Sign* algorithm to produce an endorsement with a multi-signature, which is then sent to the client. In this way, the endorsement size can be reduced to a constant level.

Upon receiving the endorsement from the endorsers, the client verifies whether the multi-signature rather than multiple signatures satisfies the endorsement policy and originates from a sufficient number of endorsers. The client then updates the proposal dynamically and executes the *SMS.Sanitize* algorithm, so as to enable the endorsement remains valid for the updated transaction. The updated transaction proposal and the aggregated endorsement are then packaged into a single transaction, which is signed and sent to the orderer. This reduces the client's verification overhead to a constant level and mitigates the transaction size, thereby improving block utilization. Importantly, our enhanced endorsement process ensures both the authentication of dynamically updated transactions by the endorsers and the preservation of update integrity by the client, while supporting tracking to prevent the shirking of endorsement responsibilities. In summary, applying our sanitizable and accountable endorsement scheme to Fabric's transaction execution process can achieve performance gains and functional optimization.

VII. Performance Evaluation and Implementation

A. Performance Evaluation

Our instantiation uses [39] and [40] as building blocks, which is the best combination for efficiency and security reasons. We first analyze the performance of various MS schemes in theory. Specifically, we compare the MS schemes in terms of computational complexity, storage overhead, interaction rounds, and communication complexity. The comparison results are shown in Table IV. In particular, the computational complexity mainly focuses on expensive operations such as exponentiation, multiplication and inverse operations. From Table IV, we can see that only scheme [28] is a three-round scheme, which does not support key aggregation and thus implicitly leads to a linear correlation between its verification time and the number of signers. Among the remaining two-round MS schemes, the key aggregation algorithm of mBCJ [37] has double the exponential operations compared to other two-round schemes, and the signing overhead of DWMS (m=2) [41] is linearly related to the number of signers, which is significantly larger than the other schemes. In addition to the comparable constant number of exponential operations with mBCJ [37] and Musig2 [39], the signing algorithm of Musig-DN [33] additionally requires expensive zero-knowledge proofs, which greatly increases the implementation complexity. In summary, Musig2 [39] outperforms the other schemes in terms of the overall performance, especially in terms of interaction rounds and computational complexity in signing and verification.

Then, we analyze the performance of typical CH functions in theory. From Table V, we know that $\mathrm{CH}_{\mathrm{dl}}$ and $\mathrm{CH}_{\mathrm{fac}}$ suffer from key leakage. That is, an attacker can compute the trapdoor key based on the information obtained and further find the collision of arbitrary messages, and thus making them impractical. $\mathrm{CH}_{\mathrm{rsa}}$ is resistant to key leakage but is based on the RSA assumption, which makes it significantly less efficient in computing CH values and finding collisions than [40]. This efficiency consideration explains why [40] was chosen in our instantiation scheme. To provide a more intuitive view of the performance of our instantiation, we additionally summarize its theoretical performance in Table VI.

B. Implementation

We enhance the original MuSig2 implementation[1] by integrating a chameleon hash function, thereby implementing our SMS scheme in Python. Performance is evaluated on a XiaoxinAir 14+ with a 2.3GHz AMD R5-5600U CPU and 16GB RAM. We choose the secp256k1 curve, known for its 256-bit security and widespread use in blockchain systems like Bitcoin and Ethereum, as the cyclic group and perform variety of operations on it. We instantiate the general hash function in chameleon hash with Sha256. Table VII summarizes the average running time and key communication overheads of our instantiation when using secp256k1, where the public key and the aggregated public key are in compressed form (i.e., a 1-byte prefix and a 32-byte x-coordinate). Note that the time costs for *KeyAgg* and *AggSign* algorithms in Table VII reflect scenarios with 3 co-signers.

Next, we evaluate the impact of the co-signer set on the overall scheme. In our instantiation scheme, the computational overhead of *KeyGen*, *Sign*, *Verify*, and *Sanitize* algorithms remains independent of the co-signer set. Conversely, the *KeyAgg* and *AggSign* algorithms exhibit linear complexity relative to the size of the co-signer set. Evaluation results, detailed in Table VIII, confirm this behavior. For instance, with $n = 100$, *KeyAgg* takes approximately 0.147s, and *AggSign* about 0.01ms. With $n = 1000$, these times increase to approximately 2.682s and 0.119ms, respectively. Notably, while *KeyAgg* governs the scheme's performance and its impact escalates with larger co-signer sets, it can be precomputed upon receipt of the message and executed only once, remaining unchanged for subsequent identical co-signer multisets. Additionally, though *AggSign*'s time cost scales linearly with the co-signer set, its overall computational impact is negligible.

Finally, we show the impact on Fabric blockchain upon integrating our instantiation. Fig.4 illustrates the general structure of a block in Fabric, comprising a block header, m transaction structures, and Block Metadata. Each transaction structure's payload includes multiple TransactionAction (TA) structures since multiple endorser nodes are required. Each TA includes the endorser's identity (consisting of a certificate and a public key) and its endorsement to the transaction. Our

[1] musig2.py [source code]. https://github.com/meshcollider/musig2-py

TABLE IV
THE PERFORMANCE COMPARISON AMONG VARIOUS MS SCHEMES

	KeyGen	*KeyAgg*	*Sign*	*Verify*	Rounds	pk	X	$\hat{\sigma}$
BN [28]	$1E$	$\perp$	$1E+(n-1)M$	$(n+1)E+nM$	3	$\|\mathbb{G}\|$	$\perp$	$\|\mathbb{G}+\mathbb{Z}_p\|$
mBCJ [37]	$2E$	$2nE+(2n-1)M$	$5E+3(\|C\|+1)M$	$6E+4M$	2	$\|\mathbb{G}\|+2\|\mathbb{Z}_p\|$	$\|\mathbb{G}\|$	$2\|\mathbb{G}\|+3\|\mathbb{Z}_p\|$
Musig-DN [33]	$1E$	$nE+(n-1)M$	$3E+(n-1)M$	$2E+1M$	2	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}+\mathbb{Z}_p\|$
DWMS [41]	$1E$	$nE+(n-1)M$	$2(n+1)E+(2n-1)M$	$2E+1M+1INV$	2	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}+\mathbb{Z}_p\|$
Musig2 [39]	$1E$	$nE+(n-1)M$	$3E+(2n-1)M$	$2E+1M$	2	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}+\mathbb{Z}_p\|$

Denote: E as an exponential operation; M as a multiplication operation; n as the size of signature set; $|\mathbb{G}|$ as the size of an element in group $\mathbb{G}$; $|\mathbb{Z}_p|$ as the size of an element in group $\mathbb{Z}_p$; $|C|$ as the number of children in the current node; INV as an inverse operation.

TABLE V
THE PERFORMANCE COMPARISON AMONG VARIOUS CHAMELEON HASH SCHEMES

	KeyGen	*Hash*	*Adapt*	Assumption	Size of CH value	Key-exposure free
CH_{dl} [42]	$1E$	$2E+1M$	$1M$	DLP	$\|\mathbb{G}\|$	×
CH_{rsa} [43]	$1E$	$2E+1M$	$1E+1M$	RSA	$\|\mathbb{Z}_N^*\|$	✓
CH_{fac} [44]	lM^2	min=$1M^2$, max=$lM+1M^2$	max=lM	FAC	$\|\mathbb{Z}_N^*\|$	×
[40]	$1E$	$3E+1M$	$2E+2M$	CDHP	$\|\mathbb{G}\|$	✓

Denote: l as the binary length of a message; $|\mathbb{Z}_N^*|$ as the size of an element in group $\mathbb{Z}_N^*$; M^2 as a square operation; DLP as the Discrete Log Problem; FAC as the factoring assumption; CDHP as the Computational Diffie-Hellman Problem.

TABLE VI
THE PERFORMANCE FIGURES OF INSTANTIATED SMS SCHEME

Computation Cost								Storage Cost and Communication Cost			
KeyGen		*KeyAgg*	*Sign*		*Verify*	*Sanitize*	*Judge*	pk		X	$\hat{\sigma}$
Signer	sanitizer	$nE+(n-1)M$	Signer	Aggregator	$5E+3M$	$3E+2M$	$2E+1M$	Signer	Sanitizor	$\|\mathbb{G}\|$	$2\|\mathbb{G}\|+2\|\mathbb{Z}_q\|$
$1E$	$2E$		$7E+3M$	$2E+4(n-1)M$				$\|\mathbb{G}\|$	$2\|\mathbb{G}\|$		

TABLE VII
THE RUNNING TIME (MS) AND THE COMMUNICATION COST (B) OF INSTANTIATED SMS SCHEME

KeyGen		*KeyAgg* ($n=3$)	*Sign*		*AggSign* ($n=3$)	*Verify*	*Sanitize*	Communication Cost		
Signer	sanitizer		*CH*	*Sign*				pk	X	$\hat{\sigma}$
0.408	0.803	2.723	4.246	7.091	$1.876*10^{-4}$	3.533	3.891	33	33	130

TABLE VIII
TIME OVERHEAD (MS) OF THE SCHEME WITH DIFFERENT NUMBER OF CO-SIGNERS

Algorithm / Number	*KeyGen*		*KeyAgg*	*Sign*		*AggSign*	*Verify*	*Sanitize*
	Signer	sanitizer		*CH*	*Sign*			
$n=3$	0.408	0.803	2.569	4.246	7.091	$1.876*10^{-4}$	3.533	3.891
$n=10$	0.416	0.808	11.870	4.324	7.181	$1.489*10^{-3}$	3.634	3.958
$n=50$	0.402	0.802	63.262	4.299	7.014	$5.870*10^{-3}$	3.551	3.913
$n=100$	0.406	0.803	126.977	4.317	7.150	$1.145*10^{-2}$	3.619	3.898
$n=1000$	0.402	0.807	1290	4.333	7.105	0.119	3.505	3.938

instantiation can aggregate signatures on the same transaction from multiple endorsers into a single aggregated one so as to compress the block space. In addition, by verifying the validity of the aggregated signature rather than individual signatures from each endorser, it confirms whether a sufficient number of specified endorsers have endorsed the proposal, which in turn reduces the block confirmation time.

To evaluate the performance improvements of integrating our instantiation into the Fabric, we conducted comparative analyses on endorsement sizes and verification time costs with ECDSA (i.e., the current signature algorithm applied in the Fabric). Fig.5 presents results for a single transaction, where with 3 endorsers, our scheme reduces endorsement size by 12.84% compared to ECDSA. This reduction increases to nearly 40% with 1000 endorsers. At the block level, as depicted in Fig.6, with 10 endorsers, storage space decreases by approximately 31% regardless of whether the block contains 1500 or 5000 transactions, and by nearly 38% with 50 endorsers.

The runtime of verifying endorsement information for both a single transaction and a block is presented in Fig.7 and Fig.8, respectively. It is observed that with 3 endorsers, ECDSA verification time is 0.948ms faster than our instantiation. This difference arises because our approach requires aggregating endorsers' public keys before verification, involving computationally expensive exponential operations. However, as the

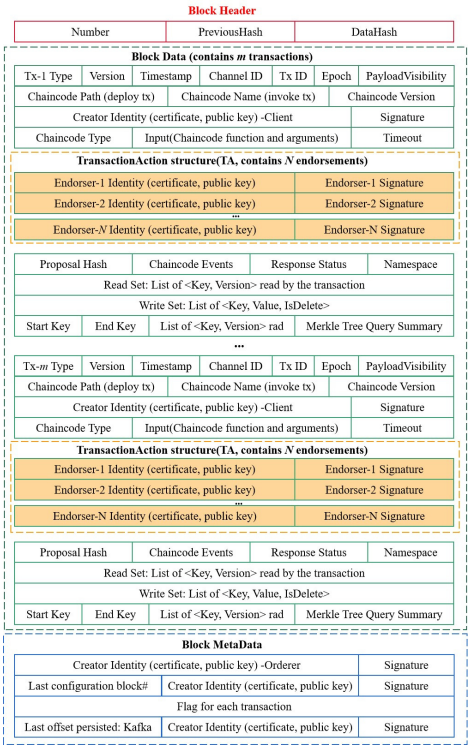


Fig. 4. Block Structure in Hyperledger Fabric

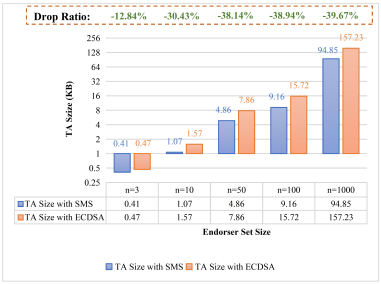
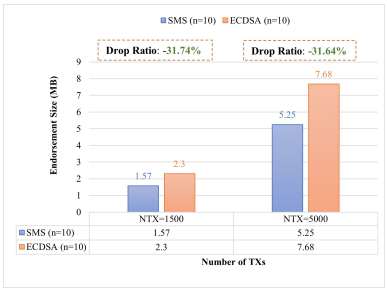


Fig. 5. Total TAs Size (KB) in One Transaction Using Instantiated SMS vs ECDSA

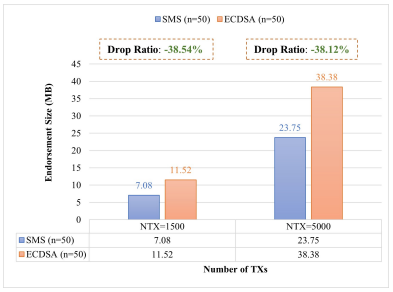
number of endorsers increases, the verification time of our instantiation gradually becomes less than that of ECDSA. As illustrated in Fig.7, when the number of endorsers increases from 10 to 50, our scheme exhibits a verification time reduction ranging from 9.2% to nearly 25%, demonstrating greater efficiency with more endorsers. Furthermore, if the composition of endorsers is predetermined or the endorser set remains constant across multiple transactions, in which case the verification time when using our instantiation will be further shortened.

VIII. CONCLUSION

In this paper, we introduce the concept of sanitizable multi-signature (SMS), detailing its general construction and practical application. SMS enables co-signers to audit data while allowing the sanitizer to update information to maintain the validity of the signature for the updated data without requiring interaction with the co-signers. Leveraging SMS, we design a sanitizable and accountable endorsement scheme that reduces



(a) TAs size in one block when n=10



(b) TAs size in one block when n=50

Fig. 6. Total TAs size in one block using instantiated SMS vs ECDSA

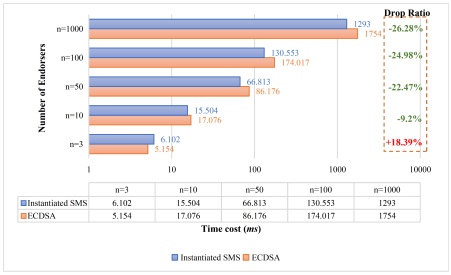
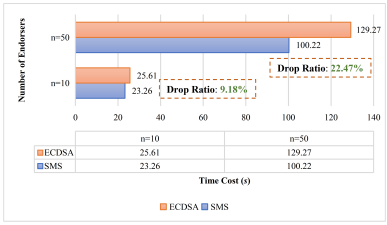
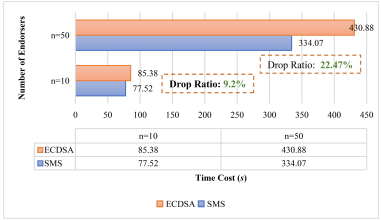


Fig. 7. Total runtime to verify signature sets in one transaction using instantiated SMS vs ECDSA



(a) TAs size in one block when N_{TX}=1500



(b) TAs size in one block when N_{TX}=5000

Fig. 8. Total runtime to verify signature sets in one block using instantiated SMS vs ECDSA

transaction size to a constant level and significantly lowers verification overhead compared to existing methods. By incorporating an accountability mechanism, the proposed scheme can further prevent the abuse of authority and false accusations among participants. The proposed scheme not only meets authenticity and high-timeliness requiurements for dynaimc transactions in Fabric, provides a novel technical solution for processing such transactions, but also enhances transaction processing speeed, thereby improving the overall performance of Fabric. Our security analysis rigorously demonstrates the robustness of the proposed scheme. Performance evaluations indicate that SMS substantially reduces time overhead and transaction size on the Fabric platform compared to the default ECDSA scheme. When verifying multiple endorsements, our scheme decreases storage space by approximately 30%-40% and a verification time reduction ranging from 9.2% to nearly 26.3%, thus improving the overall efficiency and applicability of the Hyperledger Fabric network.

For future research, we plan to integrate SMS with other blockchain platforms to assess its versatility and performance in different environments. Additionally, we aim to optimize the endorsement process by incorporating SMS with other existing endorsement strategies in Fabric. These efforts will further enhance the efficiency and security of the endorsement process, broadening the applicability and impact of our proposed scheme.

REFERENCES

[1] E. Androulaki, A. De Caro, M. Neugschwandtner, and A. Sorniotti, "Endorsement in hyperledger fabric," in *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019, pp. 510–519.

[2] M. ElMessiry and A. ElMessiry, "Blockchain framework for textile supply chain management: Improving transparency, traceability, and quality," in *International conference on blockchain*. Springer, 2018, pp. 213–227.

[3] G. Ateniese, D. H. Chou, B. De Medeiros, and G. Tsudik, "Sanitizable signatures," in *Computer Security–ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005. Proceedings 10*. Springer, 2005, pp. 159–177.

[4] C. Brzuska, M. Fischlin, A. Lehmann, and D. Schröder, "Unlinkability of sanitizable signatures," in *Public Key Cryptography–PKC 2010: 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings 13*. Springer, 2010, pp. 444–461.

[5] K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, S. Tezuka, and H. Imai, "Digitally signed document sanitizing scheme with disclosure condition control," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 88, no. 1, pp. 239–246, 2005.

[6] Y. Xiao, P. Zhang, and Y. Liu, "Secure and efficient multi-signature schemes for fabric: An enterprise blockchain platform," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 1782–1794, 2021. [Online]. Available: https://doi.org/10.1109/TIFS.2020.3042070

[7] C. Li, Y. Wu, and F. Yu, "An improved schnorr-based multi-signature scheme with application to blockchain," in *2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*. IEEE, 2021, pp. 858–863.

[8] P. Zhang, Y. Huang, F. Ge, and Y. Liu, "Group-oriented multi-signature supporting monotonic endorse policies in hyperledger fabric," in *2023 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2023, pp. 256–264.

[9] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, pp. 36–63, 2001.

[10] A. Bilzhause, H. C. Pöhls, and K. Samelin, "Position paper: The past, present, and future of sanitizable and redactable signatures," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 2017, pp. 1–9.

[11] C. Brzuska, H. C. Pöhls, and K. Samelin, "Non-interactive public accountability for sanitizable signatures," in *Public Key Infrastructures, Services and Applications: 9th European Workshop, EuroPKI 2012, Pisa, Italy, September 13-14, 2012, Revised Selected Papers 9*. Springer, 2013, pp. 178–193.

[12] X. Bultel, P. Lafourcade, R. W. Lai, G. Malavolta, D. Schröder, and S. A. K. Thyagarajan, "Efficient invisible and unlinkable sanitizable signatures," in *Public-Key Cryptography–PKC 2019: 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part I 22*. Springer, 2019, pp. 159–189.

[13] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig, "Chameleon-hashes with ephemeral trapdoors: And applications to invisible sanitizable signatures," in *Public-Key Cryptography–PKC 2017: 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II 20*. Springer, 2017, pp. 152–182.

[14] S. Canard, F. Laguillaumie, and M. Milhau, "Trapdoor sanitizable signatures and their application to content protection," in *Applied Cryptography and Network Security: 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings 6*. Springer, 2008, pp. 258–276.

[15] J. Lai, X. Ding, and Y. Wu, "Accountable trapdoor sanitizable signatures," in *Information Security Practice and Experience: 9th International Conference, ISPEC 2013, Lanzhou, China, May 12-14, 2013. Proceedings 9*. Springer, 2013, pp. 117–131.

[16] K. Samelin and D. Slamanig, "Policy-based sanitizable signatures," in *Topics in Cryptology–CT-RSA 2020: The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24–28, 2020, Proceedings*. Springer, 2020, pp. 538–563.

[17] S. Canard and A. Jambert, "On extended sanitizable signature schemes," in *Cryptographers' Track at the RSA Conference*. Springer, 2010, pp. 179–194.

[18] D. Derler and D. Slamanig, "Rethinking privacy for extended sanitizable signatures and a black-box construction of strongly private schemes," in *International Conference on Provable Security*. Springer, 2015, pp. 455–474.

[19] C. Badertscher, C. Matt, and U. Maurer, "Strengthening access control encryption," in *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*. Springer, 2017, pp. 502–532.

[20] I. Damgård, H. Haagh, and C. Orlandi, "Access control encryption: Enforcing information flow with cryptography," in *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part II 14*. Springer, 2016, pp. 547–576.

[21] I. Afia and R. AlTawy, "Unlinkable policy-based sanitizable signatures," in *Cryptographers' Track at the RSA Conference*. Springer, 2023, pp. 191–221.

[22] S. Krenn, K. Samelin, and D. Sommer, "Stronger security for sanitizable signatures," in *International Workshop on Data Privacy Management*. Springer, 2015, pp. 100–117.

[23] S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig, "Protean signature schemes," in *International Conference on Cryptology and Network Security*. Springer, 2018, pp. 256–276.

[24] ——, "Fully invisible protean signatures schemes¡? show [aq id= q1]?¿," *IET Information Security*, vol. 14, no. 3, pp. 266–285, 2020.

[25] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple schnorr multi-signatures with applications to bitcoin," *Designs, Codes and Cryptography*, vol. 87, no. 9, pp. 2139–2164, 2019.

[26] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of cryptology*, vol. 4, pp. 161–174, 1991.

[27] P. Wuille, J. Nick, and T. Ruffing, "Schnorr signatures for secp256k1," *Bitcoin Improvement Proposal*, 2018.

[28] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 390–399.

[29] S. Micali, K. Ohta, and L. Reyzin, "Accountable-subgroup multisignatures," in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, 2001, pp. 245–254.

[30] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *International Workshop on Public Key Cryptography*. Springer, 2002, pp. 31–46.

[31] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters, "Sequential aggregate signatures and multisignatures without random oracles," in *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*. Springer, 2006, pp. 465–485.

[32] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2018, pp. 435–464.

[33] J. Nick, T. Ruffing, Y. Seurin, and P. Wuille, "Musig-dn: Schnorr multi-signatures with verifiably deterministic nonces," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1717–1731.

[34] H. K. Alper and J. Burdges, "Two-round trip schnorr multi-signatures via delinearized witnesses," Cryptology ePrint Archive, Paper 2020/1245, 2020, https://eprint.iacr.org/2020/1245. [Online]. Available: https://eprint.iacr.org/2020/1245

[35] A. Bagherzandi, J. H. Cheon, and S. Jarecki, "Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma," in *ACM conference on Computer and communications security*, 2008.

[36] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford, "Keeping authorities" honest or bust" with decentralized witness cosigning," in *2016 IEEE Symposium on Security and Privacy (SP)*. Ieee, 2016, pp. 526–545.

[37] M. Drijvers, K. Edalatnejad, B. Ford, E. Kiltz, J. Loss, G. Neven, and I. Stepanovs, "On the security of two-round multi-signatures," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1084–1101.

[38] A. Bagherzandi, J.-H. Cheon, and S. Jarecki, "Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma," in *Proceedings of the 15th ACM conference on Computer and communications security*, 2008, pp. 449–458.

[39] J. Nick, T. Ruffing, and Y. Seurin, "Musig2: simple two-round schnorr multi-signatures," in *Annual International Cryptology Conference*. Springer, 2021, pp. 189–221.

[40] X. Chen, F. Zhang, H. Tian, B. Wei, and K. Kim, "Discrete logarithm based chameleon hashing and signatures without key exposure," *Computers & Electrical Engineering*, vol. 37, no. 4, pp. 614–623, 2011.

[41] H. Kılınç Alper and J. Burdges, "Two-round trip schnorr multi-signatures via delinearized witnesses," in *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41*. Springer, 2021, pp. 157–188.

[42] H. Krawczyk and T. Rabin, "Chameleon hashing and signatures," *Cryptology ePrint Archive*, 1998.

[43] G. Ateniese and B. de Medeiros, "On the key exposure problem in chameleon hashes," in *Security in Communication Networks: 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers 4*. Springer, 2005, pp. 165–179.

[44] M. Bellare and T. Ristov, "A characterization of chameleon hash functions and new, efficient designs," *Journal of cryptology*, vol. 27, no. 4, pp. 799–823, 2014.